令和４年度「専修学校における先端技術利活用実証研究」
（専修学校遠隔教育導入モデル構築プロジェクト）

遠隔教育によるチームプログラミング導入モデルの構築

〈事業成果報告書〉

本報告書は、文部科学省の教育推進事業委託費による委託事業として、学校法人浦山学園　富山情報ビジネス専門学校が実施した令和４年度「専修学校における先端技術利活用実証研究」の成果をとりまとめたものです。

学校法人浦山学園　富山情報ビジネス専門学校

目 次

事業成果報告 …… 5

演習ユニット …… 37

演習教材一覧 …… 38

コーディング規約 …… 111

実習ユニット …… 117

実習ユニット使用方法 …… 118

就職マッチング用ポートフォリオ作成説明書 …… 168

学習管理ユニット …… 185

チャットボット登録内容一覧 …… 186

チーム開発資料 …… 188

チーム開発方法 ……197

チーム開発評価シート ……198

チーム開発評価シート（使用例）……199

実証講座 ……201

事業評価 …… 243

令和４年度「遠隔教育によるチームプログラミング導入モデル構築」
事業の評価 …… 244

事業成果報告

令和4年度「専修学校における先端技術利活用実証研究」

遠隔教育によるチームプログラミング導入モデルの構築
事業報告

令和5年2月17日
学校法人浦山学園
富山情報ビジネス専門学校

Agenda

1. 事業の全体像
2. 令和4年度事業内容
3. 演習ユニット
4. 実習ユニット
5. 事業評価
6. 質疑応答

1. 事業の全体像

（1）専修学校における先端技術利活用実証研究

（2）専修学校遠隔教育導入モデル構築プロジェクト

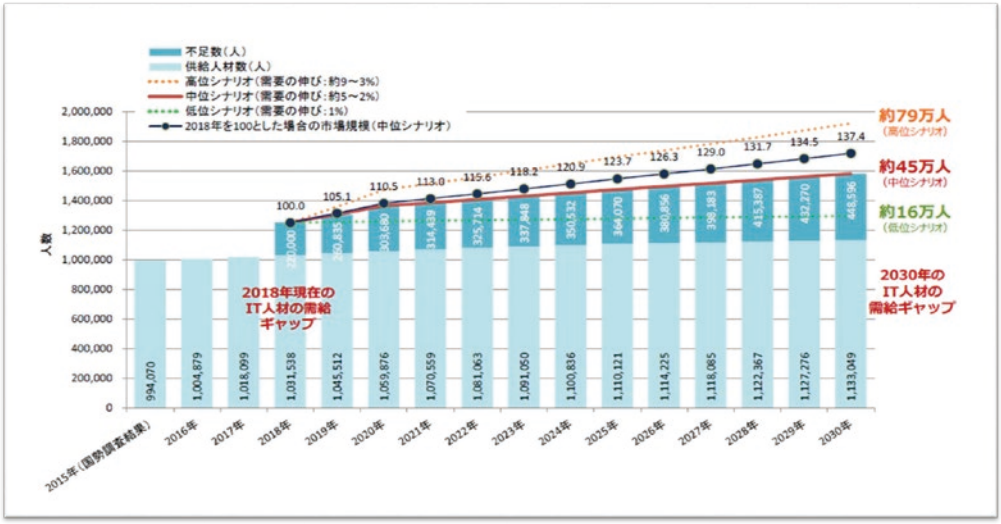
事業内容
在宅等でも、専修学校教育の質を落とすことなく提供するため、先端技術を活用した遠隔教育の実践モデルを開発し、その効果を検証する。（15箇所） 分野横断型連絡調整会議を設置し、各取組の進捗管理及び連絡調整を行ない、事業成果を体系的にとりまとめるとともに普及・定着方策を検討、展開する。（1箇所）

令和3年度からの3か年事業

（3）なぜ、プログラミング教育なのか？

要因	課題
外的要因	・プログラマ人材不足 ・プログラミング教育の必修化
内的要因	・プログラミング教育のレベル不足 ・演習授業の負担大
複合要因	・実践的インターンシップの不足 ・学生と企業との接点不足

（4）プログラマ人材不足



経済産業省　IT人材需給に関する調査　より

2030年に最大で79万人の不足

（5）小～高等学校までのプログラミング教育の必修化

小・中・高等学校別のポイント（総則及び各教科等）

- **小学校においては、文字入力など基本的な操作を習得、新たにプログラミング的思考を育成**
  各教科等の特質に応じて、児童がコンピュータで文字を入力するなどの学習の基盤として必要となる情報手段の基本的な操作を習得するための学習活動や、プログラミングを体験しながらコンピュータに意図した処理を行わせるために必要な論理的思考力を身に付けるための学習活動を計画的に実施することを明記。【総則】
- **中学校においては、技術・家庭科（技術分野）においてプログラミング、情報セキュリティに関する内容を充実**
  「計測・制御のプログラミング」に加え、「ネットワークを利用した双方向性のあるコンテンツのプログラミング」等について学ぶ。【技術・家庭科（技術分野）】
- **高等学校においては、情報科において共通必履修科目「情報Ⅰ」を新設し、全ての生徒がプログラミングのほか、ネットワーク（情報セキュリティを含む）やデータベースの基礎等について学習**
  「情報Ⅰ」に加え、選択科目「情報Ⅱ」を開設。「情報Ⅰ」において培った基礎の上に、情報システムや多様なデータを適切かつ効果的に活用し、あるいはコンテンツを創造する力を育成。【情報科】

文部科学省　新学習指導要領のポイント　より

早ければ2025年にプログラミング教育を受けた学生が専門学校へ入学してくる

（6）専門学校でのプログラミング教育の現状

課題
・基礎から教えていて、2年間で実践レベルに到らない
・学生にレベル差があって教員の負担が大きい
・出来あがったプログラムの品質が低い

要因
・失敗できない教育環境
・個別対応できない教育環境
・プログラム品質保証への意識不足

プログラミングへの興味を失う
全体的に低いプログラミングレベル

（7）産学連携課題

項目	課題
インターンシップ	・専用カリキュラム作成・実施の負担 ・セキュリティ問題
連携事業	・企業が求めるレベルへの未達
就職	・学生の実践力不足 ・実力を測る環境が無い（総合力で判断）

（8）事業目標

（a）対面と同等な教育効果を得るだけでは無く、遠隔授業の特性を活かして

・学生のレベルに応じて楽しく学習できる環境を提供する
・企業レベルの高品質なプログラミングを実現する
・IT企業も参加できる環境を提供する
・教員が簡単に導入でき、学習状況を容易に管理できる環境を提供する
・手間のかかる演習指導の負担を下げる

（b）教育環境を公開することで

・全国のIT教員が利用、交流し、プログラミング教育の向上を図る
・IT企業と学生が交流し、インターンシップ、コーオプ教育の実現、就職マッチングを実現する
・IT企業と教員が交流し、最新のIT知識、情報を教育に反映する

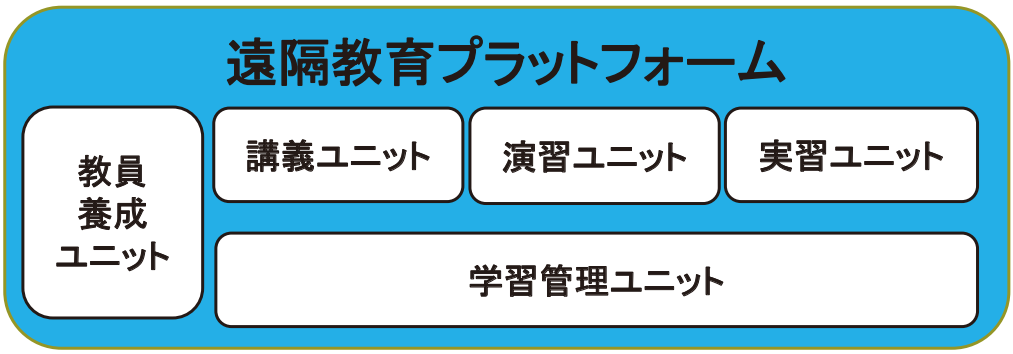
（9）事業の概要

- クラウド上に無料または安価な既存サービス、アプリケーションを組み合わせて、遠隔教育プラットフォームを構築する。

- 学生にプラットフォーム上の演習課題を取り組ませ、対面授業と同等以上の教育効果を実現しているか、企業レベルのプログラミング能力に達しているかを検証、評価する。

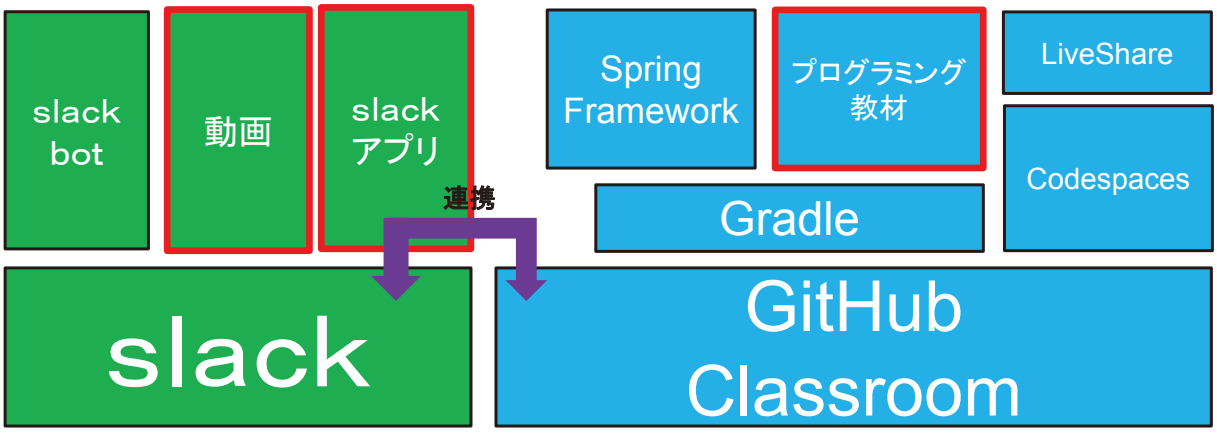
- 遠隔教育プラットフォームを全国に普及させる。

（10）遠隔教育プラットフォーム



（10）遠隔教育プラットフォーム

プラットフォームの構成



赤枠：追加開発

（10）遠隔教育プラットフォーム

各アプリケーションの役割

アプリケーション	概要	役割
slack	ビジネスチャットツール	• 教員－学生間のコミュニケーション
slackアプリ	slack上で動作する追加機能	• slack初期設定の自動化 • 課題の進捗管理 • 評価の登録
slack bot	メッセージの自動返答	• 定型質問の自動回答
GitHub Classroom	教育機能を持ったソフトウェア開発プラットフォーム	• 課題管理 • 提出課題の自動採点
Gradle	自動環境作成ツール	• 学生への開発環境の提供
Spring Framework	Java言語開発用フレームワーク	• フレームワークを使用したプログラミング環境の提供
Codespaces	クラウド上のプログラム開発環境	• 学生のプログラミング環境
LiveShare	共同プログラミングツール	• チーム開発環境

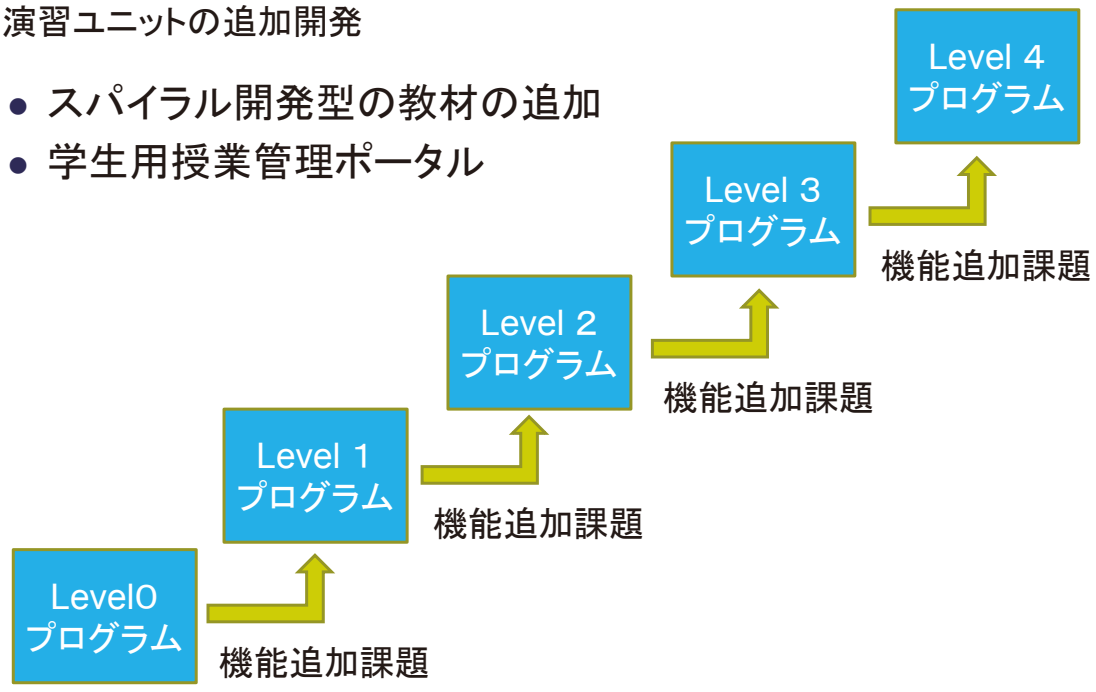
（11）事業計画

項目	令和3年度	令和4年度	令和5年度
調査	• 専門学校プログラミング教育調査 • プログラマ育成実態調査		
講義ユニット			開発 実証
演習ユニット	• 一次開発 • 実証	• 二次開発 • 実証	
実習ユニット		• 一次開発 • 実証	二次開発 実証
学習管理ユニット	• 一次開発 • 実証	• 二次開発 • 実証	三次開発 実証
教員養成ユニット			開発 実証

2. 令和4年度事業内容

（1）演習ユニットの追加開発

- スパイラル開発型の教材の追加
- 学生用授業管理ポータル



（2）学習管理ユニットの追加開発

- チャットボット機能の強化
  令和3年度の実証講座で利用が無かったため、項目を増やす共に
  AI機能を強化する
- チーム開発機能の強化
  実習ユニットでチーム開発を行うため、機能強化する

（3）実習ユニットの開発

- 企業実習（インターンシップ）対応
  企業と学生間でのプログラミング実習に対応できるユニットとして
  開発する
- チーム開発対応
  チームでプログラムを開発できる機能をサポートする
- プロトタイプ開発
  今年度はプロトタイプとして開発し検証する

（4）演習ユニットの実証

項目	内容
目的	プログラミング学習システムとして求められる学習効果、利便性があるか確認する
実施場所	3専門学校（遠隔）
方法	学習過程や成果物によって取得するアセスメント、コメント、開発速度、プログラム品質、プログラム稼働状況等の項目について比較検証する。

（5）実習ユニットの実証

項目	内容
目的	企業実習として求められる学習効果、利便性があるか確認する
実施場所	専門学校（対面、遠隔）
方法	企業から提示されたプログラミング課題を対面、遠隔方式で取り組み、学習過程や成果物によって取得するアセスメント、コメント、プログラム品質、プログラム稼働状況等の項目について検証する。

3. 演習ユニット

（1）演習ユニットの追加開発

- WebアプリケーションをMVCモデルを適用したSpringフレームワークを使用した教材を追加する
  （MVCモデル：開発現場で一般的に利用されている開発モデル）
- 提供されたベースプログラムを参考に段階的に機能追加を行う
- プログラミング言語はJava言語を使用する
- 具体的には学生用授業管理ポータル画面に各種機能を追加する

（1）演習ユニットの追加開発

学生用授業管理ポータル（Level0）

（1）演習ユニットの追加開発

レベル	内容
level1-1	ユーザー名の表示
level1-2	今日の天気の表示
level2-1	時刻別の挨拶表示
level2-2	本日の運勢占い
level2-3	本日の授業表示
level3-1	学習成果用テーブル追加
level3-2	学習成果表示
level3-3	学習成果登録
level3-4	学習成果変更
level3-5	学習成果削除
level3-6	学生成果ポータル表示
level4	ToDoリスト機能作成

（1）演習ユニットの追加開発

学生用授業管理ポータル（Level4）

（1）演習ユニットの追加開発

教材例

【Level1-2】
今日の天気を表示（API利用）：都市を変更

- **テーマ**
  - Serviceクラスの取り扱いについて
  - 天気予報APIの利用について
- **演習課題の準備（開発環境について）**
  - この演習では、Codespacesと呼ばれるサービスを使って開発を行います。
  - ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。
  - レベル0で実施した手順を参照して、演習課題の準備をしましょう。
- **演習内容**
  - Serviceの一部を書き換えて自分の住んでいる都市の天気を表示しましょう。

【Level3-2】
学習成果表示

- **テーマ**
  - 既存プログラムを参考に、CRUD機能が作成できること ページ遷移とバリデーションの仕組みが理解できること templateへの記載ができるようになること
- **演習課題の準備（開発環境について）**
  - この演習では、Codespacesと呼ばれるサービスを使って開発を行います。
  - ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。
  - レベル0で実施した手順を参照して、演習課題の準備をしましょう。
- **演習内容**
  - Lesson(授業管理機能)を参考にArchievement(実績管理機能)の実績一覧画面を作成しましょう。

（2）演習ユニットの実証

- 演習教材を遠隔で個人で実施する
- 実証期間と参加人数
  - A専門学校（以下、A校）
    - 日時：10/24 - 10/27（4日間）
    - 参加者：26名
  - B専門学校（以下、B校）
    - 日時：11/14 - 11/18 （5日間）
    - 参加者：16名
  - C専門学校（以下、C校）
    - 日時：11/28 - 12/2 （5日間）
    - 参加者：44名

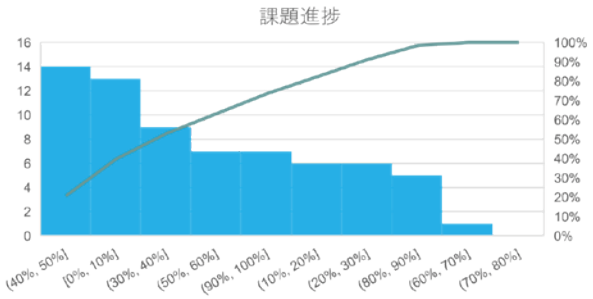
（2）演習ユニットの実証

- 事前に現地で環境設定と環境利用方法を説明
  - 実証講座内容の説明
  - 遠隔教育プラットフォームへのユーザー登録
  - 遠隔教育プラットフォームの利用方法説明
  - ベースプログラムの説明

（2）演習ユニットの実証

- 評価対象者
  - 実証講座用に払い出したアカウントが有効 かつ、
    実証講座後のアンケートに回答があった学生を対象
  - 対象：A校 26名、B校 15名、C校 27名

- 全体進捗

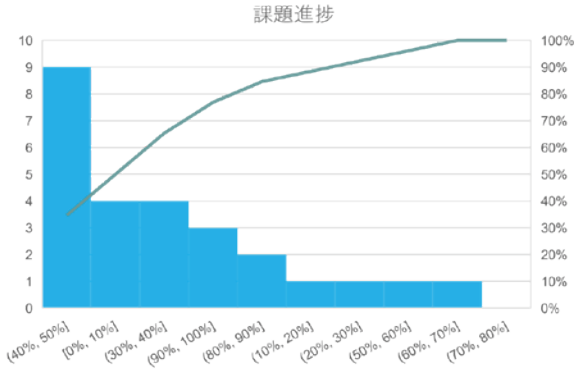


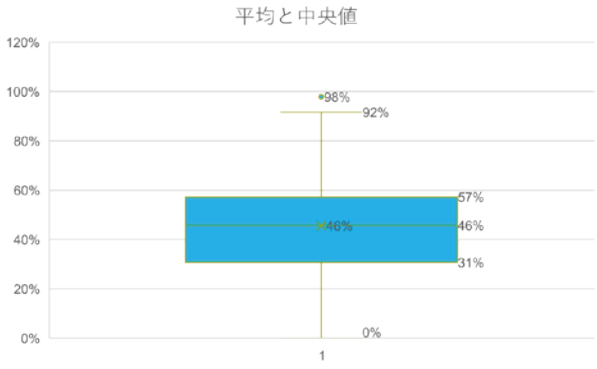
・40-50%（Level3-1まで完了）の進捗が
最も多かった

→ 多くの学生が Level3-1で頭打ちになった

（2）演習ユニットの実証

- A専門学校

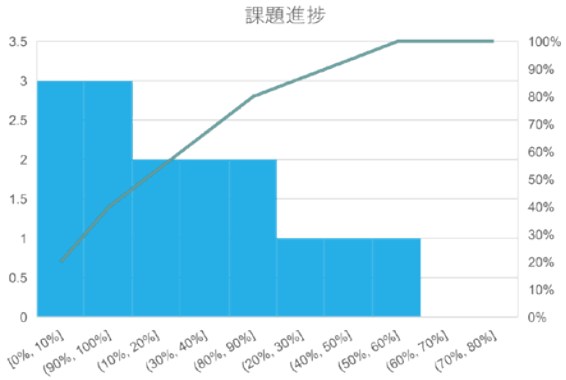


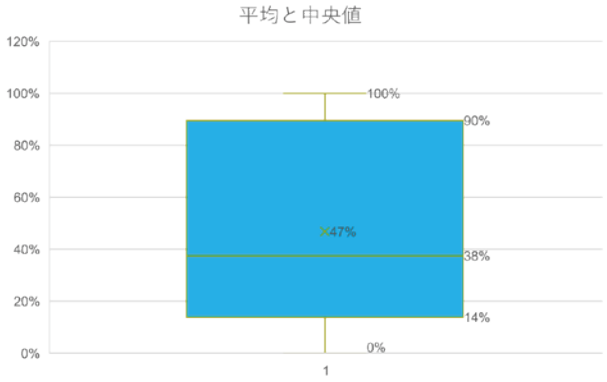


平均：46%　中央値：46%

（2）演習ユニットの実証

- B専門学校

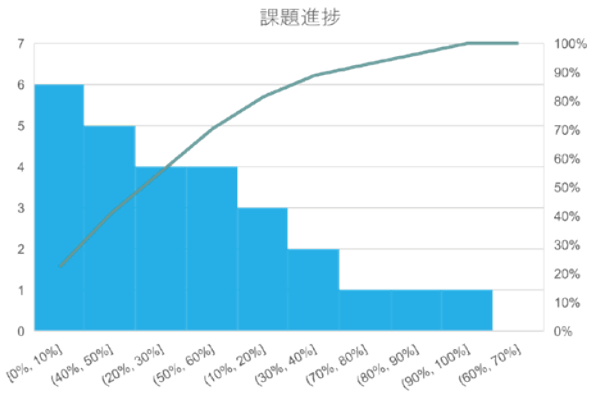


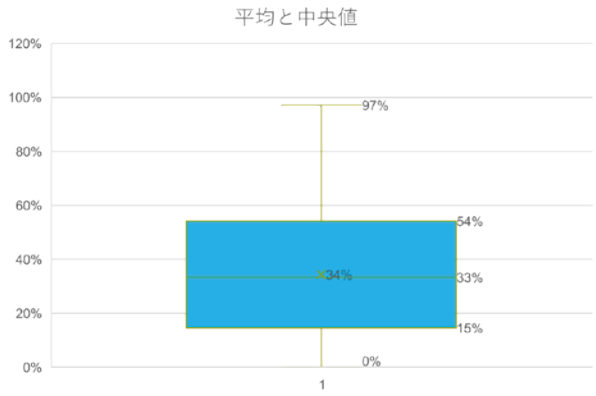


平均:47%　中央値:38%

（2）演習ユニットの実証

- C専門学校





平均:34%　中央値:33%

（2）演習ユニットの実証

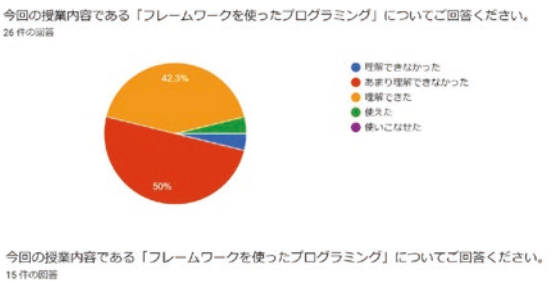
- 各学校の進捗平均
  - A校：46%
  - B校：47%
  - C校：34%

- C校の進捗がA校、B校と比べて低い結果となった

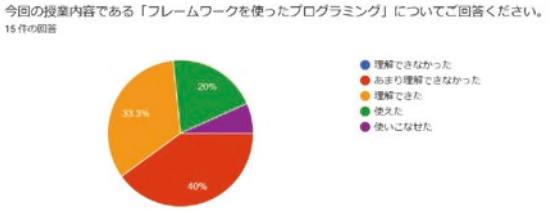
（2）演習ユニットの実証

課題の理解度に差があったかを確認
（アンケートでの回答で算出）

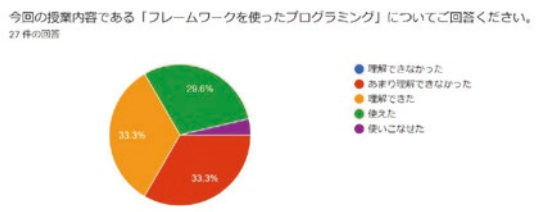
- A校



- B校



- C校



（2）演習ユニットの実証

- 演習ユニットの理解度について
  - 理解できなかったと回答
    - A校(53%) > B校(40%) > C校(33.3%)
  - 進捗
    - A校(46%) ≒ B校(47%) > C校(34%)
  - アンケート結果の理解できなかった人の割合をみたとき、A校が一番多く、C校が一番少なかった
  - 結果、今回理解度は進捗に影響しなかったと考えられる

（2）演習ユニットの実証

- 実証講座に取り組んだ時間の差があったか
  - 授業、自習で取り組んだ時間の平均算出
    - A校 ： 18.4時間
      - 授業で3コマ*4日間分 + 自習時間（アンケートで回答）
    - B校 ： 6.1時間
      - 授業時間（アンケートで回答）+ 自習時間（アンケートで回答）
    - C校：6.2時間
      - 授業時間（アンケートで回答）+ 自習時間（アンケートで回答）
  - B校とC校の学習時間は変わらないが進捗に差が出てた

（2）演習ユニットの実証

- 学習方法について
  - 学習時の状況
    - C校：2年生37名については2コマ授業時間で実施
      - 6時間のうち半分は授業時間（受動的）
    - B校：授業時間の確保はなし（授業中の自己学習時間の利用はOK）
      - B校の学生は能動的に6時間を確保し課題を進めていた
  - この受動的か能動的であったか差が課題への進捗に影響したと推測
    - 能動的 = モチベーションが高い

→ 学習時の環境と状況を確認すると大きく異なっていた

（2）演習ユニットの実証

- 現地教員の対応
  - A校
    - 毎日、開始時と終了時に声掛けを実施
  - B校
    - 担当の先生より、学生への動機づけを促す説明
      - 新しいことを学べる＋実社会ではこういった環境で仕事をするから価値があるということを伝達
      - 強制はせずできる範囲でやることを目標とさせた
  - C校
    - 2年生：別課題の締め切りが近かった（そのため、あまり強制はしていない）
    - 1年生：学生に委ねる形で放置

- 教育PFと演習課題を取り組ませる上で、学習時の環境が影響
  →学生への動機付けや声掛けと、取り組みやすい環境を
  用意する必要がある

（2）演習ユニットの実証

- Q＆Aは学生からチャット、もしくはチャットボットにコメントされたものを対象
  - 感謝・報告・承知等のやり取りを除くものを抜粋
  - B校とC校では、準備の段階でチャットボットを利用するようアナウンスし、チャットでも改めて周知
- 結果
  - A校
    - 32件（うちチャットボット0件）
  - B校
    - 9件（うちチャットボット0件）
  - C校
    - 6件（うちチャットボット2件）

（2）演習ユニットの実証

- チャットボットの利用が少ない
  - 要因
    - 登録回答が少ない
    - 学生がチャットボットへの質問利用に慣れていない
  - 対応
    - 今回の実証で学生から出たQ&Aを登録する
    - チャットボット利用のチュートリアル作成を検討する

（2）演習ユニットの実証

- 実施後アンケートの意見
  - フレームワークの利用は開発スピードが速い
  - 参考プログラムがあるので自分で進められる
  - 自分のペースで進められる
  - タイミングを気にせず質問ができる
  - 自分で考える力が付いた

  - トラブル解消に時間がかかる
  - 全体への質問になって、質問しづらい
  - 文章で質問するのが面倒
  - 本当に自分の実力になったのかが不安

4. 実習ユニット

（1）実習ユニットの開発

開発機能

- 企業別課題管理
- チーム内コミュニケーション管理
- プログラム品質管理
- プログラム評価
- 就職マッチング

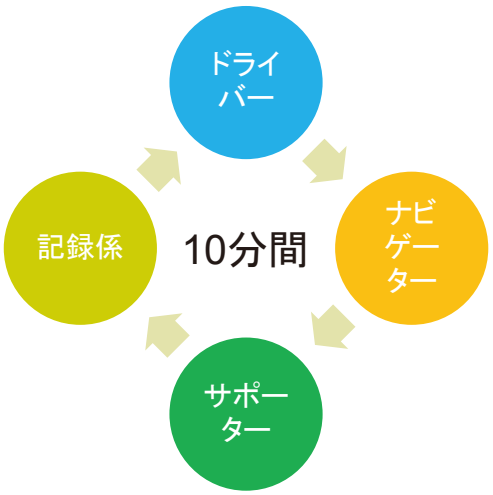
運用方法

- LiveShareを利用してプログラム共有する
- チーム開発にはモブプログラミング（モブプロ）を利用する

（1）実習ユニットの開発

モブプロ（モブプログラミング）とは

- ルール
  - 各メンバーに役職を割り振り、時間を区切って役職のローテーションを行いプログラミングを実施
- 役職
  - ドライバー：ナビゲーターの指示を受けてプログラミングを進める。
  - ナビゲーター：プログラマーにプログラミング内容の指示を出す。
  - サポーター：ナビゲーターがつまるところなど適宜アドバイスする。
  - 記録係：各順の進捗や議論時間についての記録を行う。



（1）実習ユニットの開発

モブプロ（モブプログラミング）とは

- 具体的な実施方法
  - Live Share を用いてドライバーの作業PCの内容を全メンバーで共有、同時操作できる状態で実施
- 目指す効果
  - 本来の目的はレベルの高い人の指示内容をレベルの低い人が吸収することで全体のレベルアップを図る
  - レベル差のあるメンバーが強制的にグループワークに参加させられることにより、グループワークにおける能力を測定することができる

（2）実習ユニットの実証

- 概要
  - 対面と遠隔のそれぞれで、実習ユニットを使ったチーム開発をモブプロで行い、同等の結果が得られるかの検証を行う
- 内容
  - チーム単位で演習課題を実施
  - 演習課題は、演習実証と同じものを使用
  - 遠隔チームには事前説明の時間を設ける
  - 対面と遠隔で20名ずつに分け、その中で1チーム4人を編成
    （対面：5チーム、遠隔：5チーム）

（2）実習ユニットの実証

- 実施方法
  - 実習課題
    - 演習ユニットで追加開発した教材を使用
  - 遠隔
    - 演習ユニットと同様に、事前に現地で環境設定と環境利用方法を説明
  - 対面
    - 大型ディスプレイを用意。ドライバーの作業PCの内容を映し出しながら、全員で同時に確認できるようにする
    - 各タームごとに席替えを行い、PCを使いまわす形で操作を行う

（2）実習ユニットの実証

- 遠隔
  - 1日目（90分）：準備、説明、各自でLevel0を確認、モブプロ体験
  - 2日目：各自でLevel1と2を実施
  - 3-5日目：最低3時間 level3（モブプロ）
- 対面
  - 1日目(3時間)：準備、説明、各自でLevel0-2を実施
  - 2日目(3時間)：Level3 （モブプロ）

（2）実習ユニットの実証

課題に取り組んだ時間と進捗

- 遠隔：平均4.1時間
  - チーム1：4.2時間（Level3-2到達）
  - チーム2：3.0時間（Level3-2到達）
  - チーム3：3.3時間（Level3-3到達）
  - チーム4：3.0時間（Level3-2到達）
  - チーム5：6.8時間（Level3-3到達）
- 対面：平均2.1時間
  - チーム1：2.0時間（Level3-2到達）
  - チーム2：2.0時間（Level3-2到達）
  - チーム3：2.3時間（Level3-2到達）
  - チーム4：2.1時間（Level3-2到達）
  - チーム5：2.2時間（Level3-2到達）

（2）実習ユニットの実証

- 議論時間（遠隔）
  - ナビゲーターの指示のもと、作業できていた割合
    - プログラムの入力指示、サポータへの調査依頼など、課題に関する会話ができていた時間
    - 沈黙等があるとマイナス

チーム名	平均	1巡目	2巡目	3巡目	4巡目	5巡目	6巡目	7巡目	8巡目	9巡目	10巡目	11巡目	12巡目	13巡目	14巡目	15巡目	16巡目	17巡目	18巡目
1	**83%**	80%	100%	60%	100%	60%	100%	80%	100%	80%	80%	80%	60%	80%	100%				
2	**84%**	30%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	80%	50%	50%	50%	50%
3	**89%**	90%	80%	100%	80%	90%	90%	100%	80%										
4	**83%**	70%	80%	90%	90%														
5	**84%**	100%	90%	100%	80%	90%	80%	100%	80%	80%	80%	70%	80%	80%	70%				

(2)実習ユニットの実証

- 議論時間(対面)
  - ナビゲーターの指示のもと、作業できていた割合
    - プログラムの入力指示、調査依頼、ソースに関する相談等ができていた時間
    - 沈黙等があるとマイナス

チーム名	平均	1巡目	2巡目	3巡目	4巡目	5巡目	6巡目	7巡目	8巡目	9巡目	10巡目	11巡目	12巡目	13巡目
1	**74%**	100%	50%	70%	70%	100%	60%	90%	60%	70%	70%	70%	80%	
2	**100%**	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
3	**90%**	80%	70%	80%	90%	100%	100%	100%	90%	100%				
4	**75%**	50%	80%	90%	90%	80%	80%	20%	80%	80%	80%	80%	80%	80%
5	**72%**	60%	80%	90%	80%	80%	50%	70%	70%	80%	80%	90%	50%	50%

(2)実習ユニットの実証

- 議論時間の平均をとると、遠隔が84%と対面が82%であまり差がない結果となった。この結果、実施する環境の違いがグループワークに影響しないと考えられる。
- 各課題の完了数、品質、開発速度の計測ができなかった
  - 対象者(1年生)に対する課題のレベルが高く十分なデータを計測するに至らなかった
- チーム全体のレベルアップ効果は認められなかった
  - 課題が進む際に難易度が上がるためレベルアップ効果が見えづらかった

- 実習ユニットで使用する課題内容を検討する

（2）実習ユニットの実証

- 学生からの意見
  - 協調性の大切さを再認識した
  - 判らなくなってもモチベーション維持ができた
  - 自分に向いている役割、苦手な役割を知ることができた
  - 自分とは異なる視点を知ることができた
  - 一人よりも出来た時の達成感が強い

  - 話すことに気を取られて細かい点に気付かないことがあった
  - トラブル解消に時間がかかる
  - メンバー全員の実力が低い場合に課題解決に到らない

5. 事業評価

（1）達成状況と成果

NO	作業概要	達成状況	評価
1	完成版演習ユニットの完成	○	○
2	学習管理ユニットの第2次開発	○	○
3	実習ユニットプロトタイプ開発	△	△
4	演習ユニットおよび学習管理ユニットの実証	○	○
5	実習ユニットの実証	○	△

- 演習教材はMVCモデルを用いた実践的な教材となっている
- 演習教材を複数の学校で検証し、多くの知見を得ることができた
- 実習ユニットはLiveShareとモブプロを使用することにより、遠隔でも円滑なチーム開発が可能であることを証明した

（2）課題と対応

- Level3-2でつまづかないような教材の工夫が必要である
  →講義ユニットでの教材開発を検討する
- 実習ユニットでの検証が不十分である
  →使用教材を工夫する
  →品質、開発時間を計測する仕組みを検討する

7. 質疑応答

ご清聴ありがとうございました

演習ユニット

演習教材一覧

レベル	課題内容
Level0	ベースプログラム（参照用） 学生用授業管理ポータル画面
Level1-1	ユーザー名表示機能追加
Level1-2	今日の天気表示機能追加
Level2-1	時刻別の挨拶表示機能追加
Level2-2	本日の運勢占い機能追加
Level2-3	本日の授業の表示機能追加
Level3-1	学習成果用テーブル追加
Level3-2	学習成果表示機能追加
Level3-3	学習成果登録機能追加
Level3-4	学習成果変更機能追加
Level3-5	学習成果削除機能追加
Level3-6	学生成果ポータル画面表示機能追加
Level4	ToDo リスト機能追加

【Level0】既存プログラムを理解しよう

テーマ

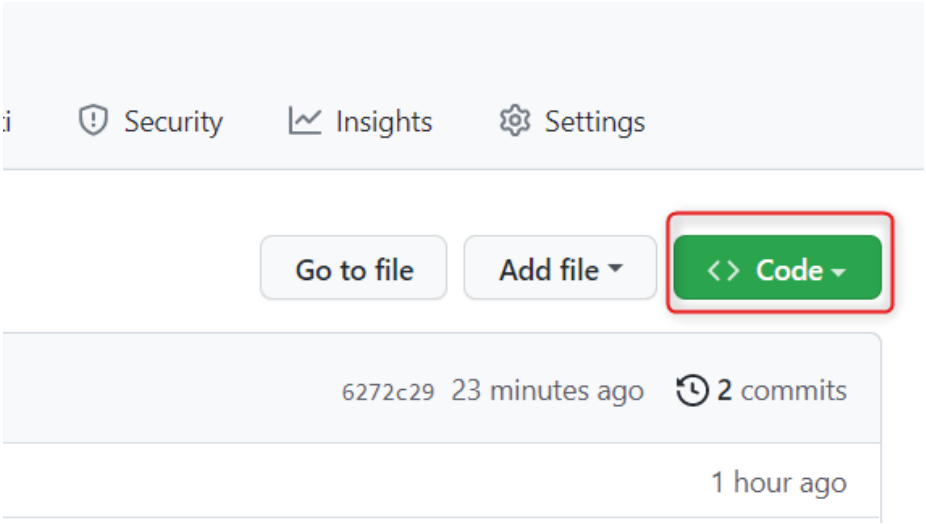
1. MVC モデルについて理解しよう
2. Spring を使った開発について理解しよう
3. データベースの利用について(H2 Database)

開発環境について

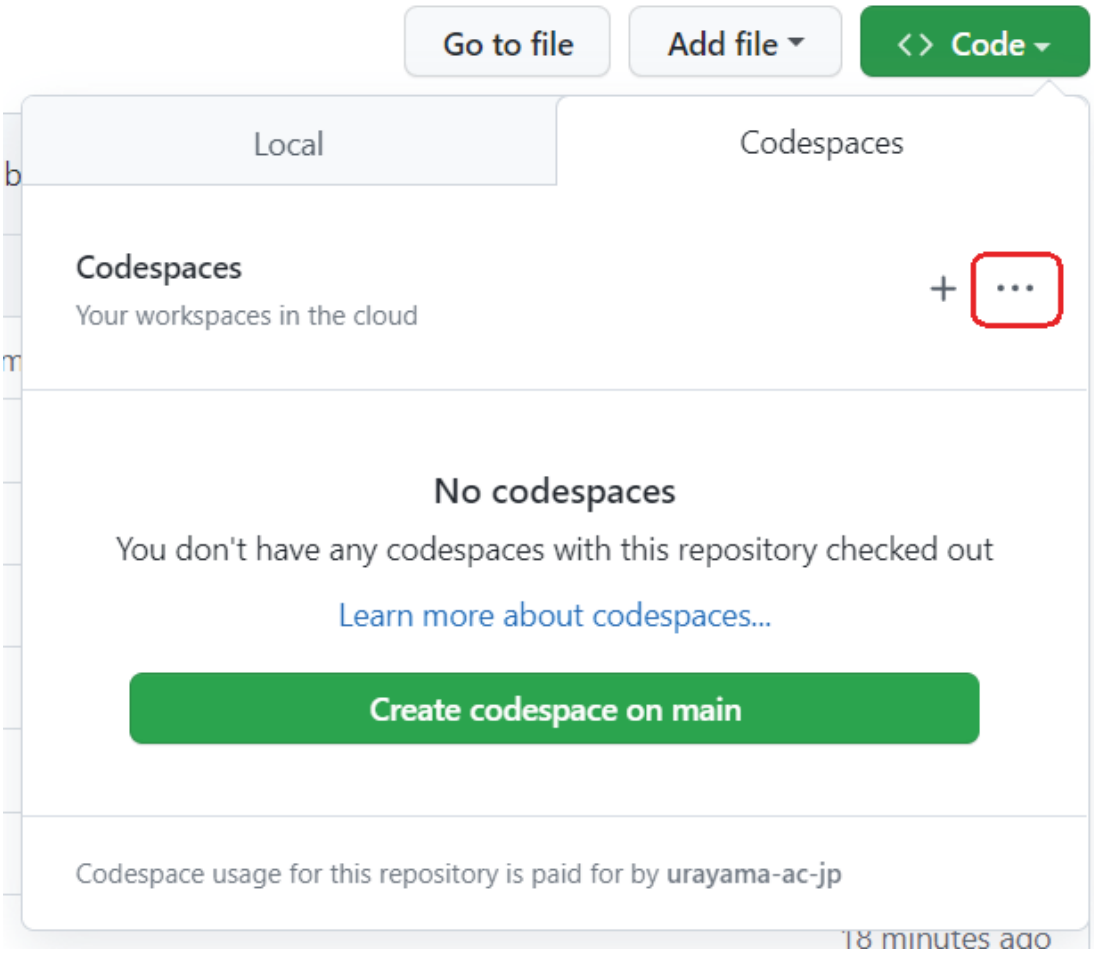
- 本演習では、Codespaces と呼ばれるサービスを使って開発を行います。
- ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。

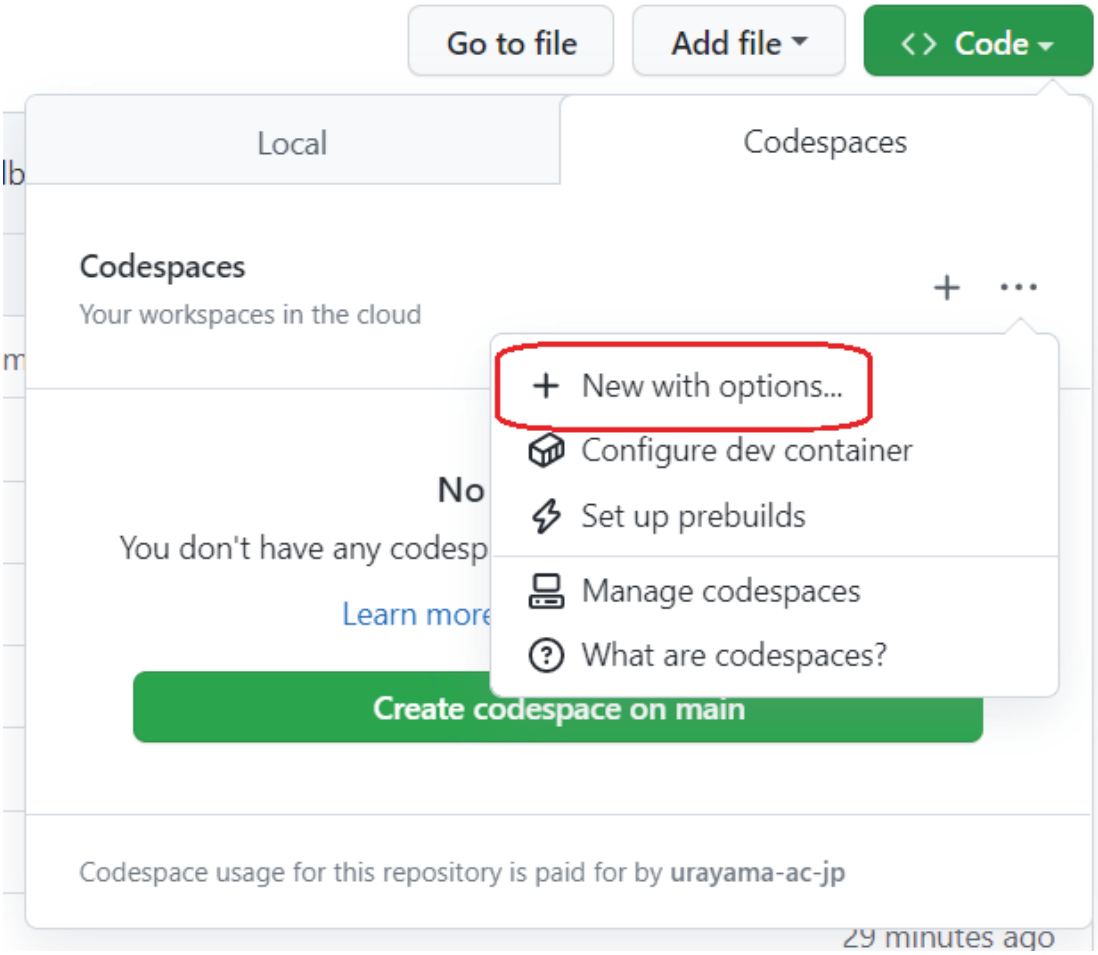
まずは動かしてみよう

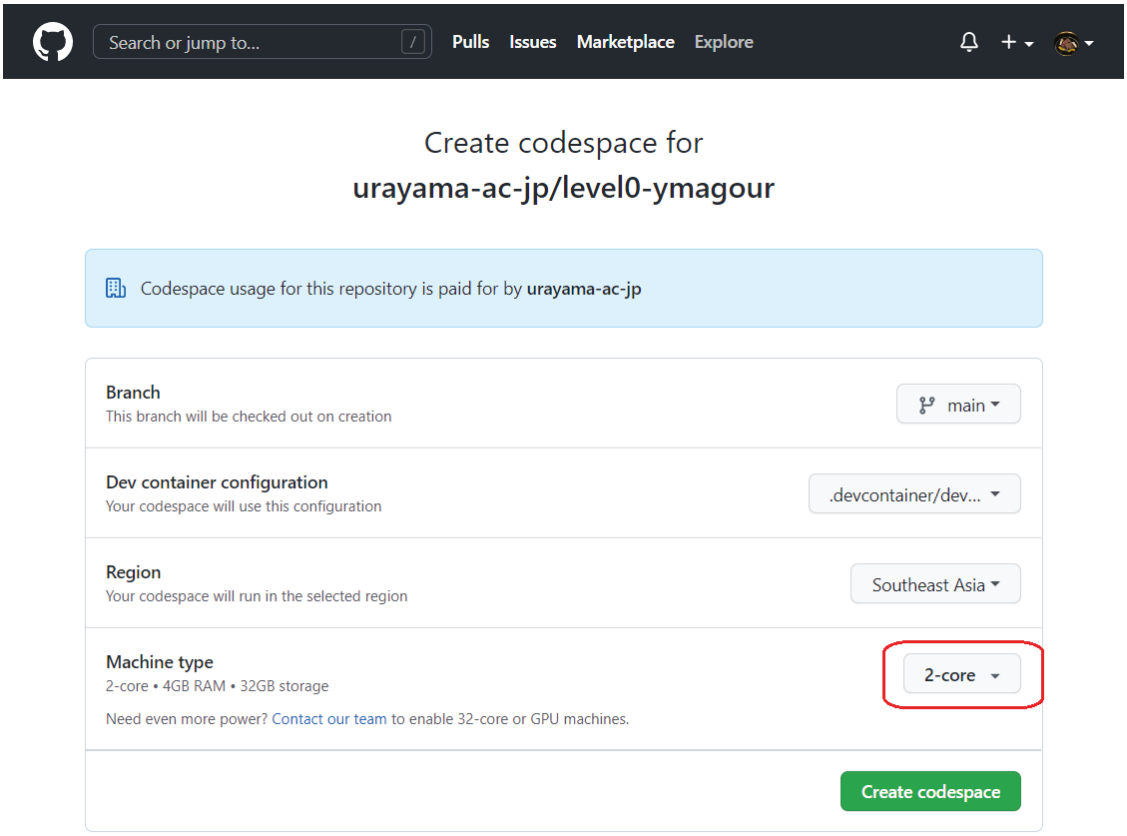
1. Code をクリックします。

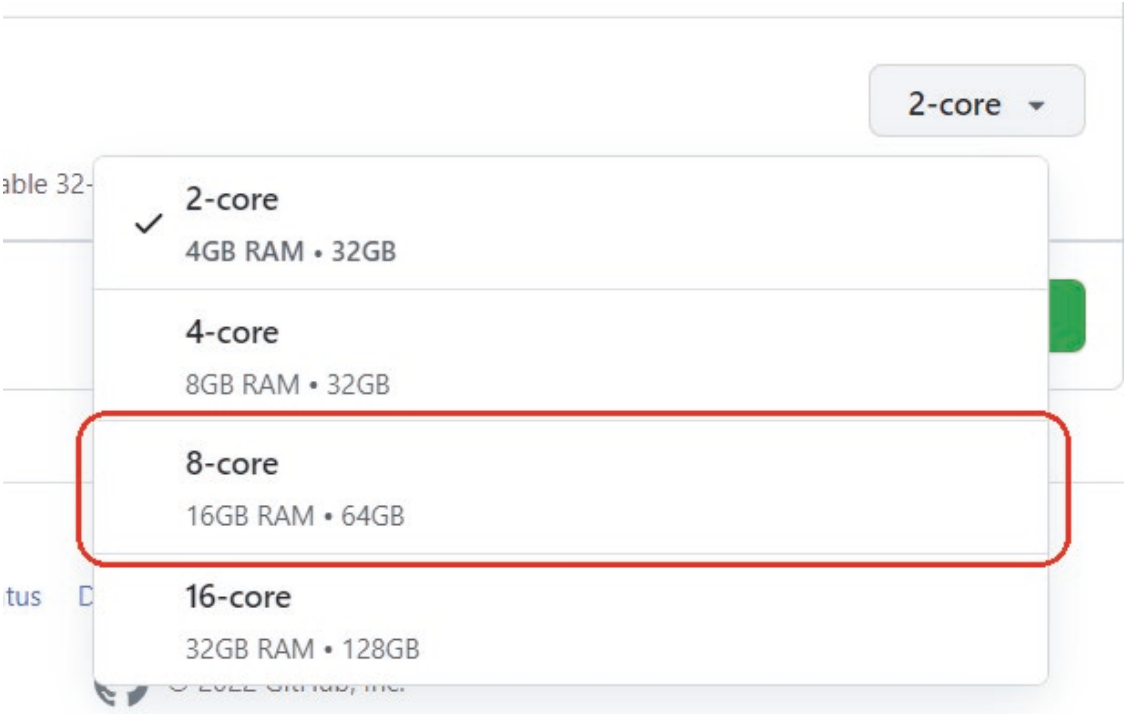


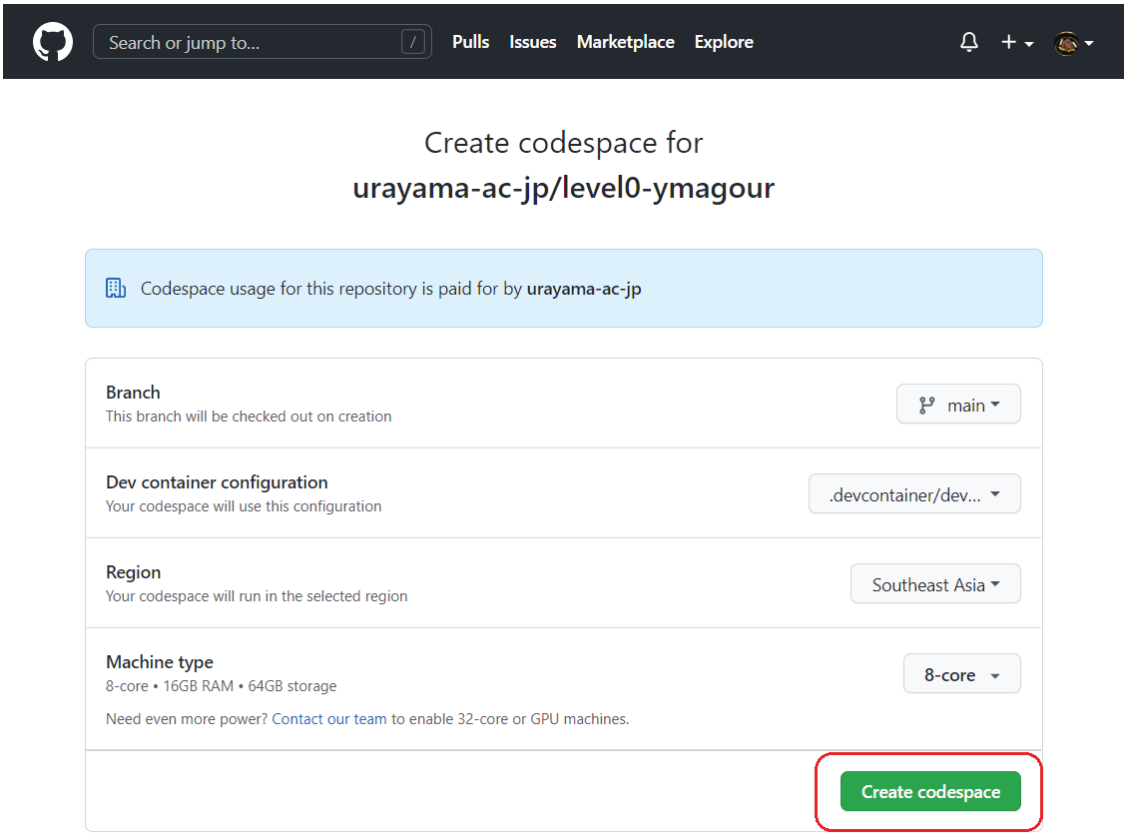
2. 矢印で示した箇所を押してください。
   初回と、2 回目以降では表示が異なりますので注意してください。
   【初回】


Go to file
Add file
Code
Local
Codespaces
Codespaces
Your workspaces in the cloud
No codespaces
You don't have any codespaces with this repository checked out
Learn more about codespaces...
Create codespace on main
Codespace usage for this repository is paid for by urayama-ac-jp

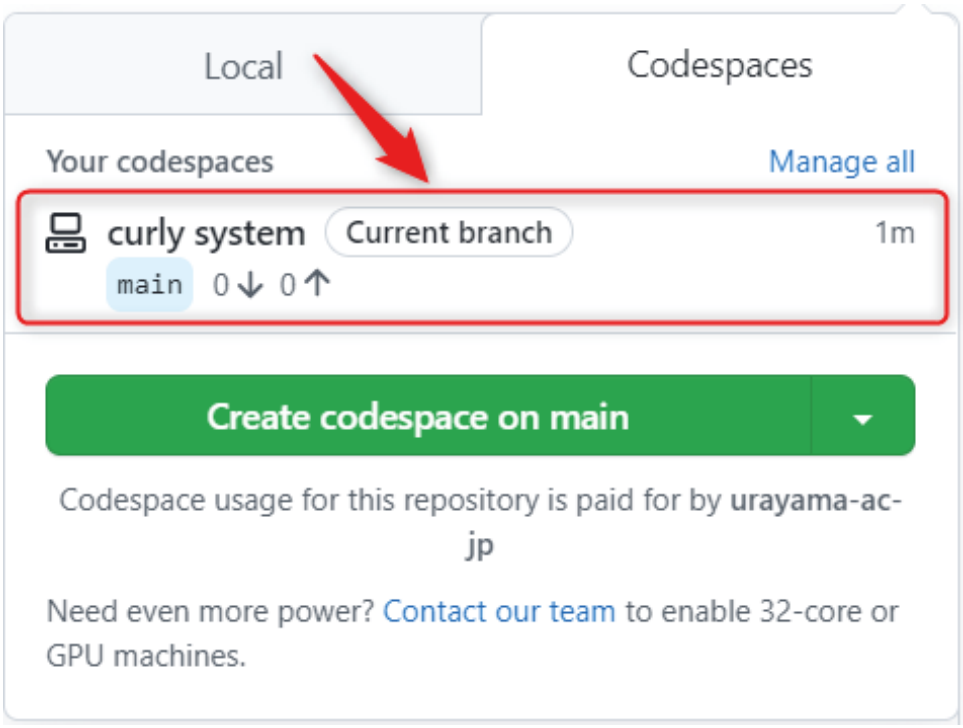

Go to file
Add file
Code
Local
Codespaces
Codespaces
Your workspaces in the cloud
New with options...
Configure dev container
Set up prebuilds
Manage codespaces
What are codespaces?
Create codespace on main
Codespace usage for this repository is paid for by urayama-ac-jp


Search or jump to...
Pulls
Issues
Marketplace
Explore
Create codespace for
urayama-ac-jp/level0-ymagour
Codespace usage for this repository is paid for by urayama-ac-jp
Branch
This branch will be checked out on creation
main
Dev container configuration
Your codespace will use this configuration
.devcontainer/dev...
Region
Your codespace will run in the selected region
Southeast Asia
Machine type
2-core • 4GB RAM • 32GB storage
Need even more power? Contact our team to enable 32-core or GPU machines.
2-core
Create codespace

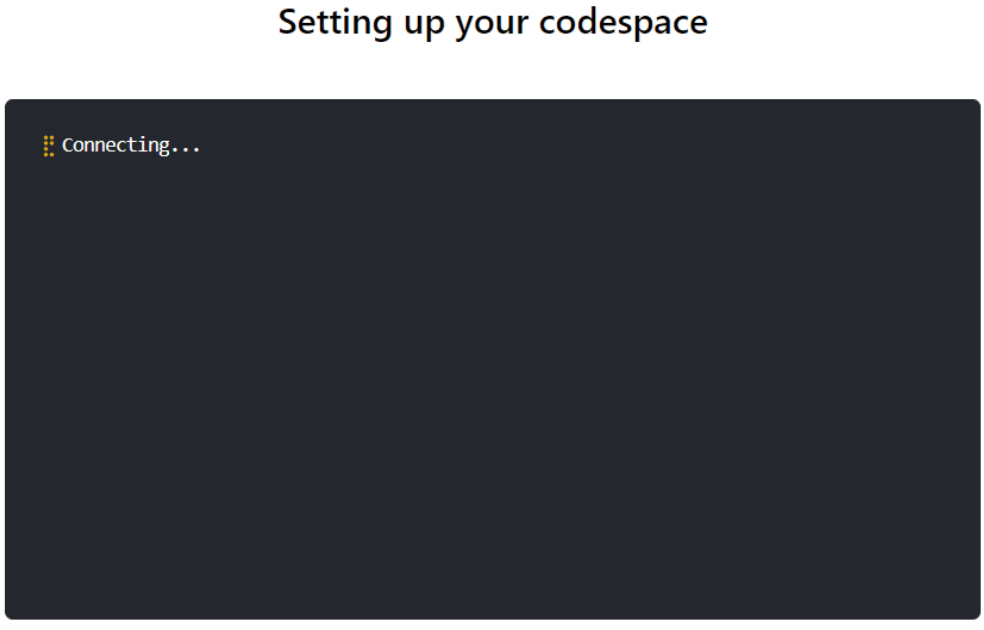

2-core
2-core
4GB RAM • 32GB
4-core
8GB RAM • 32GB
8-core
16GB RAM • 64GB
16-core
32GB RAM • 128GB


Search or jump to...
Pulls
Issues
Marketplace
Explore
Create codespace for
urayama-ac-jp/level0-ymagour
Codespace usage for this repository is paid for by urayama-ac-jp
Branch
This branch will be checked out on creation
main
Dev container configuration
Your codespace will use this configuration
.devcontainer/dev...
Region
Your codespace will run in the selected region
Southeast Asia
Machine type
8-core • 16GB RAM • 64GB storage
8-core
Need even more power? Contact our team to enable 32-core or GPU machines.
Create codespace

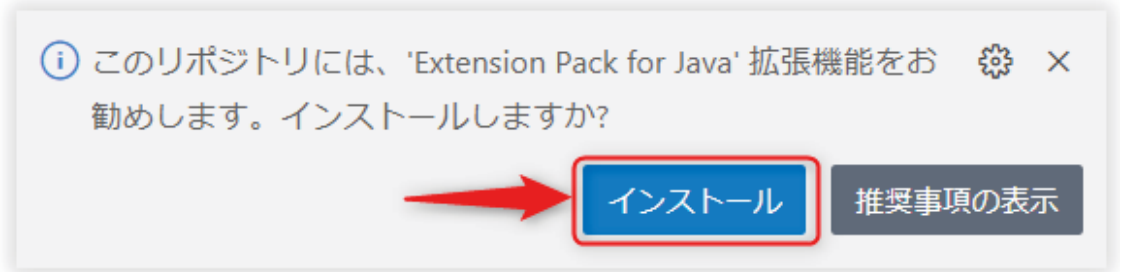
【2 回目以降】



1. しばらく待ちます。



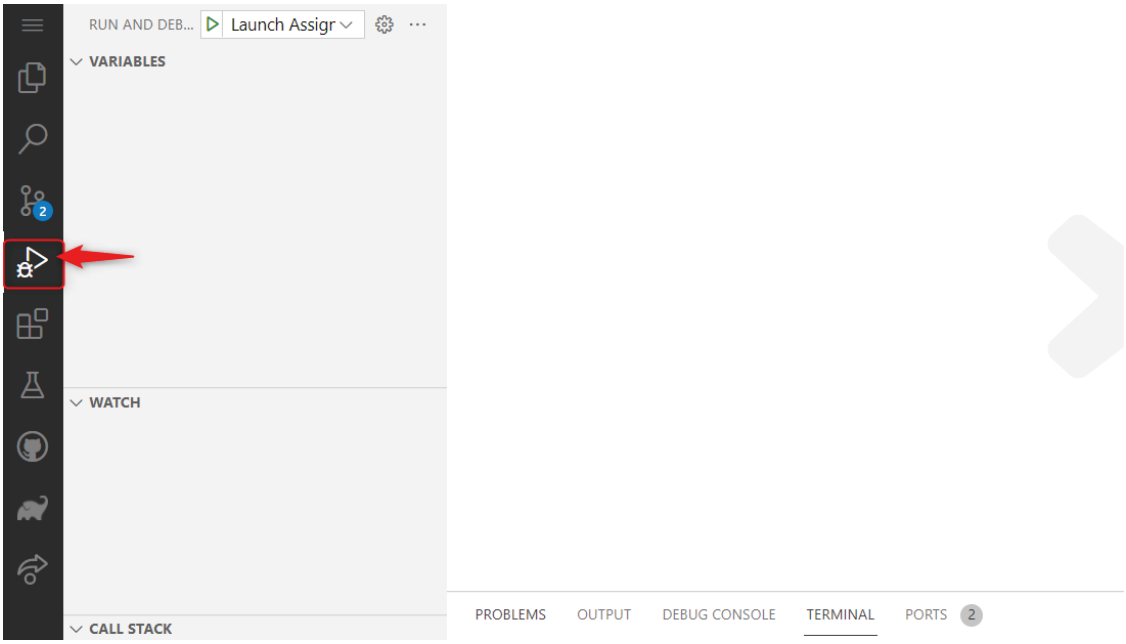
2. 画像のダイアログが表示されたら、インストールを押してください。



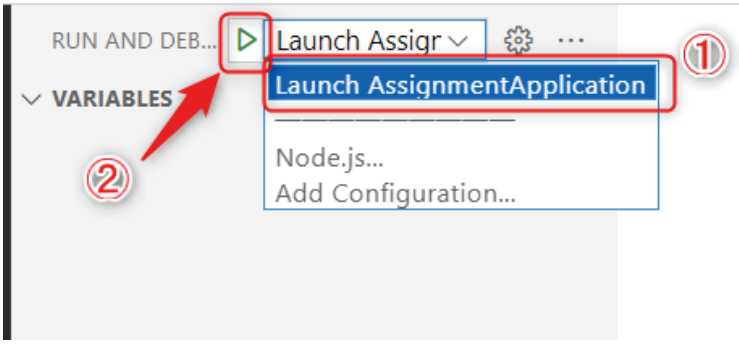
3. インストールボタンを押すと、新しいタブが開きます。緑で囲んだ部分が無効にするになったらタブを閉じましょう。

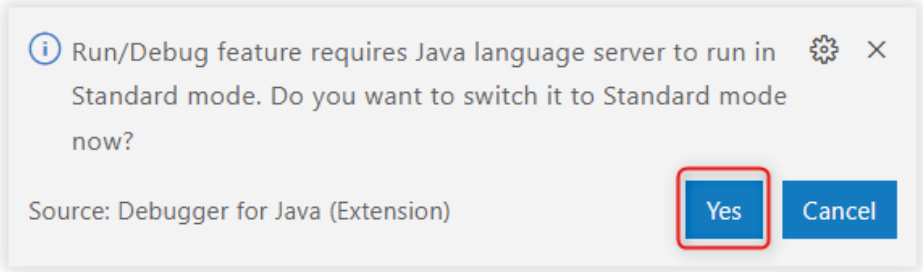
4. 作業の開始タブも閉じましょう。

5. 実行とデバッグアイコンを押します。

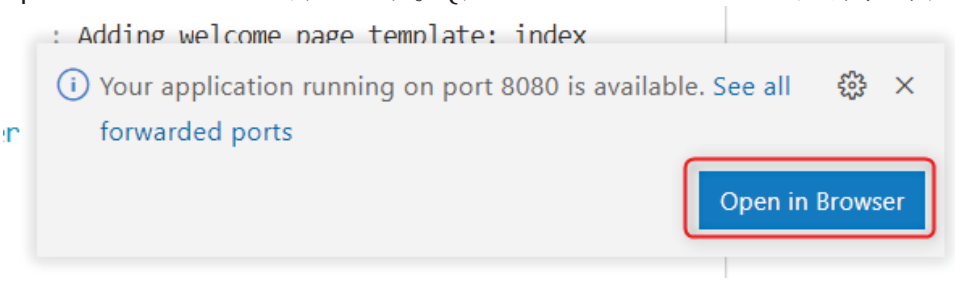


6. `Launch AssignmentApplication` が選択されていることを確認し再生ボタンを押します。 初回はダイアログが表示されます、YES を選択してください。





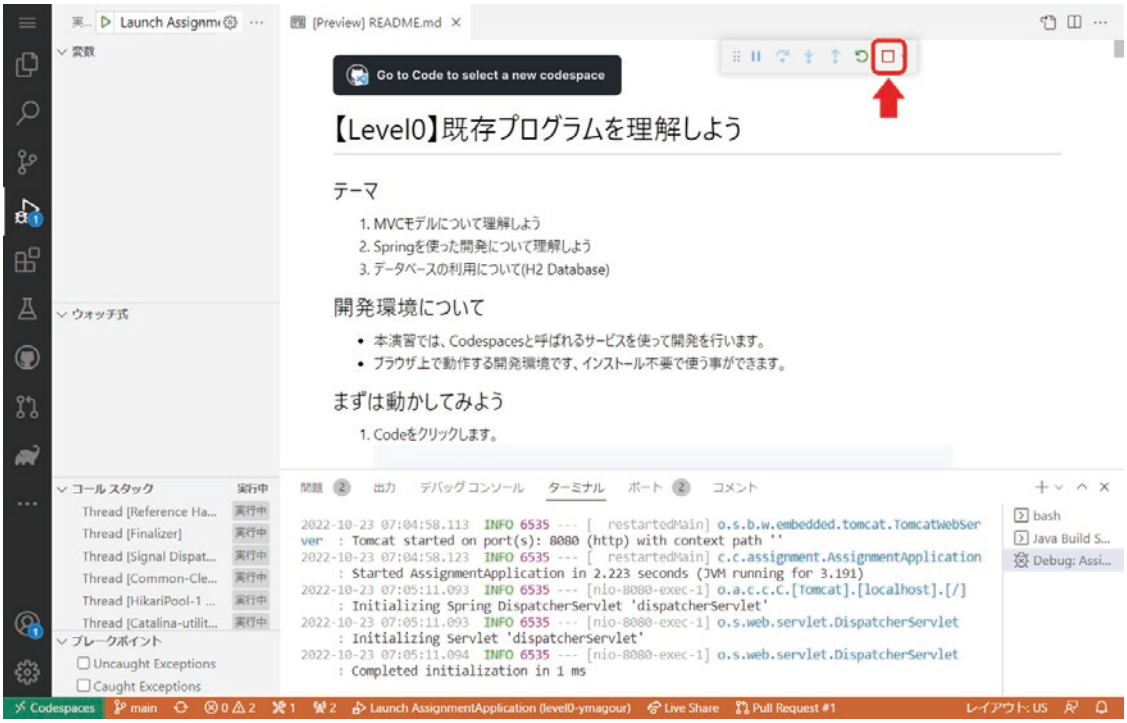
7. `Open in Browser` を押します。(表示されるまでしばらく時間が掛かります。)



8. Web ページが表示されます。

9. 確認が終わったら、タブを閉じましょう

10. デバック実行中ですので、停止ボタンを押して、デバック実行を停止しましょう。



MVC モデルについて理解しよう

MVC は、今回皆さんに学んで頂く Spring Boot などの Web フレームワークで一般的に取り入れられる Web アプリケーション構築を整理するための概念の一つです。 3 つの要素を組み合わせる事で、Web アプリケーションを構築します。 この 3 つの要素というのは、モデル（Model）、ビュー（View）、コントローラー（Controller）の事で、これらの頭文字を取って MVC と呼ばれています。

モデルとは

Web アプリケーション内のビジネスロジックを担当します。

ビューとは

ブラウザ上で表示可能な HTML を構築し、クライアントへレスポンスを送信します。

コントローラーとは

モデルとビューの橋渡しを行っています。

今回のプログラムにおける MVC

モデル

Service クラスや Dao クラスを指します。

ビュー

Thymeleaf などのテンプレートエンジンを利用して、HTML を生成する部分を指します。

コントローラー

Controller クラスを指します。

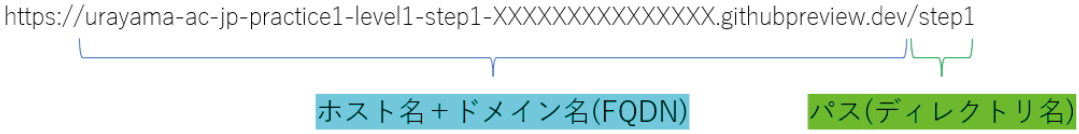
ページが表示される仕組みについて

授業管理ページを表示する際の動作を例に説明します。 1. ユーザー：Lesson ボタンを押します。 1. ブラウザ：Lesson ボタンが押されたので、パス`/lesson` に GET リクエストを送信します。 1. Web サーバー(Spring Boot)：`/lesson` に GET リクエストが来たので、対応するコントローラーである `LessonController` クラスの `index` メソッドに処理を任せます。 1. コントローラー：サービスである `LessonService` クラスの `selectAll` メソッドを使って、授業一覧データを取得しようとします。 1. サービス：Dao である `LessonDao` クラスの `selectAll` メソッド使って、授業一覧データを取得します。 1. Dao：データベースから授業一覧データを取得し、サービスへ返します。 1. サービス：Dao から受け取った授業一覧データをコントローラーへ返します。 1. コントローラー：受け取った授業一覧データと使用するテンプレート名をテンプレートエンジンに渡します。 1. テンプレートエンジン：指定されたテンプレートと授業一覧データを組合せて HTML を生成します。 1. Web サーバー(Spring Boot)：ブラウザに対して生成された HTML を返します。 1. ブラウザ：受け取った HTML を表示します。

解説

コントローラーの動き

- WEB ページのリンクボタン(他のページに移動するボタン)を押すと、ブラウザからリクエストが送信されます。
- リクエストには移動先のページの URL が含まれています。
- URL の例



- この URL で指定されるパス(ディレクトリ名)によって、処理されるクラス、メソッドが指定されます。
  リクエストは最初にコントローラーで処理されます。
- どのパスに対するリクエストをどのクラスのどのメソッドに処理させるのかは、`@RequestMapping`、`@GetMapping` などのアノテーションで指定されています。
- 今回の演習の対象である、`LessonController` クラスには `@RequestMapping("/lesson")` というアノテーションが指定されています。
- これは`/lesson` というパスに対して送られたリクエストを `LessonController` クラスにて処理する事を指定しています。
- また `index` メソッドには、`@GetMapping` とアノテーションがついていますが、これは HTTP リクエストメソッドが GET の場合は、`index` メソッドでリクエストを受け付ける事を指定しています。

```java
@Controller
@RequestMapping("/lesson")
public class LessonController {

  private final LessonService lessonService;

  @Autowired
  public LessonController(LessonService lessonService) {
    this.lessonService = lessonService;
  }

  @GetMapping
  public String index(Model model) {
    List<Lesson> list = lessonService.selectAll();

    model.addAttribute(attributeName: "lessonList", list);
    return "lesson/index";
  }
```

参考資料：HTTP リクエストメソッドについて

- `index` メソッドはサービスである `lessonService` クラスの `selectAll` メソッドを利用して、授業のリストを取得しています。
- model.addAttribute メソッドを使用して、画面表示したいデータとその属性名を model に指定しています。
  model.addAttribute メソッドの第一引数に属性名、第二引数に渡したいデータを指定しています。
- 戻り値として、使用したい View 名を指定する事で、テンプレートエンジン(Thymeleaf)の処理に移ります。
- テンプレートエンジンでは先ほど model に指定されたデータを取り出し、指定された属性名の部分にデータを展開します。

サービスクラスの動き

- サービスクラスはビジネスロジックを記載するクラスです、今回は `LessonService` クラスの `selectAll` メソッドが使用されています。

- LessonService クラスの selectAll メソッドから、Dao である ArchievementDao クラスの selectAll メソッドを呼び出して、Lesson クラスのオブジェクトのリスト取得し、コントローラーへ返却しています。

サービスクラス archievementDao.selectAll メソッド

```
 public List<Lesson> selectAll() {
    List<Lesson> list = dao.selectAll();
    return list;
  }
```

Dao クラスとは

Dao は、Data Access Object の略称でデータベース操作を担当するクラスです。

Dao クラス ArchievementDao

```
@Override
public List<Lesson> selectAll() {
String sql = "SELECT id, name, day, period, teacher FROM LESSON";
List<Map<String, Object>> resultList = jdbcTemplate.queryForList(sql);
List<Lesson> list = new ArrayList<Lesson>();
for (Map<String, Object> result : resultList) {
  Lesson lesson = new Lesson();
  lesson.setId((int) result.get("id"));
  lesson.setName((String) result.get("name"));
  lesson.setPeriod((int) result.get("period"));
  lesson.setTeacher(((String) result.get("teacher")));

  int day = (int) result.get("day");
  lesson.setDay(day);
  // 曜日番号 → 曜日名 変換
  lesson.setDayName(DayOfWeek.of(day).getDisplayName(TextStyle.SHORT, Loc
ale.JAPAN));

  list.add(lesson);
}

return list;
}
```

SQL 文を用いて、LESSON テーブルから jdbcTemplate.queryForList メソッドを利用してデータを取得しています。
Map 型で取得されますので、Lesson クラスの List 型オブジェクトに変換し、Service クラスへ値を返却しています。

テンプレートエンジンとは

今回の授業では、Spring Boot での使用が推奨されている、Thymeleaf(タイムリーフ)というテンプレートエンジンを使用しています。
テンプレートエンジンとは、データとテンプレートを合体させ文字列を作る仕組みの事です。
生成した HTML をブラウザに応答し、ブラウザが HTML を解釈し WEB ページの表示を行っています。 今回は `lesson/index` のテンプレートを利用しています。

テンプレート `lesson/index`(一部抜粋)

```
<table class="table">
  <thead>
    <tr>
      <th scope="col">id</th>
      <th scope="col">授業名</th>
      <th scope="col">曜日</th>
      <th scope="col">担当</th>
      <th scope="col"></th>
    </tr>
  </thead>
  <tbody>
    <tr th:each="lesson : ${lessonList}">
      <td th:text="${lesson.id}" scope="row"></td>
      <td th:text="${lesson.name}"></td>
      <td th:text="${lesson.dayName}+${lesson.period}">
      </td>
      <td th:text="${lesson.teacher}"></td>
      <td>
        <a class="btn btn-info rounded-circle p-0 pt-1" th:href="@{/lesso
n/detail/{id}(id=${lesson.id})}"
          style="width:2rem; height:2rem;">
          <i class="bi bi-pencil-fill"></i>
        </a>
        <a class="btn btn-outline-secondary rounded-circle p-0 pt-1"
          th:href="@{/lesson/delete?id={id}(id=${lesson.id})}" style="wid
th:2rem; height:2rem;">
          <i class="bi bi-trash-fill"></i>
        </a>
      </td>
    </tr>
  </tbody>
</table>
```

生成された HTML(一部抜粋)

```
<table class="table">
  <thead>
    <tr>
      <th scope="col">id</th>
      <th scope="col">授業名</th>
      <th scope="col">曜日</th>
      <th scope="col">担当</th>
      <th scope="col"></th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td scope="row">1</td>
      <td>lesson1</td>
      <td>月 1</td>
      <td>○○</td>
      <td>
        <a class="btn btn-info rounded-circle p-0 pt-1" href="/lesson/det
ail/1"
          style="width:2rem; height:2rem;">
          <i class="bi bi-pencil-fill"></i>
        </a>
        <a class="btn btn-outline-secondary rounded-circle p-0 pt-1"
          href="/lesson/delete?id=1" style="width:2rem; height:2rem;">
          <i class="bi bi-trash-fill"></i>
        </a>
      </td>
    </tr>
    <tr>
      <td scope="row">2</td>
      <td>lesson2</td>
      <td>月 2</td>
      <td>○○</td>
      <td>
        <a class="btn btn-info rounded-circle p-0 pt-1" href="/lesson/det
ail/2"
          style="width:2rem; height:2rem;">
          <i class="bi bi-pencil-fill"></i>
        </a>
        <a class="btn btn-outline-secondary rounded-circle p-0 pt-1"
          href="/lesson/delete?id=2" style="width:2rem; height:2rem;">
          <i class="bi bi-trash-fill"></i>
        </a>
      </td>
    </tr>
```

```
  </tbody>
</table>
```

- th:each="変数 : ${コレクション}"と記載する事で、${コレクション}の値を 1 つずつ取り出し変数に代入し、以降の処理の中で「変数.フィールド」の形で値へアクセス出来ます。java でいう拡張 for 文のようなイメージです。 今回だと、th:each="lesson : ${lessonList}"で lessonList の値を取り出し変数 lesson に代入し、表の表示に必要な値を 1 つずつ取り出し、表示しています。
- Thymeleaf(タイムリーフ) の構文については以下を参照してください。
  参考資料：Thymeleaf の紹介

データベースの利用について(H2 Database)

既存プログラムでは、データベースとして「H2 Database」が利用されています。
Java プラットフォーム上で動くオープンソースの RDBMS です。
非常に軽量なデータベースのため、Java プログラミングを試したりテストしたい場合によく利用されます。
データの保存先はインメモリかファイルです。
既存プログラムでは、保存先をメモリと指定しています。
そのため、起動し直す度にデータは削除されますので注意してください。

起動の度に、以下 2 つのファイル内の SQL 文が実行されます。
* ~\portal\src\main\resources\schema.sql *
~\portal\src\main\resources\data.sql
schema.sql には、CREATE TABLE 文などの DDL 文を、data.sql には、INSERT 文などの DML 文を記載します。
テーブル定義を変えたい、初期データに何か手を加えたい時は上記 2 ファイルを変更してください。

補足

各コードの配置場所について

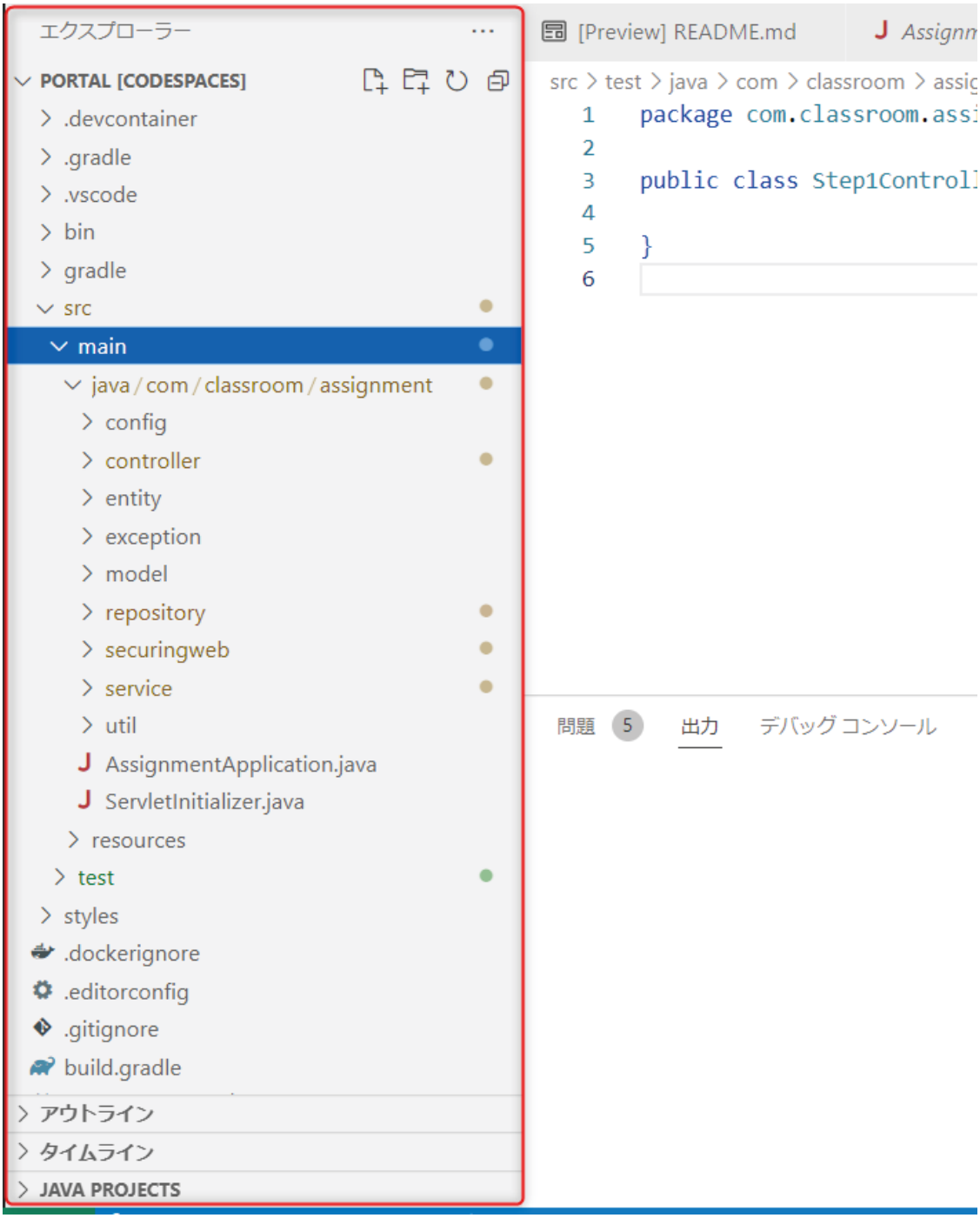
- Controller クラス
  /workspaces/portal/src/main/java/com/classroom/assignment/controller

- Service クラス
  /workspaces/portal/src/main/java/com/classroom/assignment/service

- Dao クラス
  `/workspaces/portal/src/main/java/com/classroom/assignment/repository`

- ビュー(テンプレート)
  `/workspaces/portal/src/main/resources/templates`

左側ツリーから参照してください。


エクスプローラー
PORTAL [CODESPACES]
.devcontainer
.gradle
.vscode
bin
gradle
src
main
java / com / classroom / assignment
config
controller
entity
exception
model
repository
securingweb
service
util
AssignmentApplication.java
ServletInitializer.java
resources
test
styles
.dockerignore
.editorconfig
.gitignore
build.gradle
アウトライン
タイムライン
JAVA PROJECTS
[Preview] README.md
src > test > java > com > classroom > assig
1 package com.classroom.assi
2
3 public class Step1Controll
4
5 }
6
問題 5
出力
デバッグ コンソール

【Level1-1】ユーザー名の表示（静的）

テーマ

1. Controller および Model クラスの取り扱いについて

演習課題の準備（開発環境について）

- この演習では、Codespaces と呼ばれるサービスを使って開発を行います。
- ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。
- レベル 0 で実施した手順を参照して、演習課題の準備をしましょう。

演習内容

Portal ページに表示されている、「こんにちは、ooさん」の表示を変更しましょう。
「ooさん」の部分を自分の名前に変更して下さい。

ヒント・解説

Controller クラス

Portal ページを表示するために、`PortalController` クラスが利用されます。
このクラスには`@Controller` と`@RequestMapping("/")`というアノテーションがついています。
`PortalController` クラスには `index` メソッドのみ存在します。
このメソッドには、`@GetMapping()`アノテーションがついています。

アノテーション	役割
@Controller	コントローラーである事を示すアノテーションです。
@RequestMapping("/")	パスのない URL に対してリクエストがあった際に、このクラスでリクエストを処理する事を示します。
@GetMapping()	GET リクエストを処理するメソッドである事を示します。

Model クラス

コントローラーからビュー側へデータを渡す時に使用されるクラスです。
Model クラスのオブジェクトに `model.addAttribute` メソッドを使用して値を設定

しています。
model.addAttribute メソッドの第一引数に属性名、第二引数に渡したい値を指定しています。

```
model.addAttribute("name", "○○"); // ○○を書き換えて名前を表示
```

今回は、属性名はそのままに、名前として渡している○○の値を自分の名前に変更すれば OK です。

ビューの動きについて

index メソッドは戻り値に"index"が指定されているため、index.html ファイルをテンプレートとして使用し、HTML を生成します。
model オブジェクトで渡した、属性名 name のデータについては以下の部分で使用されています。

```
<span class="text-black fw-bold" th:text="${name}">名前</span>さん
```

今回使用している、Thymeleaf というテンプレートエンジンでは、テンプレートファイルに、
th:text="${属性名}"の形式で記載すると、コントローラーで設定した、値で、タグ内の値が置き換えられます。

課題の提出

GitHub への提出

- 以下の操作を行い課題を提出しましょう。

1. Commit(コミット)

- Commit は、追加・変更したファイルを Git に登録するための操作です。

2. Push(プッシュ)

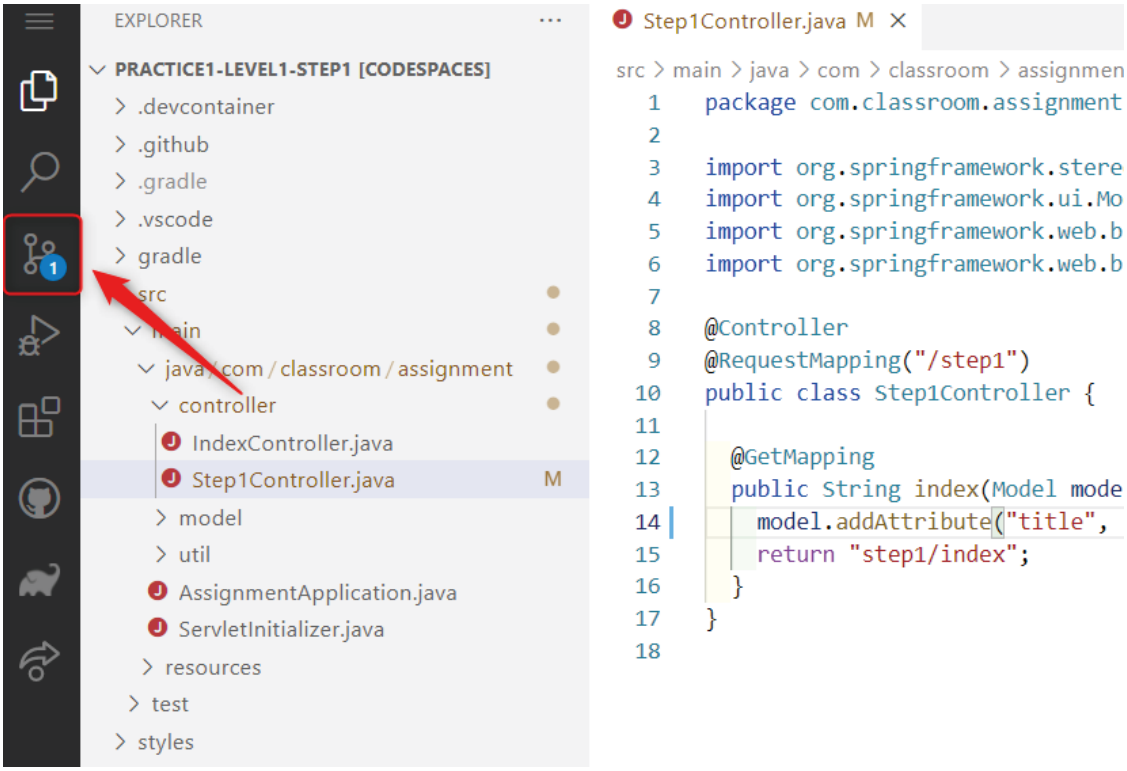
- push は、コミットした内容をローカルレポジトリからリモートリポジトリに送信する操作です。

3. Pull Request(プルリクエスト)

- Pull Request は、開発者のローカルレポジトリでの変更を他の開発者に通知する機能です。

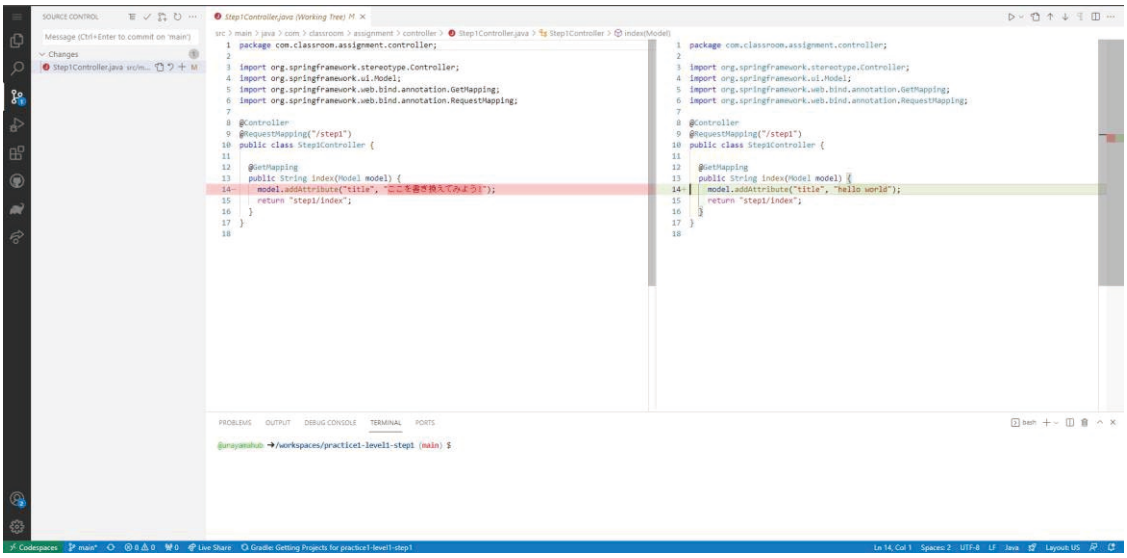
手順

1. 木の枝のようなマークを押します。



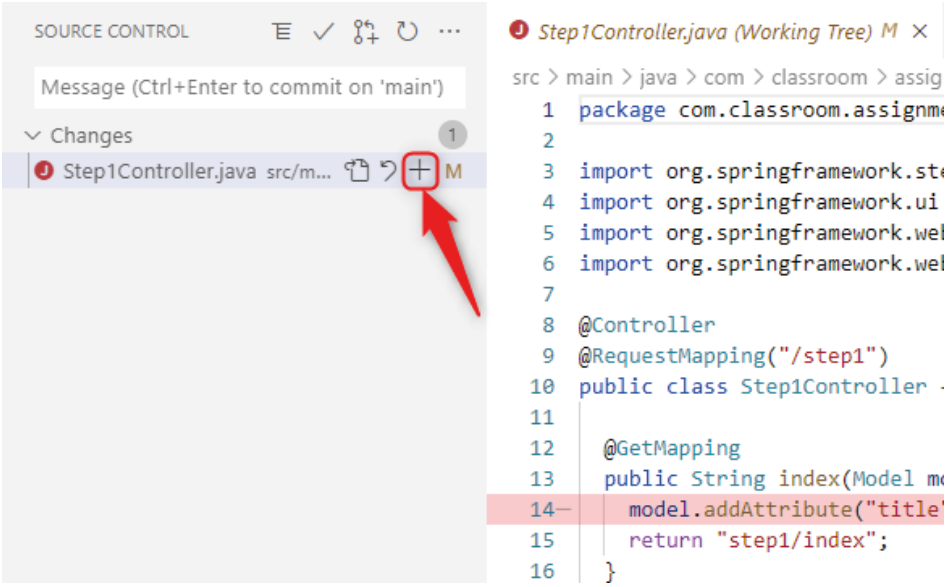
2. これまでに行ったファイルへの変更ないようが表示されています。

- 変更内容に問題がないか今一度確認しましょう。

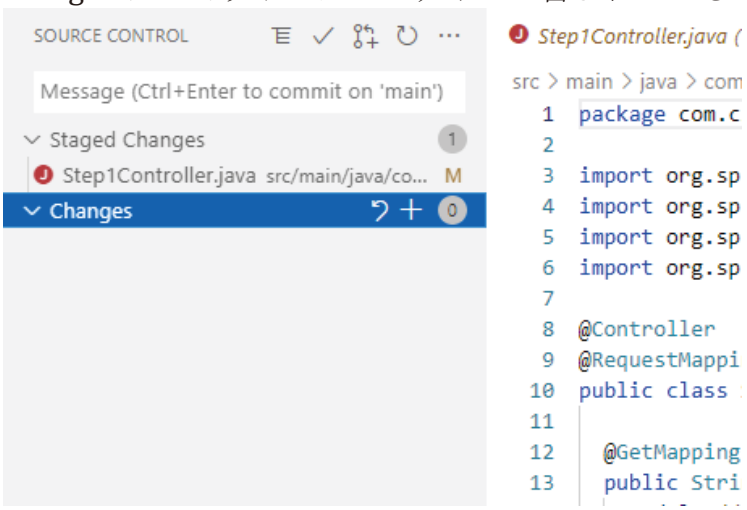


3. コミットするファイルを選択しましょう。

- コミットしたいファイルの横の＋ボタンを押しましょう。



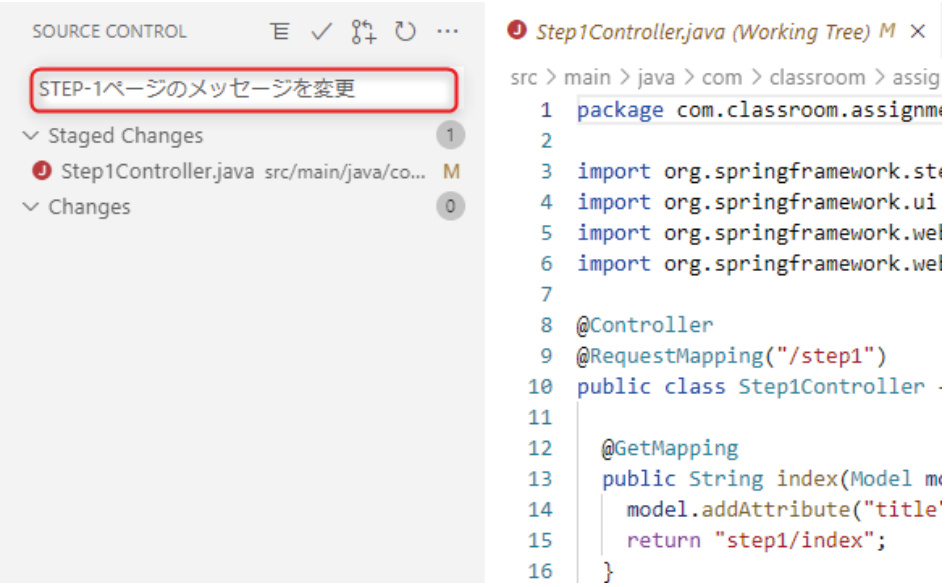
4. Staged Changes にコミットしたいファイルが含まれていることを確認しましょう。



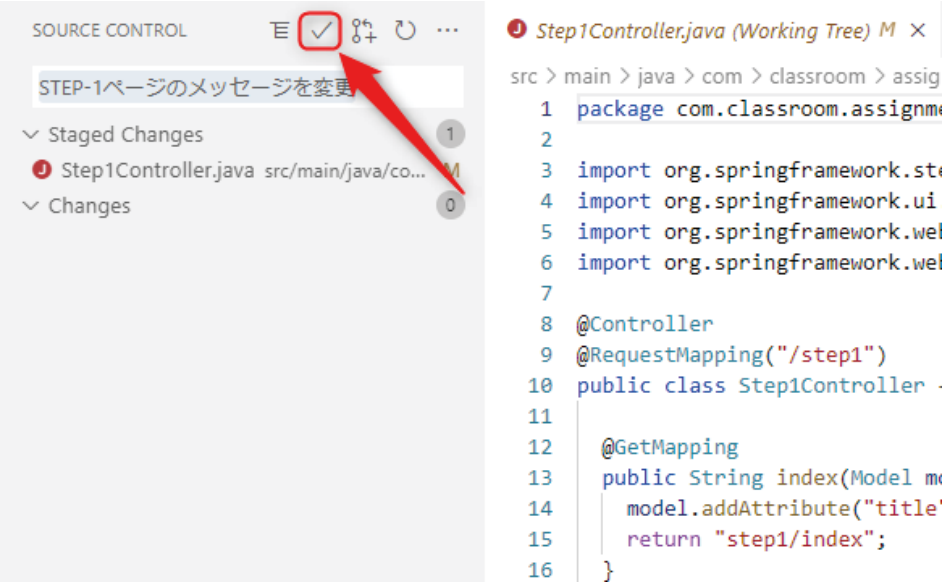
5. Message 欄に Commit Message(コミットメッセージ)を入力しましょう。

- Commit Message には、今回のコミットの追加・変更内容について記載します。

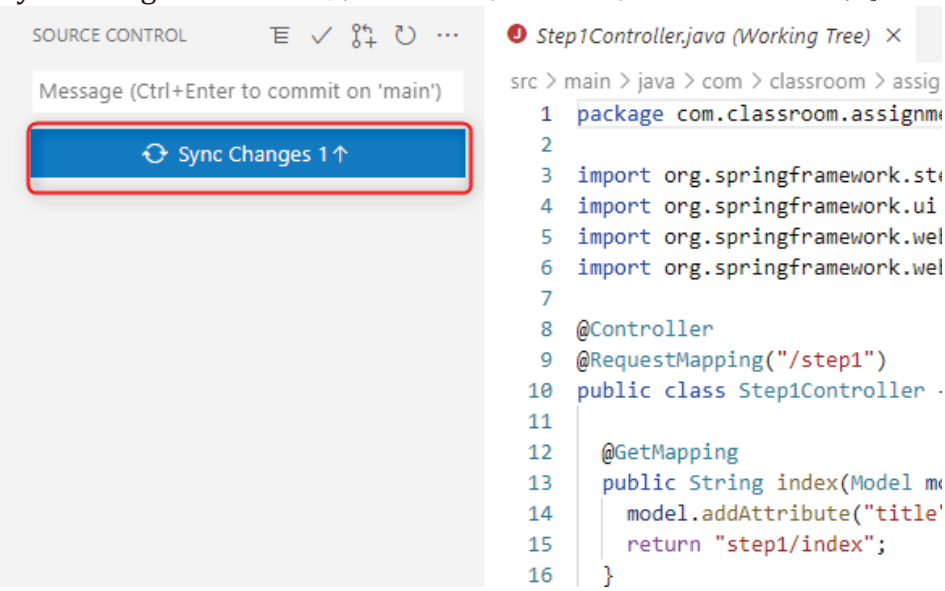
- 今回は「「ooさん」の部分を自分の名前に変更」としておきましょう。



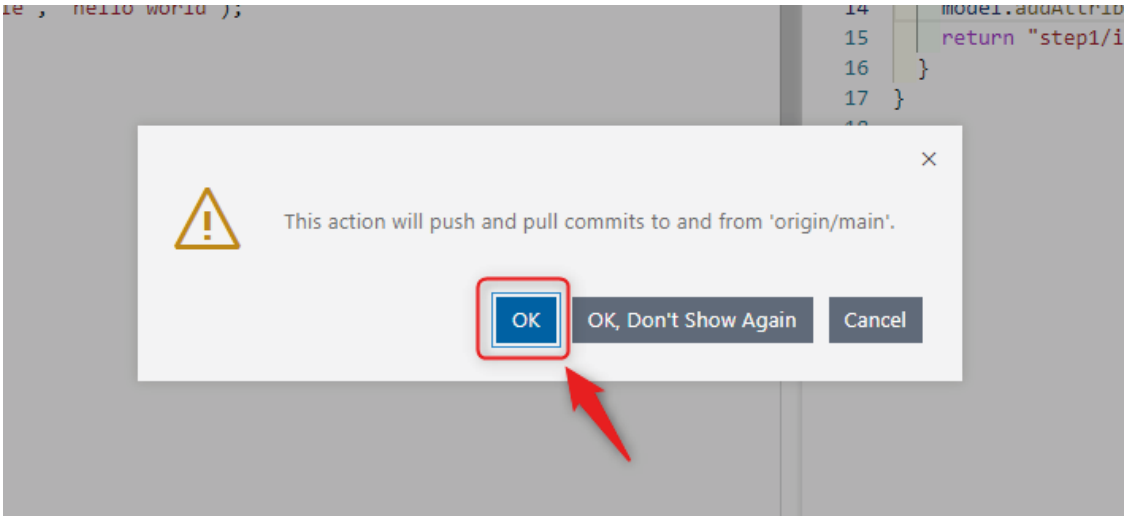
6. チェックボタンを押してコミットしましょう。



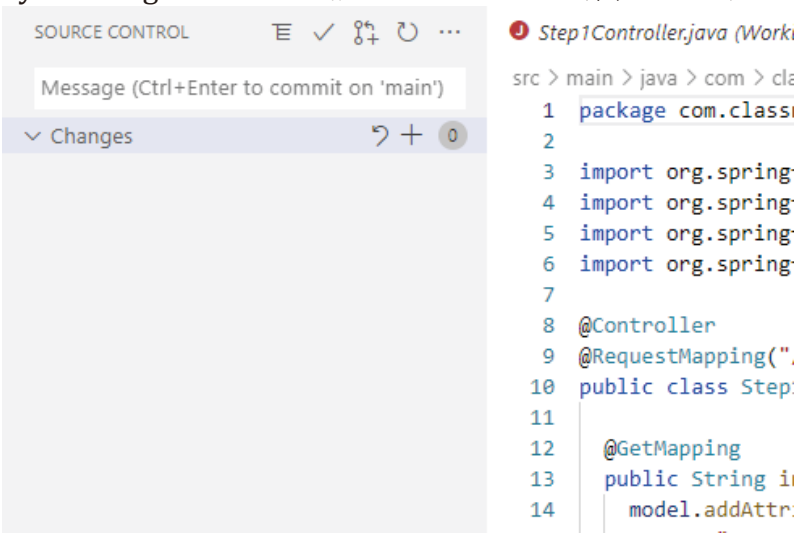
7. Sync Changes ボタンを押してコミットをプッシュしましょう。



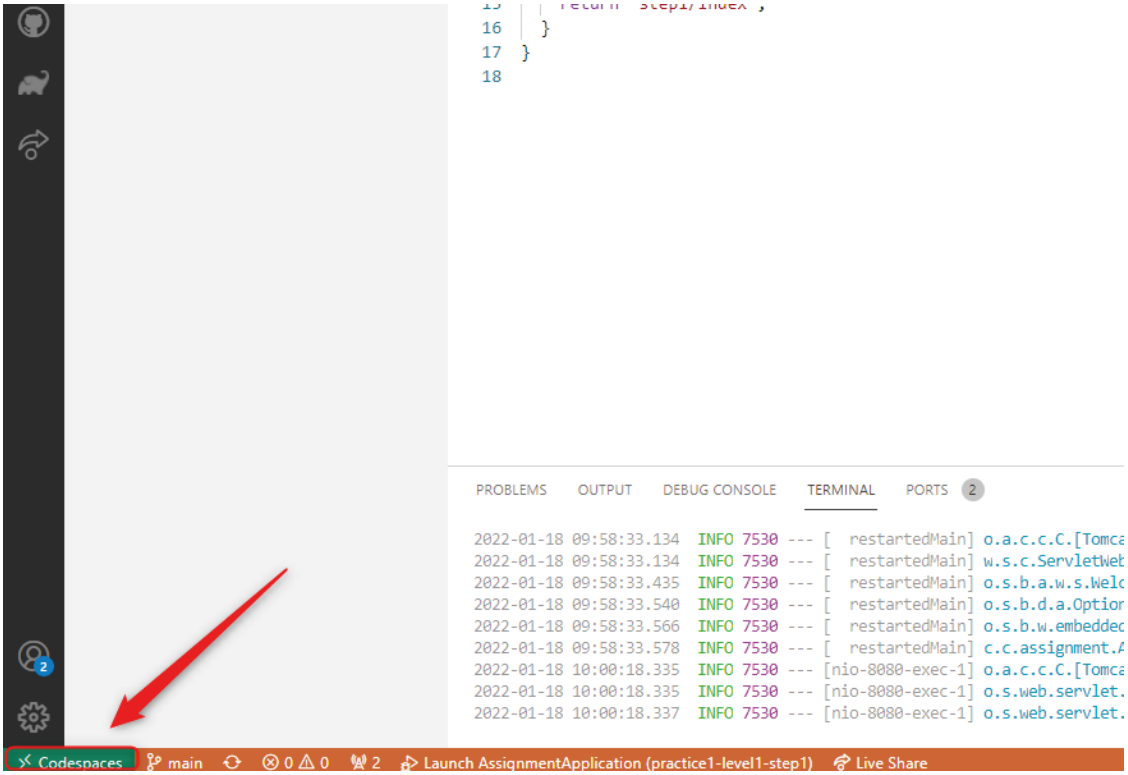
8. ダイアログが出ますので「OK」を押しましょう。



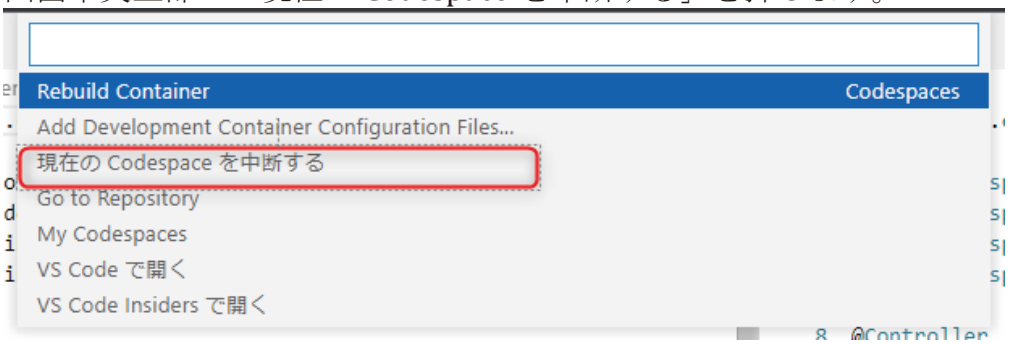
9. Sync Changes ボタンが消えていることを確認してください。



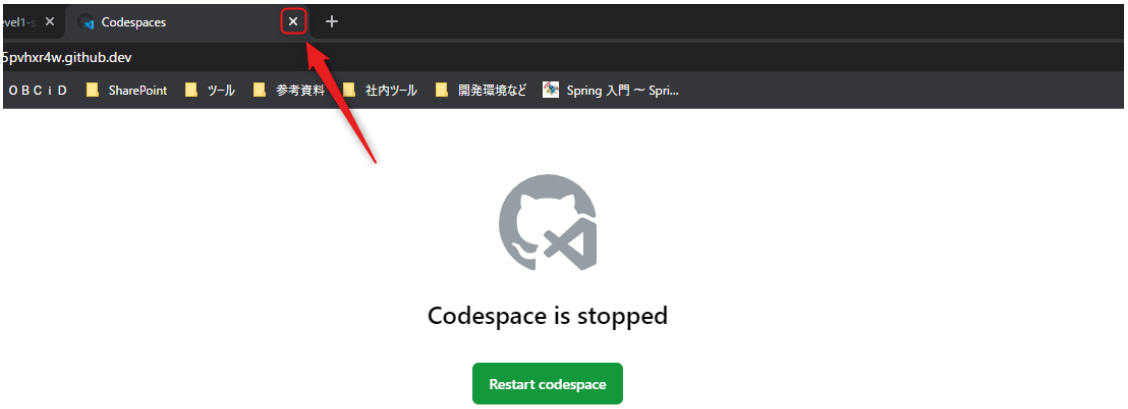
10. 画面左下の緑のボタンを押しましょう。



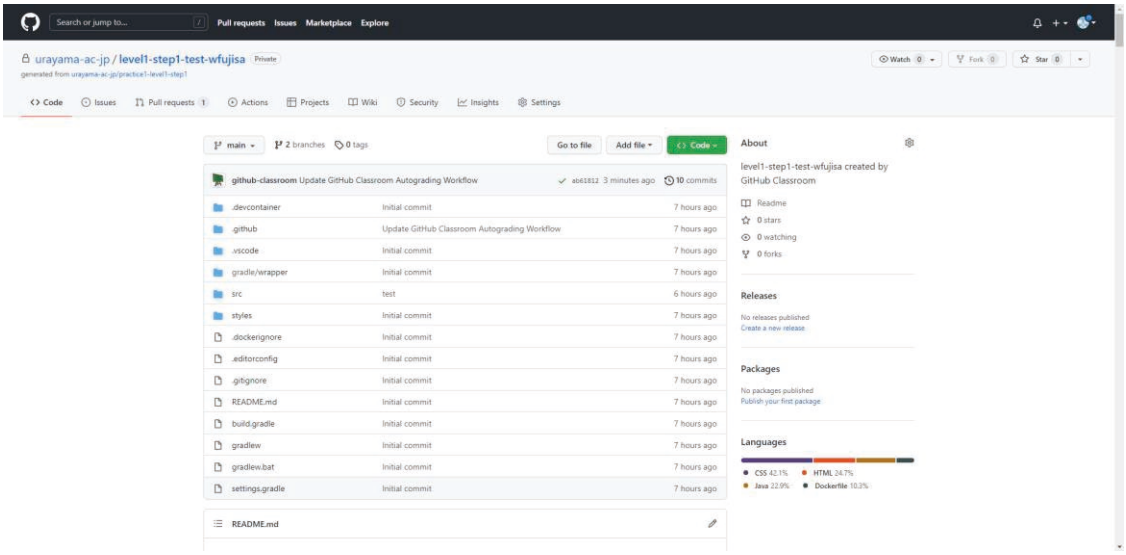
11. 画面中央上部の「現在の Codespace を中断する」を押します。



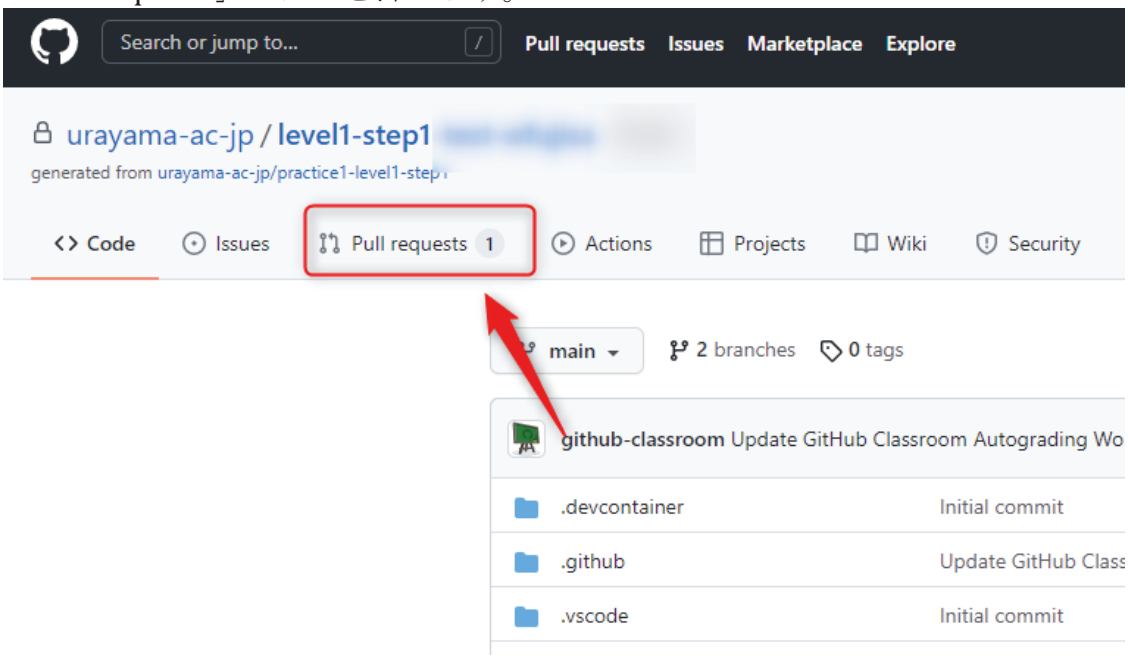
12. タブを×ボタンで閉じましょう。



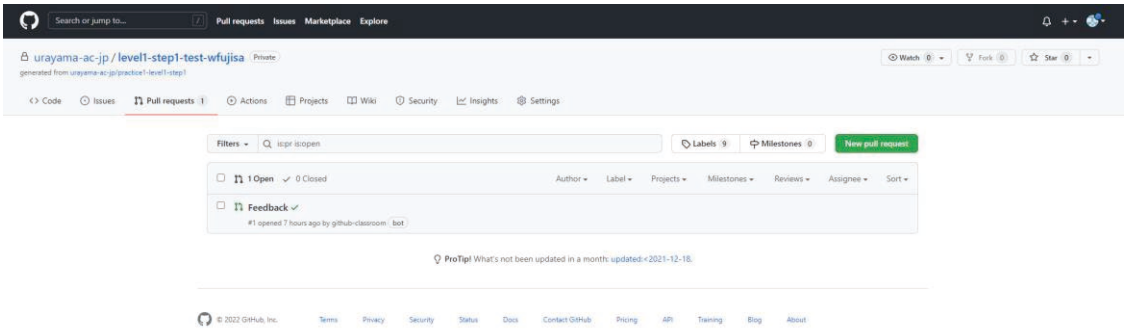
13. 画面コピーの画面に戻ります。



14. 「Pull requests」ボタンを押します。



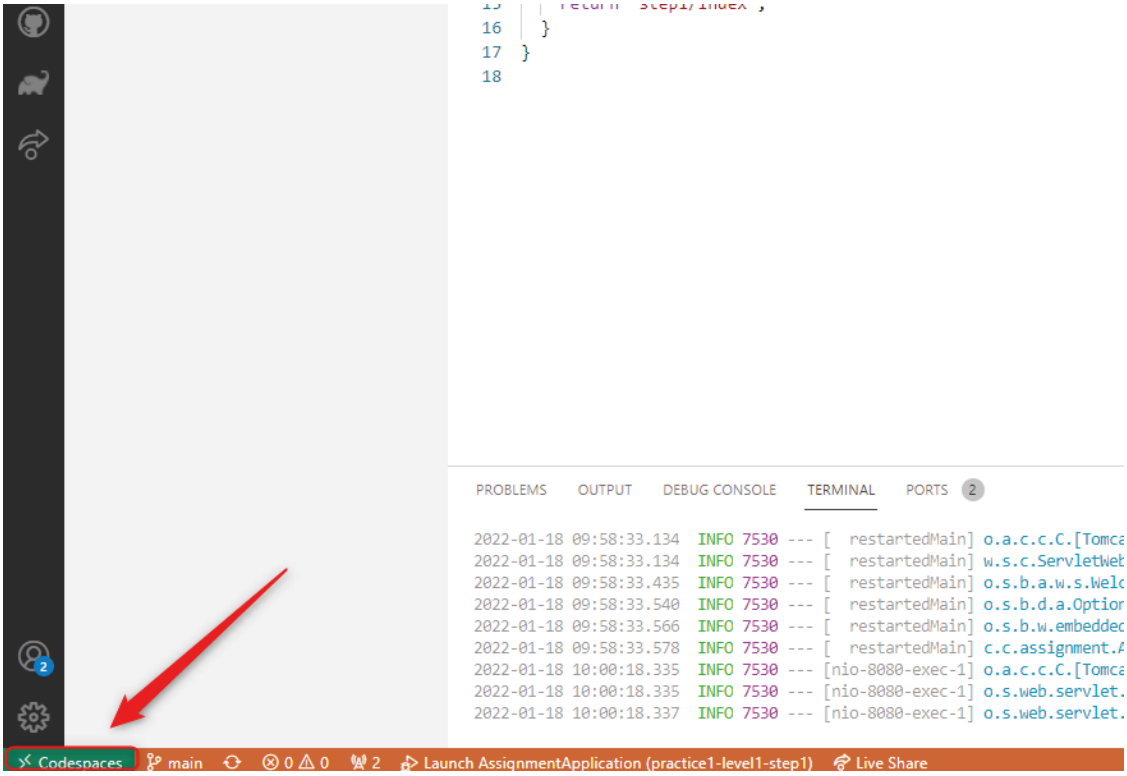
15. 講師からのフィードバックがある場合は、この画面に表示されます。



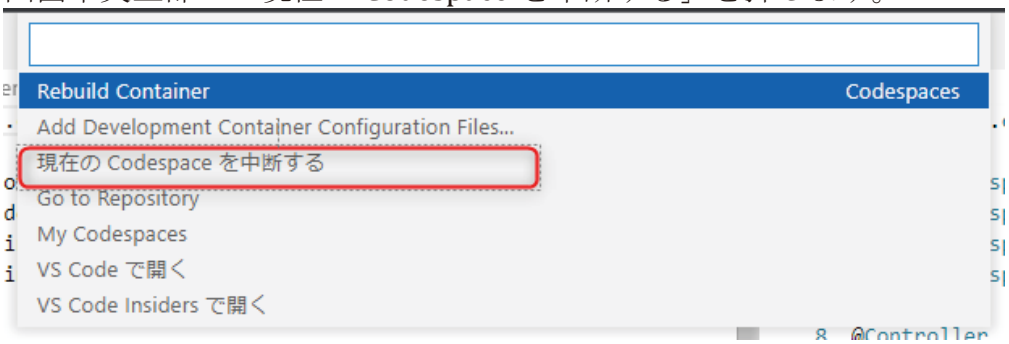
課題を提出したら、Codespaces を終了しましょう

- Codespaces はブラウザタブの×ボタンで閉じないようにしてください、以下の手順で終了してください。

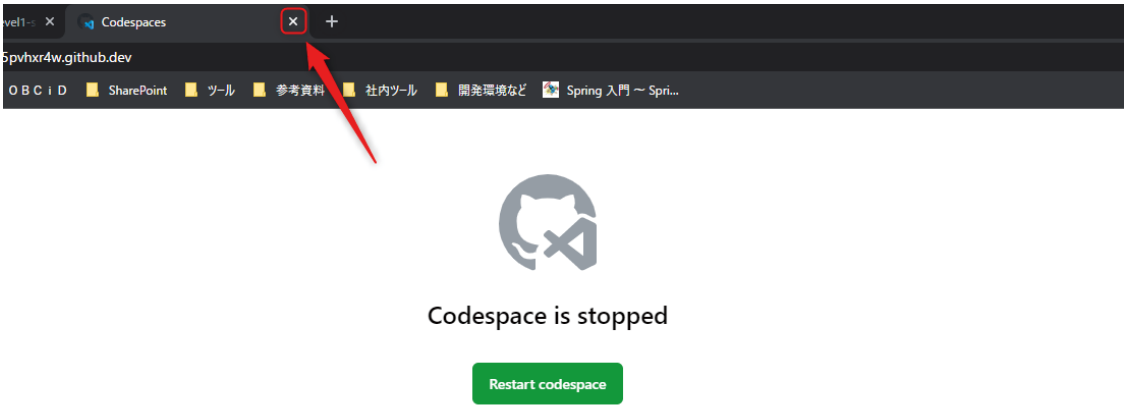
1. 画面左下の緑のボタンを押しましょう。



2. 画面中央上部の「現在の Codespace を中断する」を押します。



3. タブを×ボタンで閉じましょう。



【Level1-2】今日の天気を表示（API 利用）：都市を変更

テーマ

1. Service クラスの取り扱いについて
2. 天気予報 API の利用について

演習課題の準備（開発環境について）

- この演習では、Codespaces と呼ばれるサービスを使って開発を行います。
- ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。
- レベル 0 で実施した手順を参照して、演習課題の準備をしましょう。

演習内容

Service の一部を書き換えて自分の住んでいる都市の天気を表示しましょう。

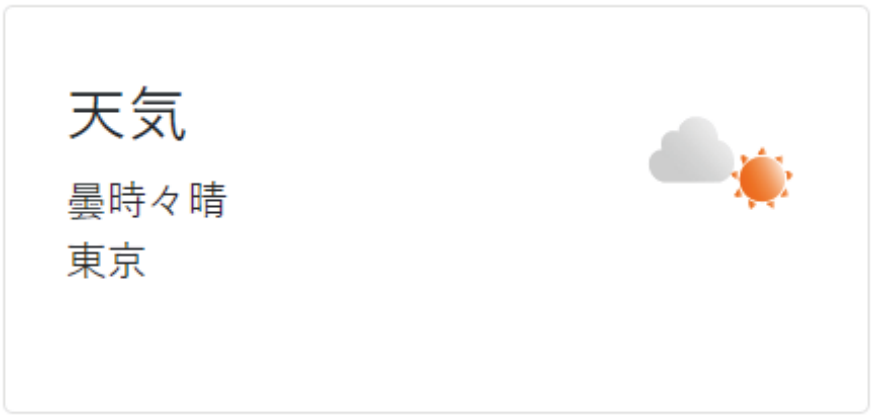
ヒント・解説

Service クラス

PortalController の index メソッド内の天気情報を取得している箇所を参照します。

```
/**
* level 1-2 天気取得都市の変更
*/
// 天気
Weather weather = weatherService.findWeather();
model.addAttribute("weather", weather);
```

天気の情報を得るために、`WeatherService` の `findWeather` メソッドが利用されます。
取得した、weather オブジェクトをビューに渡すことで、今日の天気、都市名、天気の画像を表示しています。



image

WeatherService の findWeather メソッドを確認しましょう。

```
/**
* level 1-2 天気取得都市の変更
*
* https://weather.tsukumijima.net/ を参考に自分の住んでいる都市の city code をに変更する
*/
final String cityCode = "130010"; // 東京の CityCode
final String endpoint = "https://weather.tsukumijima.net/api/forecast";

final String url = endpoint + "?city=" + cityCode;

ResponseEntity<String> response = rest.getForEntity(url, String.class);
String json = response.getBody();

ObjectMapper objectMapper = new ObjectMapper();

JsonNode jsonNode = objectMapper.readTree(json);
Weather weather = new Weather();
weather.setCity(jsonNode.get("location").get("city").textValue());
weather.setForecast(jsonNode.get("forecasts").elements().next().get("telop").textValue());
weather
    .setImage(
        jsonNode.get("forecasts").elements().next().get("image").get("url").textValue());
return weather;
```

※例外処理は省いて、コードを抽出しました。

今回は天気予報 API(livedoor 天気互換)という API を使用しています。 パラメータとして地域別に定義された ID 番号を指定する事で、指定された地域の天気情報を JSON 形式で取得できます。
参考資料：JSON

例では、変数 cityCode(地域別に定義された ID 番号)に東京を示すコードである 130010 を指定し、 https://weather.tsukumijima.net/api/forecast?city=130010 という URL を作成し URL に対して GET リクエストを行い、東京の天気の JSON データを取得しています。
取得した JSON から必要な値を取り出し、weather オブジェクトに格納し、コントローラーへ返しています。
weather オブジェクトの city フィールドには、指定した ID で取得した地域名を、
forecast フィールドには、天気を、image フィールドには、お天気の画像の URL を格納しています。

では、現在「東京」に指定されている、地点を自分が住んでいる地域に変更しましょう。 地域別に定義された ID は以下のページから探して下さい。
1 次細分区定義表

ビューの動きについて

前回と同じように、index メソッドは戻り値に"index"が指定されているため、index.html ファイルをテンプレートとして使用し、HTML を生成します。 model オブジェクトで渡した、属性名 weather のデータについては以下の部分で使用されています。

```
<div class="row">
    <div class="col">
        <div class="card-title fs-4 fw-semibold">天気</div>
        <div class="card-subtitle text-disabled" th:text="${weather.forec
ast}">天気</div>
        <div class="card-text" th:text="${weather.city}">都市</div>
    </div>
    <div class="col text-end text-primary fs-4 fw-semibold">
        <img th:src="${weather.image}" alt="weather" width="70" height="7
0" />
    </div>
</div>
```

前回 th:text="${属性名}"の形式で記載すると、コントローラーで設定した、値でタグ内の値が置き換えられるとお伝えしました。

属性に指定した値が、オブジェクトである場合は、`th:text="${属性名.フィールド名}"`の形式で指定すると、指定されたフィールド名の getter メソッドを呼び出し、フィールド値を取り出します。
取り出した値で、タグ内の値が置き換えられます。

課題の提出

- 課題の提出は課題提出の操作のコミット・プッシュ・プルリクエストを実施しましょう。

【Level2-1】おみくじ（アルゴリズムなど作成）

テーマ

1. Controller クラスと Service クラスの連携
2. 乱数の使い方

演習課題の準備（開発環境について）

- この演習では、Codespaces と呼ばれるサービスを使って開発を行います。
- ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。
- レベル 0 で実施した手順を参照して、演習課題の準備をしましょう。

演習内容

Service のメソッドで乱数を用いたおみくじアルゴリズムを実装

演習内容詳細

`FortuneService` の `omikuji` メソッドを完成させて下さい。 0 から 4 の乱数を発生させ、0: 大吉、1: 中吉、2: 小吉、3: 末吉、4: 凶 が表示されるようなアルゴリズムを作成してください。

ヒント・解説 乱数を取得するために、`java.util.Random` クラスを使用します。このクラスは `java.util` パッケージに含まれるクラスの一つです。 このクラスを利用するためには、import 文でクラスを読み込む必要があります。 既に import 文が記載されている箇所に、以下の import 文を追記してください。 `import java.util.Random;` 今回は 0 から 4 の乱数を得たい訳ですから、`nextInt(int)`メソッドを利用します。 `nextInt(int)`メソッドは 0 から引数として与えられた値未満の値の範囲でランダムな値を返します。 `nextInt(2)`とした場合は、「0,1」いずれかの値が返されます。 詳しくは次のページを参考にしてください。
https://docs.oracle.com/javase/jp/11/docs/api/java.base/java/util/Random.html#nextInt(int)

こうして、得られた数値に対応する文字列を返却するメソッドを作成してください。

課題の提出

- 課題の提出は課題提出の操作のコミット・プッシュ・プルリクエストを実施しましょう。

【Level2-2】挨拶の表示（時刻を元に表示内容を変更）

テーマ

1. Controller クラスと Service クラスの連携
2. LocalDate の使い方

演習課題の準備（開発環境について）

- この演習では、Codespaces と呼ばれるサービスを使って開発を行います。
- ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。
- レベル 0 で実施した手順を参照して、演習課題の準備をしましょう。

演習内容

時間帯に分けて、表示するあいさつ文を変更するアルゴリズムを作成してください。

演習内容詳細

`GreetingService` の `greeting` メソッドを完成させて下さい。 5~11 時(5:00~11:59) は「Good Morning」、 12~17 時(12:00~17:59) は「Good Afternoon」、 18~4 時(18:00~4:59) は「Good Evening」としてください。

ヒント解説

`greeting` メソッドは引数として、`LocalDateTime` クラスのオブジェクトを受け入れます。 `GreetingService` の呼び出し元である `PortalController` の `index` メソッドの最初の方で、オブジェクトを作成しています。

```
LocalDateTime now = LocalDateTime.now(ZoneId.of("Asia/Tokyo"));
```

`LocalDateTime.now` をメソッドを利用し、実行時点の時間情報を持った、`LocalDateTime` クラスのオブジェクトを作成します。 これは、Portal ページで日付と曜日を表示するためにも使用されています。 `GreetingService` の `greeting` メソッドでは、何時かさえ分かればいいので、時(hour)のみを取得し、int 型の変数 `hour` に格納しています。 int 型の変数 `hour` の値を参照し、適切なあいさつ文を、変数 `greeting` に格納戻り値としましょう。

課題の提出

- 課題の提出は課題提出の操作のコミット・プッシュ・プルリクエストを実施しましょう。

【Level2-3】Dashboard の本日の授業表示

テーマ

1. DAO クラスの取り扱い
2. sql の作成

演習課題の準備（開発環境について）

- この演習では、Codespaces と呼ばれるサービスを使って開発を行います。
- ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。
- レベル 0 で実施した手順を参照して、演習課題の準備をしましょう。

演習内容

DAO を修正し、本日の授業が表示されるようにする

演習内容詳細

DAO の SQL 文を修正し、曜日毎の授業が表示されるようにしましょう。

ヒント・解説

DAO クラスとは

データアクセスオブジェクトの略で「ダオ」と呼ばれます、アプリケーション内でのデータアクセスの機能を担います。
データベースアクセス専門のクラスを作って、データベースに関する事はすべてそのクラスに任せようという考えの元作られたデザインパターン(汎用的な設計パターン)です。
DBMS（データベース管理システム）が変わったりした場合(例えば Oracle から MySQL)DAO のみを変更する事で対処する事が出来るので、変更に強いデザインパターンといわれています。
今回は `LessonDao` クラスという DAO クラスを使用しています、このクラスは `lesson` テーブルに対するデータ操作を担当するクラスです。
なお、授業の情報を管理する `lesson` テーブルは以下のようなテーブル構造になっています。 |列名|データ型|概要| |---|---|---| |id|INT|このテーブルの主キーとなる ID| |name|VARCHAR(100)|授業名| |day|INT|曜日を数値で格納します。(月:1~日:7)| |period|INT|時限(1 限目、2 限目)|

`LessonDao` クラスの `selectByDay` メソッドは現在 `lesson` テーブルに格納されているレコードをすべて抽出する作りとなっています。
使用している SQL 文に手を加えて、`selectByDay` メソッドの引数で与えられた変数 `day` を条件とし、曜日で絞り込まれたレコードを返すメソッドに修正してください。
既に `id` での絞り込みを行っている、`findById` メソッドも参考にしてみましょう。

課題の提出

- 課題の提出は課題提出の操作のコミット・プッシュ・プルリクエストを実施しましょう。

【Level3-1】

テーマ

DDL, DML が作成できるようになろう

演習課題の準備（開発環境について）

- この演習では、Codespaces と呼ばれるサービスを使って開発を行います。
- ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。
- レベル 0 で実施した手順を参照して、演習課題の準備をしましょう。

演習内容

lesson を参考に DB に実績テーブルを追加しましょう。 初期データを投入する INSERT 文も作成して下さい。

演習内容詳細

実績テーブルは以下の仕様で作成してください。

テーブル名:ARCHIEVEMENT |列名|データ型|制約・属性|備考| |---|---|---|---| |id|INT|NOT NULL AUTO_INCREMENT|このテーブルの主キーとなる項目| |name|VARCHAR(100)|NOT NULL|実績名| |memo|VARCHAR(100)|NOT NULL|メモ|

id 列は主キーとするため PRIMARY KEY 制約も付与してください。

- 初期データとして以下のレコードを投入する INSERT 文を作成してください。

name	memo
IT パスポート	
基本情報	
portal 作成	http://github.com

ヒント解説

起動の度に、以下 2 つのファイル内の SQL 文が実行されます。 * `~\portal\src\main\resources\schema.sql` * `~\portal\src\main\resources\data.sql`

schema.sql には、CREATE TABLE 文などの DDL[^1]を、data.sql には、INSERT 文などの DML 文[^2]を記載します。 テーブル定義を変えたい、初期データに何か手を加えたい時は上記 2 ファイルを変更してください。 以下の構文を参考に、それぞれのファイルに CREATE TABLE 文と、INSERT 文を追記してください。
[^1]: データベースにおいて、データ構造の定義に用いられる SQL。
[^2]: データベースにおいて、データの検索・新規登録・更新・削除を行うための SQL。

CREATE TABLE 文の構文

CREATE TABLE 文はデータベースにテーブルを作成する SQL 文です。
今回の演習では CREATE TABLE 文を作成するため、参考に構文を記載します。

```
CREATE TABLE テーブル名
(
   列名 1 データ型 制約・属性 ,
   列名 2 データ型 制約・属性 ,
   PRIMARY KEY(主キー制約を付与する列名)
);
```

INSERT 文の構文

INSERT 文はテーブルにデータを追加する SQL 文です。

```
INSERT INTO テーブル名 (列名 1, 列名 2) VALUES (値 1, 値 2);
```

AUTO_INCREMENT 属性が設定されている列への値の指定は不要です、列名自体記載しないか、NULL 値を入れるようにしてください。
NOT NULL の制約が設定されている列へは、NULL 値が格納出来ないため、空文字 ''を格納してください。

データ型について

テーブルのデータ型について解説します。 |データ型|説明| |---|---| |VARCHAR(n)| 文字列を格納するデータ型です、カッコ内には格納する文字列の最大桁数を記載してください。| |INT|-2,147,483,647 から 2,147,483,647 の範囲の整数値を格納するデータ型です。| |BOOLEAN|true,false の真偽値を格納するデータ型です。|

制約・属性について

テーブルの制約・属性について解説します。 |制約・属性|説明| |---|---| |NOT NULL|NULL 値が入れられなくなる制約です、必須項目にしたい項目に指定します。| |AUTO_INCREMENT|自動で連番を格納する属性です、属性が設定された列に

格納されている値+1 の値を自動的に格納します。| |PRIMARY KEY|主キー制約です、主キー列に指定します。主キー列とは、テーブル内の行を一意に識別する事の出来る列の事です。この制約を付与する事で重複する値を格納出来なくなります。|

課題の提出

- 課題の提出は課題提出の操作のコミット・プッシュ・プルリクエストを実施しましょう。

【Level3-2】

テーマ

既存プログラムを参考に、CRUD 機能が作成できること ページ遷移とバリデーションの仕組みが理解できること template への記載ができるようになること

演習課題の準備（開発環境について）

- この演習では、Codespaces と呼ばれるサービスを使って開発を行います。
- ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。
- レベル 0 で実施した手順を参照して、演習課題の準備をしましょう。

演習内容

Lesson(授業管理機能)を参考に Archievement(実績管理機能)の実績一覧画面を作成しましょう。

演習内容詳細

実績を保持する、ARCHIEVEMENT テーブルの内容を表示する、以下のような画面を作成しましょう。

以下の順序で実装を行ってください。

1. データベース情報を保持する、Entity クラスである `Archievement` クラスを作成してください。
2. DAO クラスである、`ArchievementDao` クラスを作成してください。
3. サービスとして、`ArchievementService` クラスを作成してください。

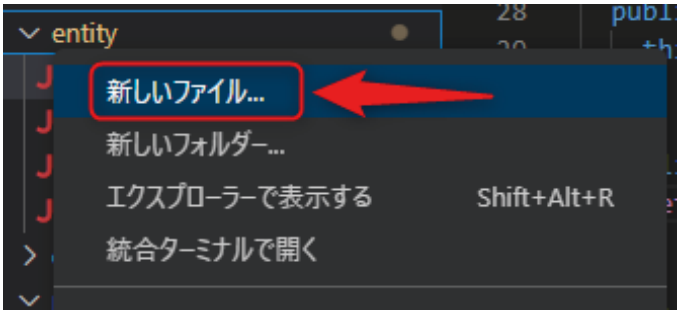
4. ビューのテンプレートファイルとして`~\resources\templates`配下に新たに、`archievement`ディレクトリを作成し、`index.html`ファイルを作成してください。 **なお、今回はテンプレートファイルを用意しておりますので、そちらを使ってください。**
5. コントローラーとして、`ArchievementController`クラスを作成してください。

それぞれに対応する、授業管理機能(Lesson)のファイルを参考に作成してください。

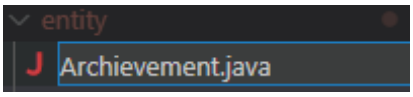
Archievement	Lesson
ArchievementController	LessonController
ArchievementService	LessonService
Archievement	Lesson
ArchievementDao	LessonDao
archievement/index.html	lesson/index.html

ファイルの新規作成方法

1. ファイルを新規作成したい、ディレクトリの上で右クリック、新しいファイルを押します。



2. ファイル名を拡張子を含めて入力し、Enter キーを押します。 ファイル名とクラス名は同じにしておきましょう。



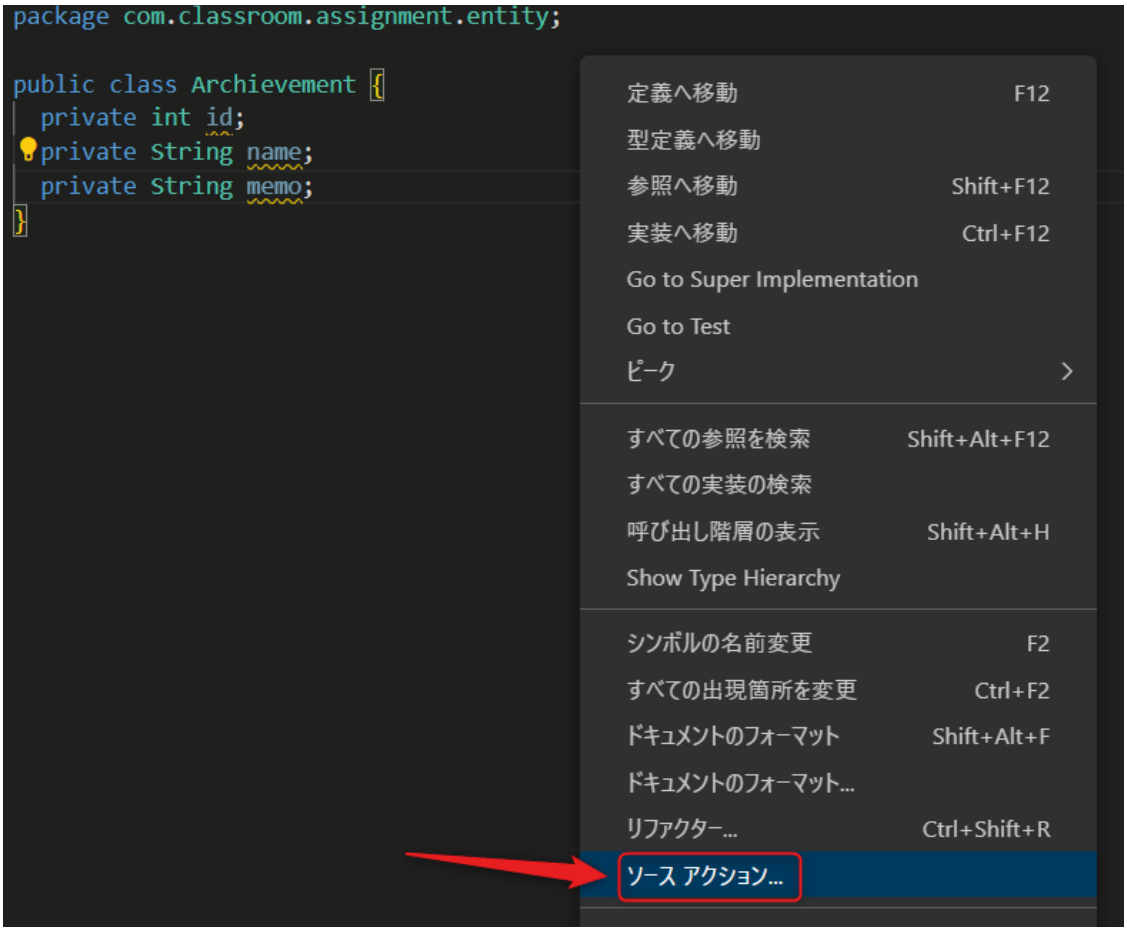
1.Entity クラスの作成

まずは Entity クラスを作成しましょう、`Archievement.java`ファイルを作成してください。 Entity クラスは 1 つのオブジェクトがデータベーステーブルの一行に対応するように作成します。 今回作成するテーブルには、3 つの列が存在します。 それぞれの列に対応する、メンバ変数を作成しましょう。 なおデータベース

のデータ型と、JAVA の変数のデータ型は以下の対応表に沿って作成してください。 |データベース|JAVA| |---|---| |VARCHAR|String| |int|int| |boolean|boolean|

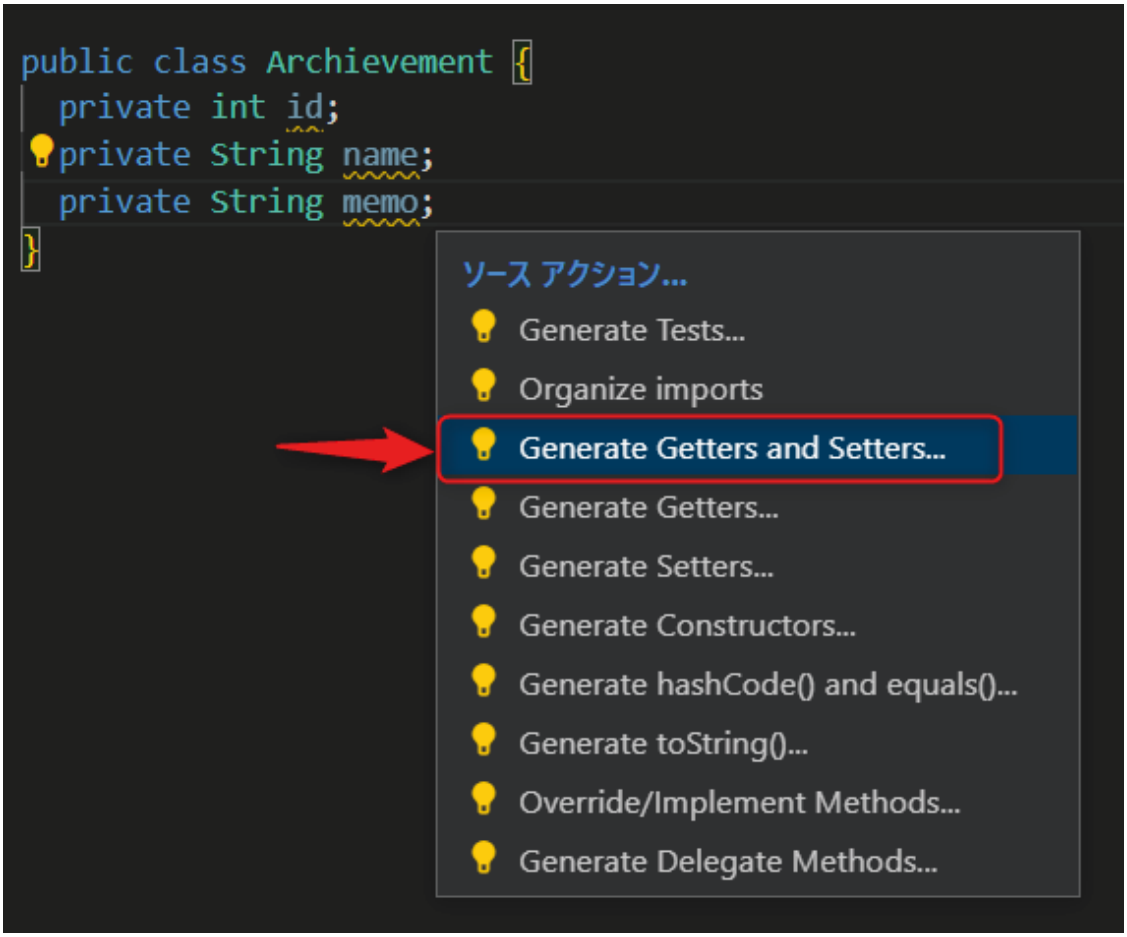
メンバ変数名と、データベースのカラム名は今回は同じにしてください。
Constructor(コンストラクター)と、Getter/Setter(ゲッター/セッター)を作成します。 参考：constructor について 参考：Getter について 参考：Setter について

Getter/Setter(ゲッター/セッター)の作成 1. 右クリックし、ソースアクションを選択してください。



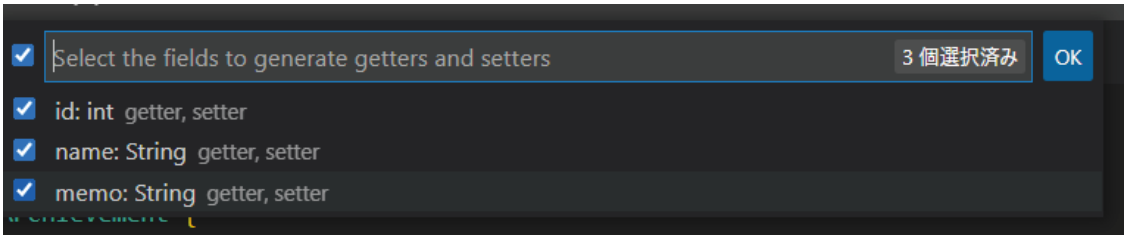
image

1. `Generate Getters and Setters...`を選択してください。



image

1. 全てにチェックを入れて OK を押してください。



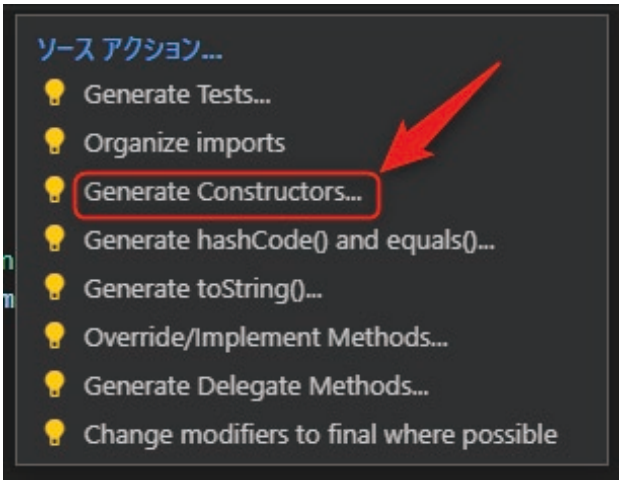
2. Getter/Setter(ゲッター/セッター)が生成された事を確認してください。

```
public int getId() {
  return id;
}

public void setId(int id) {
  this.id = id;
}

public String getName() {
  return name;
}

public void setName(String name) {
  this.name = name;
}

public String getMemo() {
  return memo;
}

public void setMemo(String memo) {
  this.memo = memo;
}
```

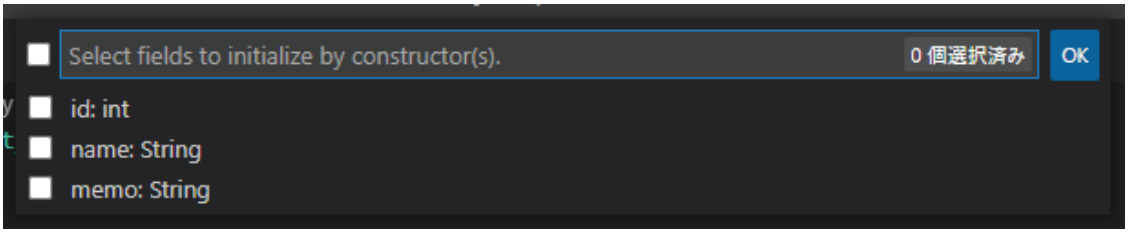
Constructor(コンストラクター)の作成

1. 再度、右クリックし、ソースアクションを選択してください。

2. `Generate Constructors` を選択してください。



3. constructor の引数にする変数を指定してください。 引数 0 の constructor と全てのメンバ変数が引数となる constructor を作成して下さい。



4. constructor が作成された事を確認して下さい。

```
public Archievement() {}

public Archievement(int id, String name, String memo) {
  this.id = id;
  this.name = name;
  this.memo = memo;
}
```

2.DAO クラスの作成

2-1.インターフェイスを作成(DAO クラス)

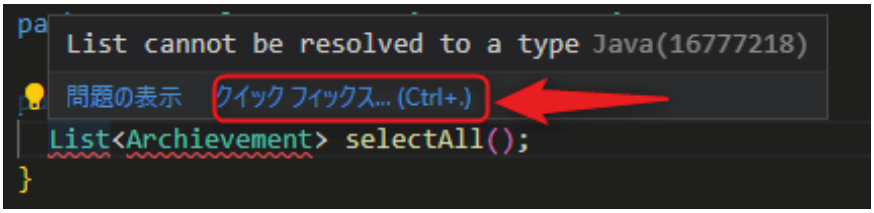
まずは、インターフェイスを作成しましょう。 `ArchievementDao.java` ファイルを作成してください。 今回は一覧画面の表示だけですので、テーブル上の全てのレコードを取得する `selectAll` メソッドを定義してください。 以下の構文で定義してください、戻り値のデータ型は `Archievement` クラスオブジェクトのリストとし、引数はなしとしてください。

戻り値のデータ型 メソッド名(引数1, 引数2, ...)

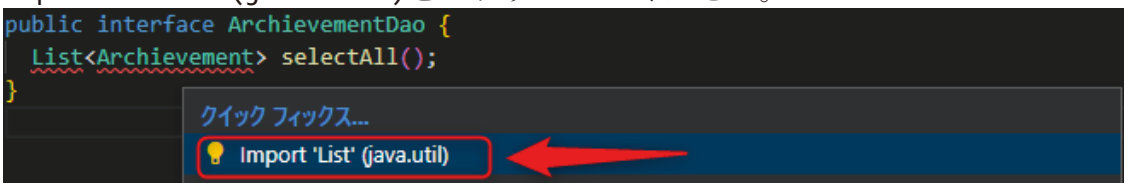
```
public interface ArchievementDao {
  List<Archievement> selectAll();
}
```

この状態では、得体のしれないクラスが戻り値として指定されている状態ですので、クイックフィックス機能を使って、クラスをインポートして下さい。

1. クイックフィックスをクリックしてください。



2. import 'List' (java.util)をクリックしてください。



同じように、Archievement クラスについても import を行って下さい。
今回は以下の 2 つが import されていれば OK です。

```
import java.util.List;
import com.classroom.assignment.entity.Archievement;
```

よくある間違えとして、異なるパッケージの同名のクラスを import することがよくあります。
自分の目的にあったクラスが import されている事を確認しましょう。

2-2.実装クラスの作成(DAO クラス)

ArchievementDaoImpl.java ファイルを作成してください。

1. まずはクラス名と実装を行うインタフェースを定義します。 実装クラスの名前は、インタフェース名＋Impl とする事が多いのでそのようにします。

   ```
   public class ArchievementDaoImpl implements ArchievementDao { }
   ```

SQL を実行するために、Spring で用意されている、JdbcTemplate クラスを使用します。 このクラスを使うためには、通常、都度 new を行ってクラスをインスタンスする必要があります。

```
JdbcTemplate jdbcTemplate = new JdbcTemplate();
```

Spring の機能である DI(Dependency Injection=依存性注入)という仕組みと使う事で、Spring が自動でインスタンスを生成して、変数に格納する事が出来ます。 この機能を使うためには constructor に@Autowired というアノテーションを付与し以下のように記述します。 ``` private final JdbcTemplate jdbcTemplate;

@Autowired public ArchievementDaoImpl(JdbcTemplate jdbcTemplate)
{ this.jdbcTemplate = jdbcTemplate; }

この際、`JdbcTemplate`クラスの import が必要となります。

import org.springframework.jdbc.core.JdbcTemplate;

2. `selectAll`メソッドを実装して下さい。
変数`sql`に、`ARCHIEVEMENT`テーブルから全ての列(id, name, memo)を取得する SELECT 文を設定して下さい。
`jdbcTemplate.queryForList(sql);`で SQL を実行し結果を取得しています。
取得した、結果を拡張 for 文を利用して`List<Archievement>`型の変数に格納し戻り値としてください。
※LessonDaoImpl`で行っている、「曜日番号 → 曜日名 変換」の処理は必要ありません。

3.サービスクラスの作成

3-1.インターフェイスを作成(サービスクラス)
`ArchievementService.java`ファイルを作成してください。
先ほどの DAO クラスと同じように、テーブル上の全てのレコードを取得する`selectAll`メソッドを定義してください。
戻り値のデータ型は`Archievement`クラスオブジェクトのリストとし、引数はなしとしてください。

3-2.実装クラスの作成(サービスクラス)
`ArchievementServiceImpl.java`ファイルを作成してください。
DI を使って、`ArchievementDao`クラスを注入してください。

private final ArchievementDao dao;

@Autowired public ArchievementServiceImpl(ArchievementDao dao) { this.dao = dao; }

先ほど実装した、`ArchievementDao`クラスの`selectAll`メソッドを呼び出し、`List<Archievement>`型の変数を取得し、取得した変数を返却してください。

4.ビューのテンプレートファイルの作成
なお、今回はテンプレートファイルを用意しておりますので、そちらを使ってください。
`~\resources\templates\archievement`配下に、`index.html`ファイルが存在する事を確認してください。

5. コントローラーの作成
`ArchievementController.java`ファイルを作成してください。
DI を使って、`ArchievementService`クラスを注入してください。

private final ArchievementService archievementService;

@Autowired public ArchievementController(ArchievementService archievementService) { this.archievementService = archievementService; } ```

パス/archievement に GET リクエストがあった場合にリクエストを処理される index メソッドを作成してください。 ArchievementService クラスの selectAll メソッドを使用して、実績一覧を取得し、model オブジェクトの archievementList 属性に値を設定し、 戻り値に先ほど存在を確認した、ビューのテンプレートファイル、archievement/index を設定してください。

課題の提出

- 課題の提出は課題提出の操作のコミット・プッシュ・プルリクエストを実施しましょう。

【Level3-3】

テーマ

既存プログラムを参考に、CRUD 機能が作成できること ページ遷移とバリデーションの仕組みが理解できること template への記載ができるようになること

演習課題の準備（開発環境について）

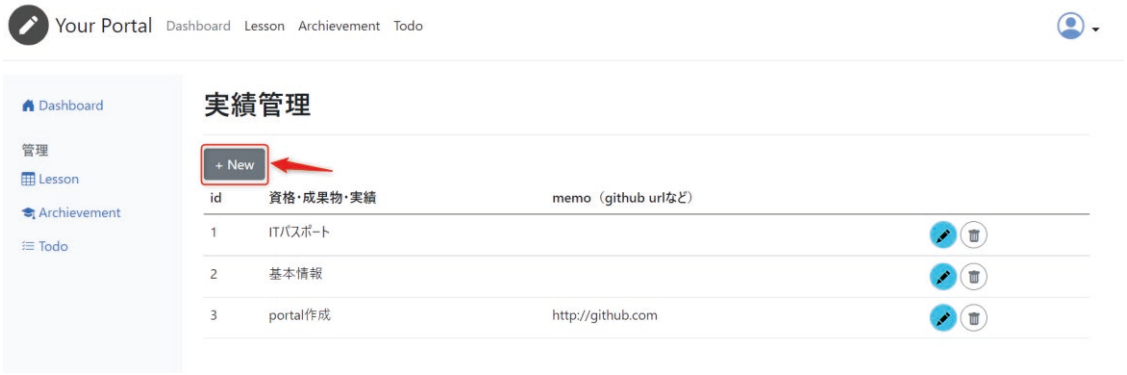
- この演習では、Codespaces と呼ばれるサービスを使って開発を行います。
- ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。
- レベル 0 で実施した手順を参照して、演習課題の準備をしましょう。

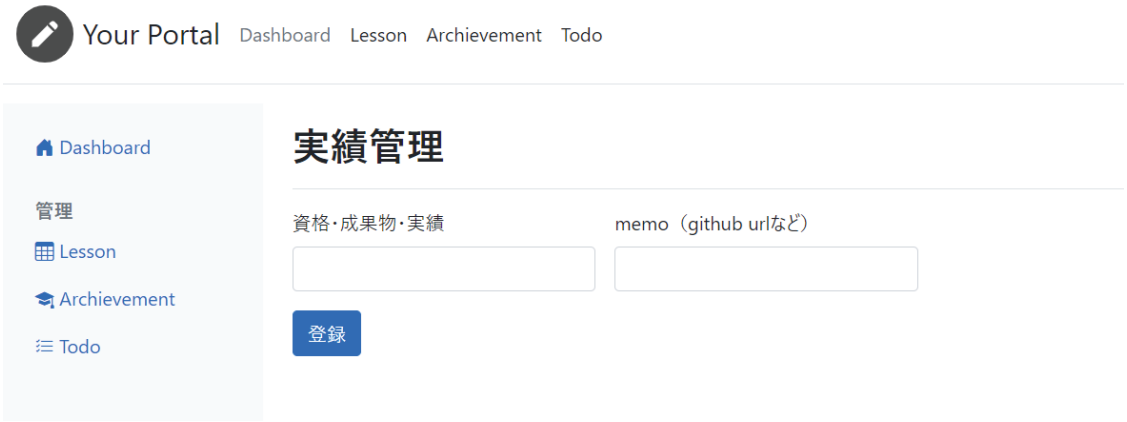
演習内容

Lesson(授業管理機能)を参考に Archievement(実績管理機能)の実績登録画面を作成しましょう。

演習内容詳細

実績登録画面を作成しましょう。 実績管理画面の New ボタンを押下して遷移します。





以下の順序で実装を行ってください。

1. リクエスト情報を保持する、Form クラスである `ArchievementForm` クラスを作成してください。
2. `ArchievementController` クラスに、登録画面を表示するために、`regist` メソッドを作成してください。
3. 登録画面で入力されたデータを格納するために、`ArchievementDao` クラスに `insertArchievement` メソッドを追加してください。
4. `ArchievementService` クラスに `insert` メソッドを追加し、`ArchievementDao` クラスの `insertArchievement` メソッドを呼び出してください。
5. `ArchievementController` クラスに、登録のためのリクエストを受け付ける `insert` メソッドを作成してください。

それぞれに対応する、授業管理機能(Lesson)のファイルを参考に作成してください。

Archievement	Lesson
ArchievementController	LessonController
ArchievementService	LessonService
ArchievementForm	LessonForm
Archievement	Lesson
ArchievementDao	LessonDao

1.Form クラスの作成

まずは Form クラスを作成しましょう、`ArchievementForm.java` ファイルを作成してください。 ブラウザから送信されたリクエストデータを保持するためのクラスです。 画面上の入力項目にフィールド変数が対応するように作成します。 ビュー

のテンプレートファイルとして~\templates\archievement\new.html がすでに用意されています。 このテンプレートファイルでは、name と memo という名前のフィールド変数が存在する、クラスのオブジェクトをやり取りして、リクエスト送信する、画面表示を行うようになってますので、name と memo という名前のフィールド変数持つクラスを作成します。 それぞれのフィールド変数は最終的には、Entity クラスのオブジェクトに格納され、データベースに insert されます、Entity クラスの対応するフィールド変数とデータ型は同じにしておきましょう。 フィールド変数 name には@NotEmpty(message = "入力してください")とアノテーションを付与してください、こうする事で、name 項目をフォームから入力する際に必須項目とすることが出来ます。 前回の演習内容を参考に Constructor(コンストラクター)と、Getter/Setter(ゲッター/セッター)を作成してください。

2.ArchievementController クラスに regist メソッドを追加

regist メソッドを作成してください、/archievement/new のパスに対してリクエストが送信された際に、このメソッドが使われれるように@GetMapping("/new")とアノテーションを付与してください。 入力項目に何も入力されていない状態で画面表示を行いたいので、model オブジェクトの属性値 archievementForm に空のオブジェクト(new ArchievementForm())を格納し、戻り値にビューのテンプレートファイル、archievement/new を設定してください。

3.ArchievementDao クラスに insertArchievement メソッドを追加

ArchievementDao クラスに insertArchievement メソッドを追加してください。 戻り値のデータ型は int、引数は Archievement クラスのオブジェクトを受け付けるようにしてください。 selectAll メソッドを作ったように、インターフェイス、実装クラスどちらにも追加してください。 変数 sql に、ARCHIEVEMENT テーブルにデータを追加するための INSERT 文を設定してください。 この時 VALUES のかっこ内の値を指定する個所には、「?」を指定してください。

```
String sql = "INSERT INTO ARCHIEVEMENT (name, memo) VALUES(?, ?)";
```

「?」はバインドパラメータと呼ばれる物です、データベースに対して、一度 SQL 文を送信した後、バインドパラメータに入れ込む値を別で送信します。 バインドパラメータを使用する事で、SQL インジェクションを防ぐ事が出来ます。
JdbcTemplate クラスの update メソッドを使用して SQL 文を実行します。 第一引数には、SQL 文を、第二引数以降は、バインドパラメータに指定したい値を設定します。 戻り値として、実行された SQL 文が影響を与えた行数が返却されます。 そのまま、insertArchievement メソッドの戻り値として返して下さい。 今回は 1 件 INSERT を行っただけですので 1 が戻り値として返されるはずです。

4.ArchievementService クラスに insert メソッドを追加

ArchievementService クラスに insertArchievement メソッドを追加してください。 戻り値はなし(void を指定)、引数は Archievement クラスのオブジェクトを受け付けるようにしてください。 selectAll メソッドを作ったように、インターフェイス、実装クラスどちらにも追加してください。 先ほど実装した、ArchievementDao クラスの insertArchievement メソッドを呼び出し、実行された SQL 文が影響を与えた行数を取得してください。 0 が返された場合は、INSERT の処理が正常に完了してませんので、NotInsertException クラスの例外を投げるようにしてください。

5.ArchievementController クラスに insert メソッドを追加

ArchievementController クラスに、登録のためのリクエストを受け付ける insert メソッドを追加してください。 ビューのテンプレートファイル、archievement/new にて、登録ボタンを押した際には、パス/archievement/insert に対して、POST リクエストを送信するように指定されています。 このリクエストを受け取る事が出来るように、@PostMapping("/insert")とアノテーションを付与してください。 引数部分は以下のように設定してください、それぞれの引数と付与されているアノテーションの意味について解説します。

```
@PostMapping("/insert")
  public String insert(
      @Validated @ModelAttribute("archievementForm") ArchievementForm arc
hievementForm,
      BindingResult result, Model model) {
```

- @Validated @ModelAttribute("archievementForm") ArchievementForm archievementForm
- @Validated ・・・ バリデーション(入力チェックを有効にするアノテーションです。)
- @ModelAttribute("archievementForm") ・・・ 属性名の指定が archievementForm であるリクエストデータを引数に格納する事を示しています。
- BindingResult result ・・・ 入力チェックの結果などが格納されます。
- Model model ・・・ ビューに値を渡すための引数です。

入力チェック結果にエラーが存在しないか確認し、存在すれば、再度画面表示を行うように指定します。 この時入力チェックの結果も返されます、入力チェックの結果をテンプレートエンジンが確認し、赤字でエラーメッセージを表示するよう HTML の生成を行っています。

```
if (result.hasErrors()) {
  model.addAttribute("archievementForm", archievementForm);
  return "archievement/new";
}
```

その後、Form クラスのオブジェクトから、Entity クラスのオブジェクトに入力データを移し替え、`ArchievementService` クラスの `insertArchievement` メソッドを呼び出し、登録処理を実施します。

```
Archievement archievement = new Archievement();
archievement.setName(archievementForm.getName());
archievement.setMemo(archievementForm.getMemo());

archievementService.insertArchievement(archievement);
```

その後、`/archievemen` のパスに対してリダイレクトを行うよう、戻り値に指定します。 リダイレクトする事によって、同クラスの `index` メソッドが実行され、実績一覧の画面が表示されます。

課題の提出

- 課題の提出は課題提出の操作のコミット・プッシュ・プルリクエストを実施しましょう。

【Level3-4】

テーマ

既存プログラムを参考に、CRUD 機能が作成できること ページ遷移とバリデーションの仕組みが理解できること template への記載ができるようになること

演習課題の準備（開発環境について）

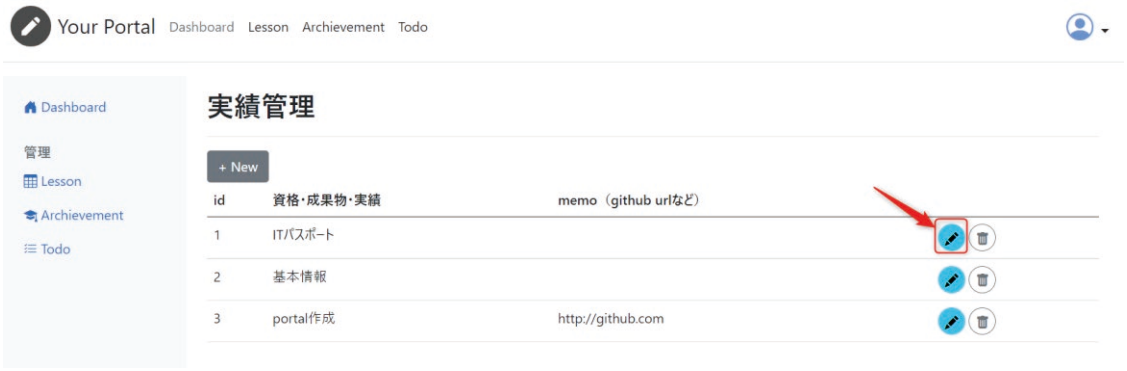
- この演習では、Codespaces と呼ばれるサービスを使って開発を行います。
- ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。
- レベル 0 で実施した手順を参照して、演習課題の準備をしましょう。

演習内容

Lesson(授業管理機能)を参考に Archievement(実績管理機能)の実績変更画面を作成しましょう。

演習内容詳細

実績登録画面を作成しましょう。 実績管理画面の詳細ボタンを押下して遷移します。



以下の順序で実装を行ってください。

1. リクエスト情報を保持する、Form クラスである `ArchievementForm` クラスに `id` 項目を追加してください。
2. 変更画面で表示する、登録済みの実績情報取得するため、`ArchievementDao` クラスに `findById` メソッドを追加してください。
3. 変更画面で入力された内容で、実績を更新するため `ArchievementDao` クラスに `updateArchievement` メソッドを追加してください。
4. 今回追加した DAO クラスのメソッドを使用するため `ArchievementService` クラスに `findArchievement` メソッドを追加してください。
5. 今回追加した DAO クラスのメソッドを使用するため `ArchievementService` クラスに `updateArchievement` メソッドを追加してください。
6. `ArchievementController` クラスに、変更画面を表示するための、`detail` メソッドを追加してください。
7. `ArchievementController` クラスに、変更処理を行うための、`update` メソッドを追加してください。

それぞれに対応する、授業管理機能(Lesson)のファイルを参考に作成してください。

Archievement	Lesson
ArchievementController	LessonController
ArchievementService	LessonService
ArchievementForm	LessonForm
Archievement	Lesson
ArchievementDao	LessonDao

1.ArchievementForm クラスへの id 項目の追加

実績の変更画面を作るに当たって、どの実績に対する更新処理を行うかを特定する必要があります。 更新時に、実績毎に割り振られている id もリクエストデータ含めるようにし、どの実績に対する更新であるかを特定出来るようにします。 ビューのテンプレートファイルとして~\templates\archievement\detail.html がすでに用意されています。 このテンプレートファイルでは、id と name と memo という名前のフィールド変数が存在する、クラスのオブジェクトをやり取りして、リクエスト送信する、画面表示を行うようになってますので、id と name と memo という名前のフィールド変数持つクラスが必要です。 前回の演習で作成した、ArchievementForm クラスへ id を追加して対応する事としてください。 なお、id はユーザ入力値とはなりませんので、入力チェックの必要有りません。

2.ArchievementDao クラスに findById メソッドを追加

ArchievementDao クラスに findById メソッドを追加してください。 戻り値のデータ型は Archievement クラス、引数は id を int 型で受け付けるようにしてください。 インターフェイス、実装クラスどちらにも追加してください。 変数 sql に、ARCHIEVEMENT テーブルから、id を指定して、データを抽出する SELECT 文を設定してください。 この時条件となる id の値を指定する個所には、「?」を指定してください。

```
String sql = "SELECT id, name, memo FROM ARCHIEVEMENT WHERE id = ?";
```

主キーである、id を指定して抽出してますので、必ず 1 件のみ取得されるはずです。 このような場合は、JdbcTemplate クラスの queryForMap メソッドを使用して SQL 文を実行し結果を取得します。 第一引数には、SQL 文を、第二引数以降は、バインドパラメータに指定したい値を設定します。 List 型ではない、単純に Map 型のオブジェクトで得られますので、Archievement クラスのオブジェクトに詰め替え、戻り値とします。

```
Map<String, Object> result = jdbcTemplate.queryForMap(sql, id);
Archievement archievement = new Archievement();
archievement.setId((int) result.get("id"));
archievement.setName((String) result.get("name"));
archievement.setMemo((String) result.get("memo"));
return archievement;
```

3.ArchievementDao クラスに updateArchievement メソッドを追加

ArchievementDao クラスに updateArchievement メソッドを追加してください。 戻り値のデータ型は int、引数は Archievement クラスのオブジェクトを受け付ける

ようにしてください。 インターフェイス、実装クラスどちらにも追加してください。 変数 sql に、ARCHIEVEMENT テーブルに対して更新を行う、UPDATE 文を設定してください。 値を指定する個所には、「?」を指定してください。

```
String sql = "UPDATE ARCHIEVEMENT SET name = ?, memo = ? WHERE id = ?";
```

JdbcTemplate クラスの update メソッドを使用して SQL 文を実行します。 第一引数には、SQL 文を、第二引数以降は、バインドパラメータに指定したい値を設定します。 戻り値として、実行された SQL 文が影響を与えた行数が返却されます。そのまま、メソッドの戻り値として返して下さい。 今回は id を指定して、1 件 UPDATE を行いますので、必ず 1 が返されるはずです。

4.ArchievementService クラスに findArchievement メソッドを追加

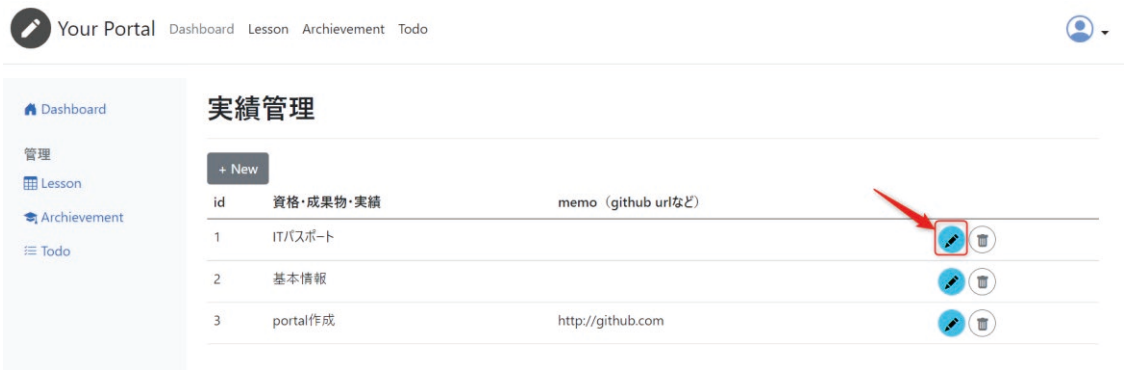
ArchievementService クラスに findArchievement メソッドを追加してください。戻り値のデータ型は Archievement クラス、引数は id を int 型で受け付けるようにしてください。 インターフェイス、実装クラスどちらにも追加してください。先ほど実装した、ArchievementDao クラスの findById メソッドを呼び出し、Archievement 型の変数を取得し、取得した変数を返却してください。

5.ArchievementService クラスに updateArchievement メソッドを追加

ArchievementService クラスに updateArchievement メソッドを追加してください。 戻り値のデータ型は int、引数は Archievement クラスのオブジェクトで受け付けるようにしてください。 インターフェイス、実装クラスどちらにも追加してください。 先ほど実装した、ArchievementDao クラスの updateArchievement メソッドを呼び出し、実行された SQL 文が影響を与えた行数を取得してください。0 が返された場合は、INSERT の処理が正常に完了してませんので、NotUpdateException クラスの例外を投げるようにしてください。

6. ArchievementController クラスに、変更画面を表示するための、detail メソッドを追加

実績管理が面面の詳細ボタンのリンクには id が含まれています。 例えば画像の「id=1」のリンクを押下すると/archievement/detail/1 というパスへ遷移します。



このようにパスの最後にどの ID の内容を変更するかを指定し、画面遷移をしています。 このように、パスに引数を持たせた物を、パスパラメータと呼びます。 詳細ボタン押下時のリクエストを処理するために、`detail` メソッドに次のようなアノテーションと引数を指定します。

```
@GetMapping("/detail/{id}")
  public String detail(@PathVariable("id") int id, Model model) {
```

{id}で「id」という属性名で、パスパラメータがある事を示し、`@PathVariable("id")`のアノテーションを引数の前につける事で、「id」という属性名のパラメータ値を `int id` に代入するといった事を行ってくれています。 取得した `id` で `ArchievementService` クラスの `findArchievement` メソッドを利用し、`Archievement` クラスのオブジェクトを取得し、`ArchievementForm` クラスのオブジェクトへ入れ替え、`model` オブジェクトの `archievementForm` 属性に値を設定し、変更画面のビューのテンプレートファイルである、`archievement/update` を設定してください。

7. `ArchievementController` クラスに、変更処理を行うための、`update` メソッドを追加

`ArchievementController` クラスに、登録のためのリクエストを受け付ける `update` メソッドを追加してください。 ビューのテンプレートファイル、`archievement/update` にて、更新ボタンを押した際には、パス `/archievement/update` に対して、POST リクエストを送信するように指定されています。 このリクエストを受け取る事が出来るように、`@PostMapping("/update")` とアノテーションを付与してください。 引数部分は以下のように設定してください、それぞれの引数と付与されているアノテーションの意味について解説します。

```
@PostMapping("/update")
  public String update(
      @Validated @ModelAttribute("archievementForm") ArchievementForm arc
```

```
hievementForm,
      BindingResult result, Model model) {
```

- @Validated @ModelAttribute("archievementForm") ArchievementForm archievementForm
- @Validated ・・・ バリデーション(入力チェックを有効にするアノテーションです。)
- @ModelAttribute("archievementForm") ・・・ 属性名の指定が `archievementForm` であるリクエストデータを引数に格納する事を示しています。
- BindingResult result ・・・ 入力チェックの結果などが格納されます。
- Model model ・・・ ビューに値を渡すための引数です。

入力チェック結果にエラーが存在しないか確認し、存在すれば、再度画面表示を行うように指定します。 この時入力チェックの結果も返されます、入力チェックの結果をテンプレートエンジンが確認し、赤字でエラーメッセージを表示するよう HTML の生成を行っています。

```
if (result.hasErrors()) {
  model.addAttribute("archievementForm", archievementForm);
  return "archievement/update";
}
```

その後、Form クラスのオブジェクトから、Entity クラスのオブジェクトに入力データを移し替え、`ArchievementService` クラスの `updateArchievement` メソッドを呼び出し、更新処理を実施します。

```
Archievement archievement = new Archievement();
archievement.setId(archievementForm.getId());
archievement.setName(archievementForm.getName());
archievement.setMemo(archievementForm.getMemo());

archievementService.updateArchievement(archievement);
```

その後、`/archievement` のパスに対してリダイレクトを行うよう、戻り値に指定します。 リダイレクトする事によって、同クラスの `index` メソッドが実行され、実績一覧の画面が表示されます。

課題の提出

- 課題の提出は課題提出の操作のコミット・プッシュ・プルリクエストを実施しましょう。

【Level3-5】

テーマ

既存プログラムを参考に、CRUD 機能が作成できること ページ遷移とバリデーションの仕組みが理解できること template への記載ができるようになること

演習課題の準備（開発環境について）

- この演習では、Codespaces と呼ばれるサービスを使って開発を行います。
- ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。
- レベル 0 で実施した手順を参照して、演習課題の準備をしましょう。

演習内容

lesson を参考に実績削除機能を作成しましょう。

演習内容詳細

実績削除機能を作成しましょう。
実績管理画面のゴミ箱ボタンを押すことで機能するようにしましょう。

以下の順序で実装を行ってください。

1. id 指定で実績を削除するために ArchievementDao クラスに deleteArchievement メソッドを追加してください。
2. 今回追加した DAO クラスのメソッドを使用するため ArchievementService クラスに deleteArchievement メソッドを追加してください。
3. ArchievementController クラスに、削除のリクエストを処理する delete メソッドを追加してください。

それぞれに対応する、授業管理機能(Lesson)のファイルを参考に作成してください。

Archievement	Lesson
ArchievementController	LessonController
ArchievementService	LessonService
ArchievementForm	LessonForm
Archievement	Lesson
ArchievementDao	LessonDao

1.ArchievementDao クラスに deleteArchievement メソッドを追加

JdbcTemplate クラスの update メソッドを使用して以下の SQL 文を実行するようにしてください。
先ほどの演習で作成した、updateArchievement メソッドなどを参考に作成してみましょう。

```
DELETE FROM ARCHIEVEMENT WHERE id = ?
```

戻り値として、実行された SQL 文が影響を与えた行数(削除件数)を返すようにしてください。

2.ArchievementService クラスに delete メソッドを追加

ArchievementDao クラスの deleteArchievement メソッドを呼び出し、id 指定で削除を行うようにしてください。
戻り値として 0 が返された場合は、DELETE の処理が正常に完了してませんので、NotDeleteException クラスの例外を投げるようにしてください。 先ほどの演習で作成した、updateArchievement メソッドなどを参考に作成してみましょう。

3. `ArchievementController` クラスに、削除処理を行うための、`delete` メソッドを追加

パス`/detail/{id}`に対して送信された GET リクエストを処理するようにしてください。 先ほどの演習で作成した `detail` メソッドなどを参考に作成してみましょう。 処理後は、`/archievemen` のパスに対してリダイレクトを行うようにしてください。

課題の提出

- 課題の提出は課題提出の操作のコミット・プッシュ・プルリクエストを実施しましょう。

【Level3-6】

テーマ

既存プログラムを参考に、CRUD 機能が作成できること ページ遷移とバリデーションの仕組みが理解できること template への記載ができるようになること

演習課題の準備（開発環境について）

- この演習では、Codespaces と呼ばれるサービスを使って開発を行います。
- ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。
- レベル 0 で実施した手順を参照して、演習課題の準備をしましょう。

演習内容

potal ページに実績一覧を表示しよう

演習内容詳細

potal ページに実績一覧を表示出来るようにしましょう、実績は全件表示してください。

以下の順序で実装を行ってください。

1. `PortalController` クラスの、`index` メソッドに処理を追加して下さい。

1. `PortalController` クラスの、`index` メソッドに処理を追加

`ArchievementService` クラスの `selectAll` メソッドを利用し、実績一覧を取得し、`index` のテンプレートファイルを参照し、適切な属性名で、取得した実績一覧を model オブジェクトに設定してください。

課題の提出

- 課題の提出は課題提出の操作のコミット・プッシュ・プルリクエストを実施しましょう。

【Level4】

テーマ

要件をもとに自力で CRUD 機能が作成できること、自力で thymeleaf の記述ができることを目標に演習に取り組みましょう。

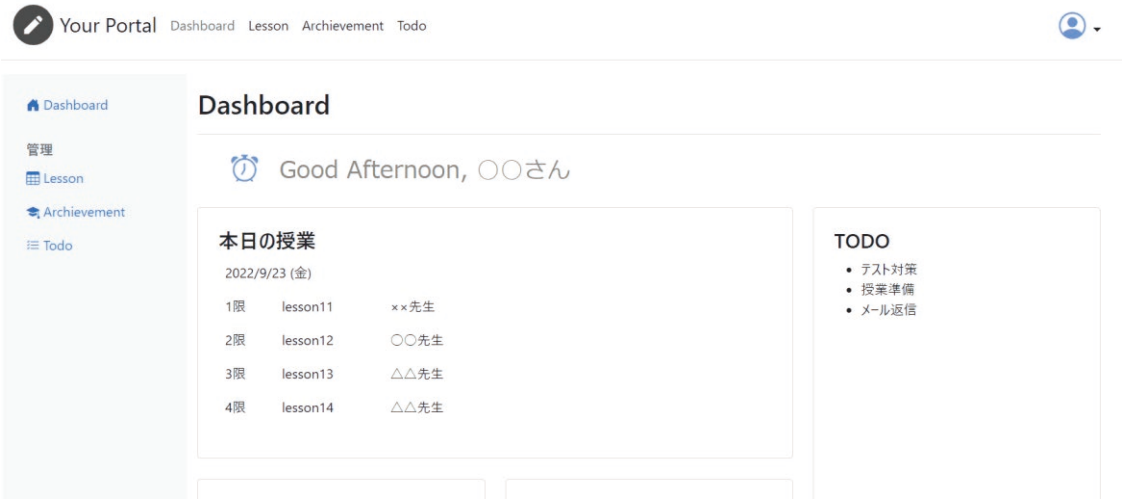
演習課題の準備（開発環境について）

- この演習では、Codespaces と呼ばれるサービスを使って開発を行います。
- ブラウザ上で動作する開発環境です、インストール不要で使う事ができます。
- レベル 0 で実施した手順を参照して、演習課題の準備をしましょう。

演習内容

TODO リスト機能を実装しましょう。

演習内容詳細



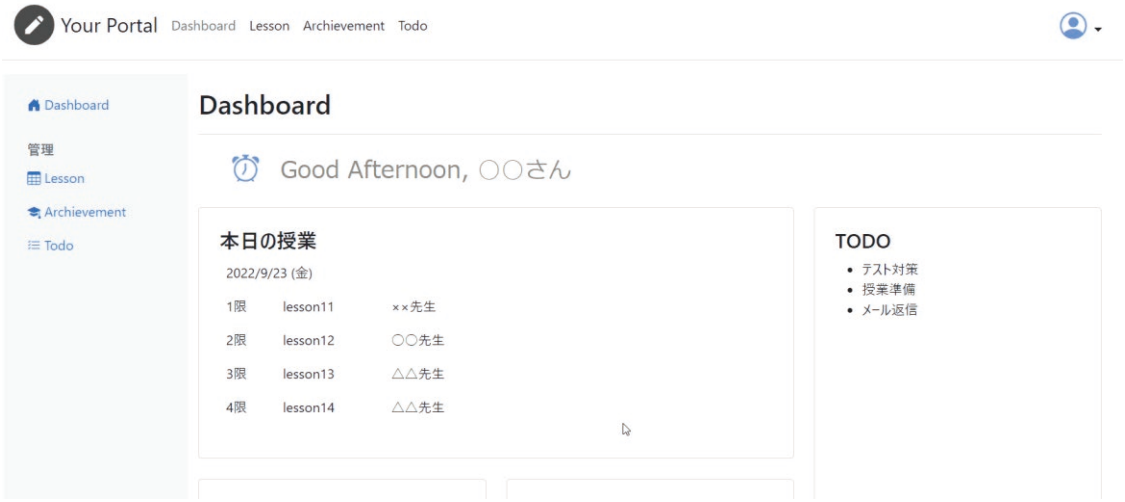
4_main

TODO の管理を行う機能を実装しましょう、これまでの演習内容を参考に取り組みましょう。
以下の機能を実装してください、全て 1 つのページで完結するように作成しましょう。
ビューのテンプレートファイルは既に用意されている物を使用してください。
`src\main\resources\templates\todo\index.html` に配置されています。

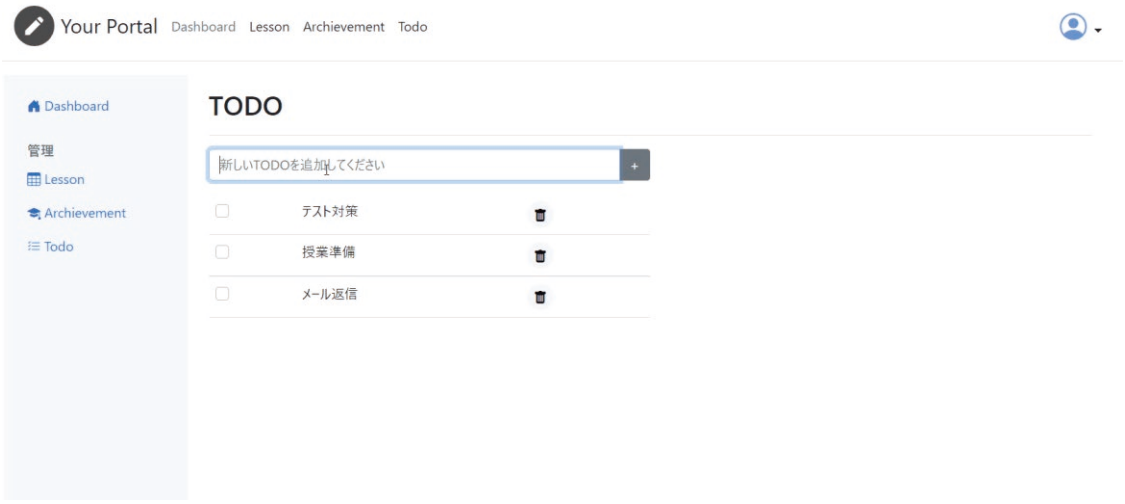
model オブジェクトへ値を格納する際に使用する属性名は、テンプレートファイルを参照し、適切なものを設定してください。

実装する機能について

1. 表示機能 `/todo` のパスのページを参照した際に表示される画面に、TODO の一覧を表示してください。

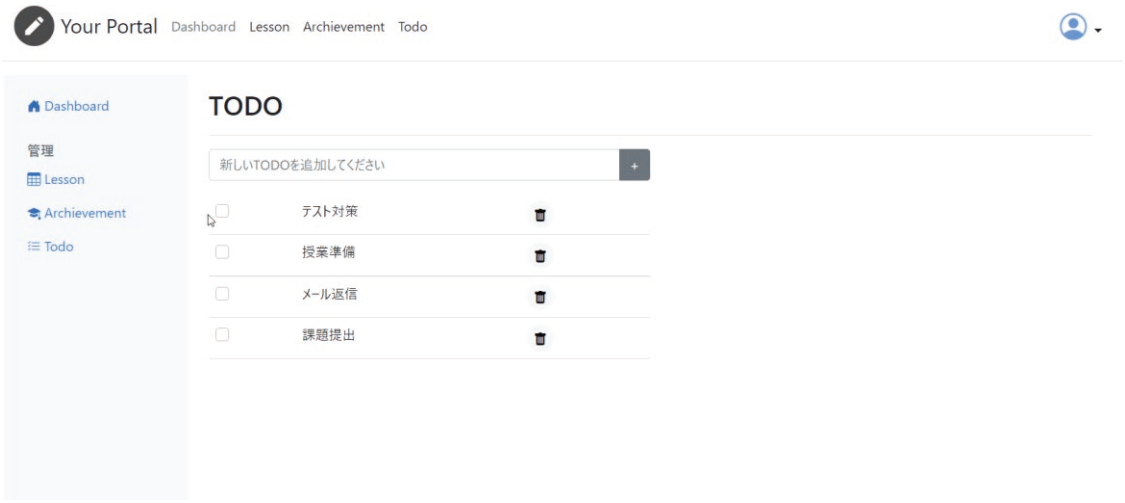


2. 登録機能 入力欄に入力された内容を TODO として登録出来るようにしてください。

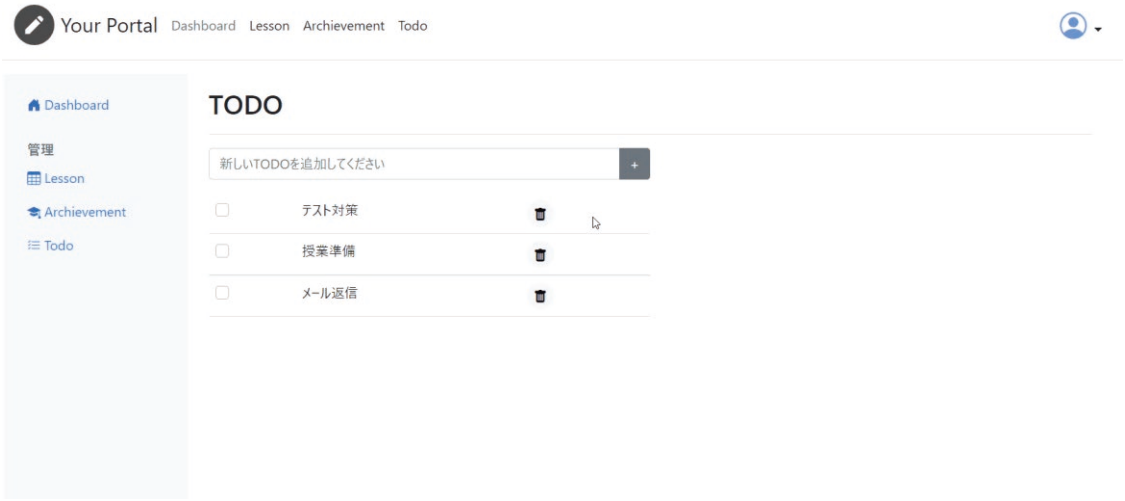


3. 更新機能(TODO の完了未完了) 表示された TODO のチェックボタンの状態を保持して下さい、チェックボタンを押すたびに `TODO` テーブルの `checked` カラ

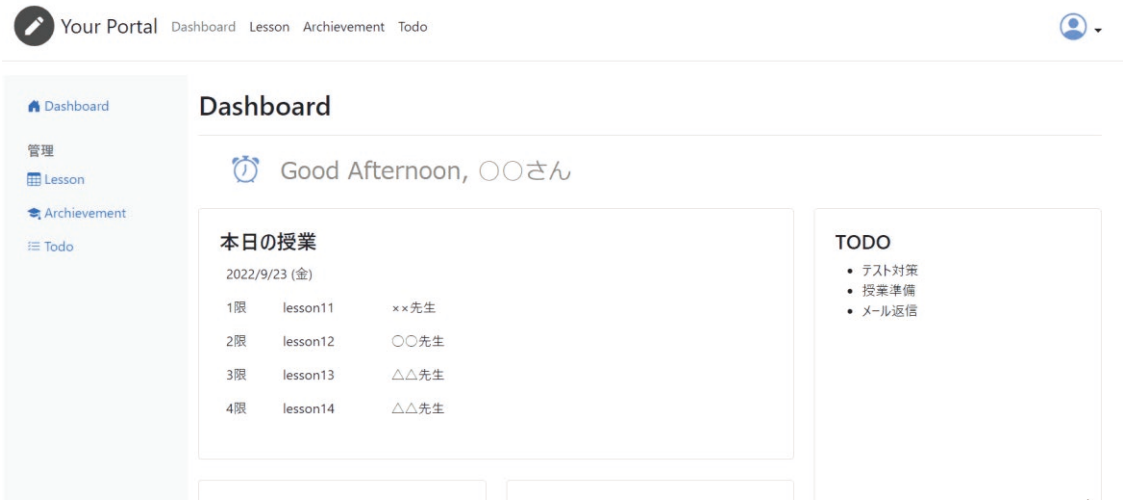
ムの値を更新してください。



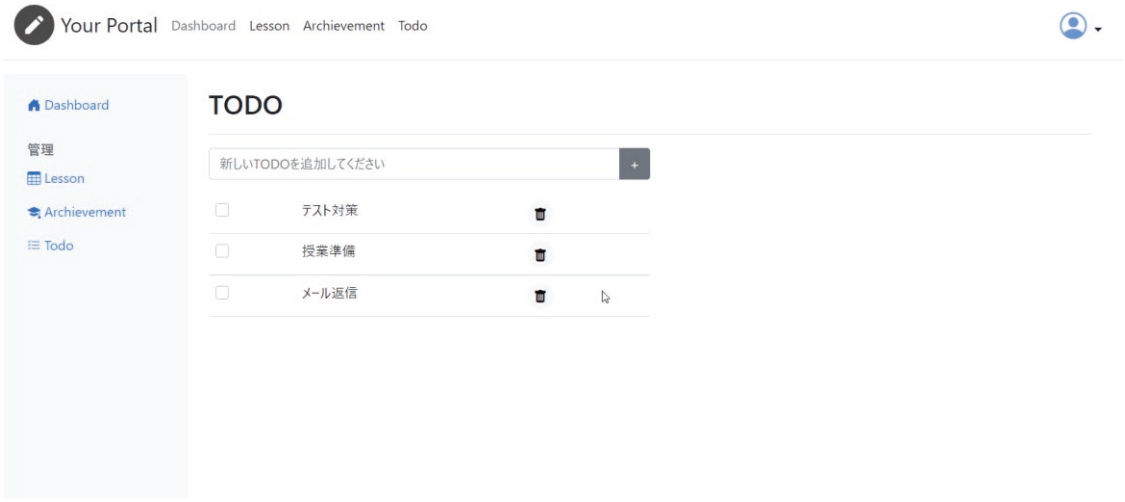
4. 削除機能 削除ボタンが押された行のレコードを TODO テーブルから物理削除してください。



5. 未完了の TODO の Dashboard への表示機能 未完了の TODO を Dashboard ページに表示してしてください。



6. バリデーションチェック機能 TODO を入力せずに登録ボタンを押した際に、「入力してください」とメッセージが出るようにしてください。



7. エラーハンドリング機能 以下のエラー時に、指定された例外クラスの例外を throw してください。

課題の提出

- 課題の提出は課題提出の操作のコミット・プッシュ・プルリクエストを実施しましょう。

演習ユニット - コーディング規約 -

概要

- 演習ユニット内で指定するコーディング規約について記載する

教育PFで指定されている規約一覧

対象ファイル

- java, properties, xml

規約

FileTabCharacter

- タブが含まれていないか

OuterTypeFilename

- ファイル名とClassの型名が一致しているか

IllegalTokenText

- 禁止した記述が無いか

AvoidEscapedUnicodeCharacters

- Unicodeエスケープを使用していないか

AvoidStarImport

- Import文にアスタリスク(*)が含まれていないか

OneTopLevelClass

- トップレベルクラスが自身のソースファイル以外で定義されていないか

NoLineWrap

- 改行されていないか

EmptyBlock

- 空ブロックが無いか

NeedBraces

- コードが{}で囲まれているか

LeftCurly

- {の記述ルール

RightCurly

- }の記述ルール。主にIF-ELEやTRY-CATCH-FINALLYのように条件文が分岐する場合の書き方

WhitespaceAfter

- tokenの後の空白スペースが無いかチェックする

WhitespaceAround

- Tokenの'前後'に空白スペースが無いか

OneStatementPerLine

- 1行に1つのstatmentが記述されているか

MultipleVariableDeclarations

- 変数の宣言が 1 行にまとめられているか

ArrayTypeStyle

- 配列の宣言の記述をチェックする

MissingSwitchDefault

- Switch文にdefaultが記述されているか

FallThrough

- switch文の各条件内の処理にbreak,return,throw,continueなどが記述されているか

UpperEll

- long型の定数宣言が大文字のLとなっているか

ModifierOrder

- 修飾子の順番がJavaの推奨に従っているか

EmptyLineSeparator

- tokenの間に空行を入れているか

SeparatorWrap

- 改行の仕方が規定通りか

PackageName

- packageの命名規則

TypeName

- Class,Interface,enum,annotationの命名規則

MemberName

- メンバ変数の命名規則

ParameterName

- メソッドのパラメータの命名規則

LambdaParameterName

- ラムダ式の引数の名前をチェック

CatchParameterName

- chatchパラメータの命名規則

LocalVariableName

- ローカル変数の命名規則

PatternVariableName

- パターン変数名が指定されたパターンに合致しているか

ClassTypeParameterName

- クラスの型パラメータの命名規則(総称型の仮型引数など)

RecordComponentName

- インターフェースの型パラメータの命名規則

RecordTypeParameterName

- レコード型パラメータ名が指定されたパターンに適合しているか

MethodTypeParameterName

- メソッドの型パラメータの命名規則

InterfaceTypeParameterName

- インターフェースの型パラメータの命名規則

NoFinalizer

- ファイナライザーが呼ばれていないこと

GenericWhitespace

- 総称型の宣言<>にスペースが入っていないか

Indentation

- インデントレベルが正しいか

AbbreviationAsWordInName

- 略称が規定通りか

NoWhitespaceBeforeCaseDefaultColon

- スイッチブロックのコロンの前に空白がないこと

OverloadMethodsDeclarationOrder

- オーバーロードメソッドがまとまって宣言されていること

VariableDeclarationUsageDistance

- 変数を宣言してから使用するまでのSTEP数が規規定通りか

CustomImportOrder

- Import宣言の順番を規定する

MethodParamPad

- メソッドのパラメータの記述が規定通りか

NoWhitespaceBefore

- tokenの前に空白スペースがないか

ParenPad

- 括弧とパラメータの間の記述が規定通りか

OperatorWrap

- 演算子で改行する際の改行位置のルール

AnnotationLocation

- アノテーションを付ける位置に関する規約

NonEmptyAtclauseDescription

- JavaDocTokenに説明があるか

InvalidJavadocPosition

- Javadoc が正しい位置に配置されているかどうか

JavadocTagContinuationIndentation

- Javadocの説明を改行するときのインデントルール

SummaryJavadoc

- Javadocに非推奨なフレーズが無いか

JavadocParagraph

- Javadocの段落定義

RequireEmptyLineBeforeBlockTagGroup

- Javadocにブロックタグがある場合、その前に1行の空行があるかどうか

AtclauseOrder

- Javadoの@の宣言順

JavadocMethod

- メソッドのJavadoc定義

MissingJavadocMethod

- メソッドのJavadocコメント有無をチェック

MissingJavadocType

- クラス、enum、インターフェース、およびアノテーション・インターフェース定義の Javadoc コメントが欠落していないかどうか

MethodName

- メソッドの命名規則

SingleLineJavadoc

- Javadocが１行で収まる場合に@が含まれていないこと

EmptyCatchBlock

- 空のcatchブロックが無いこと

CommentsIndentation

- コメント行のインデントをコードのインデントと揃えるか

実習ユニット

実習ユニット使用方法

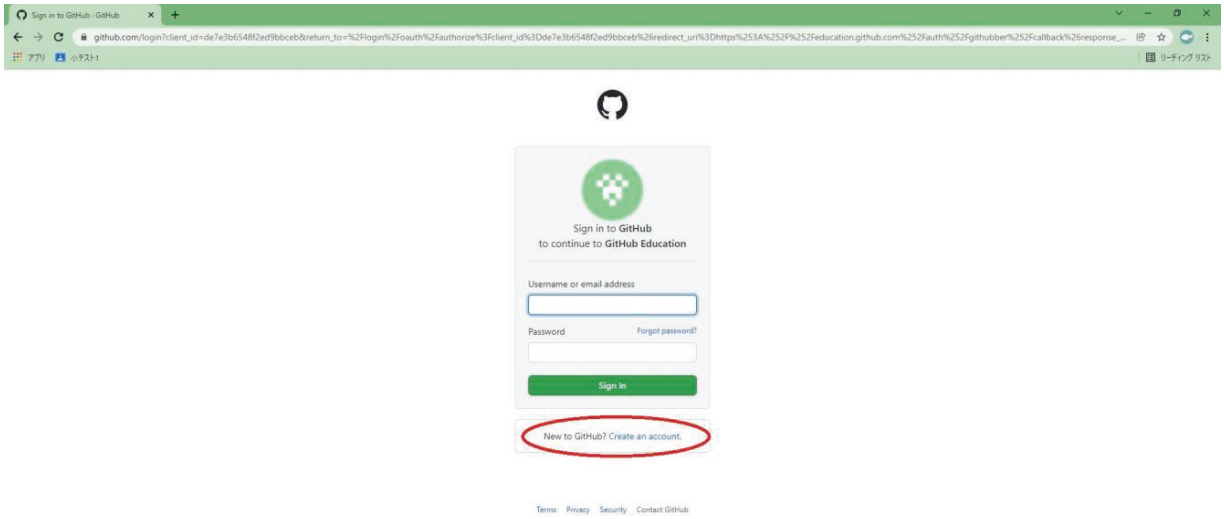
GitHub の登録について

---

1. 以下のリンクから Github のサインイン https://github.com/

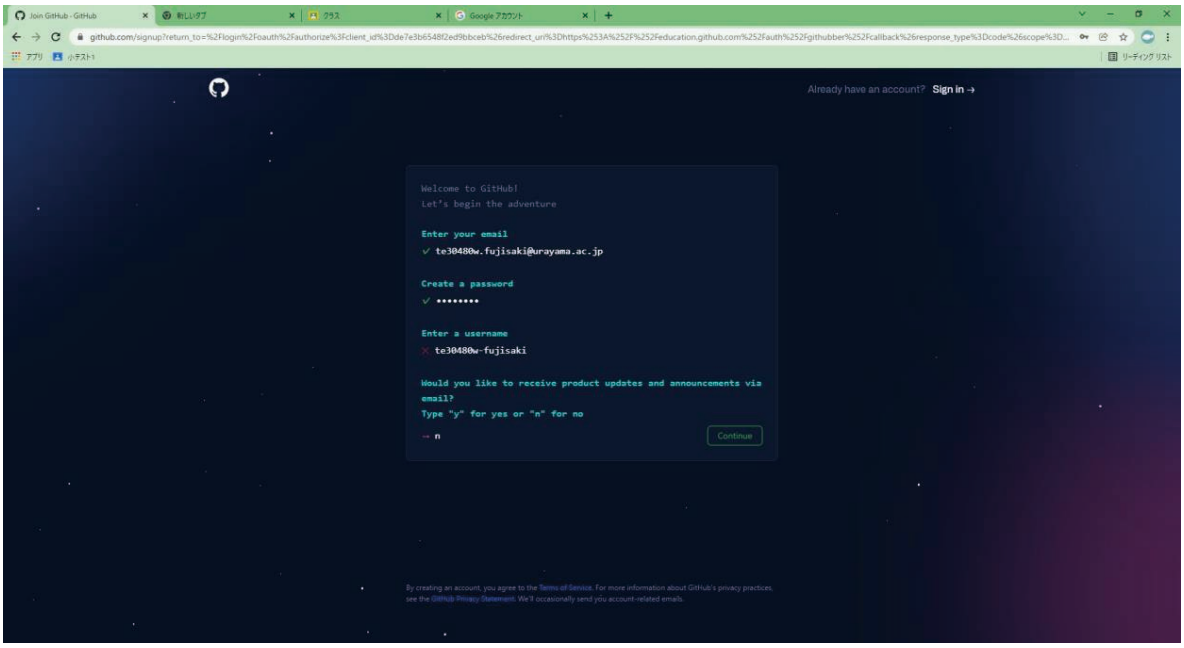
2. 画面右上の Sign in をクリックする

3. Sign in ページで Create an account. をクリックする

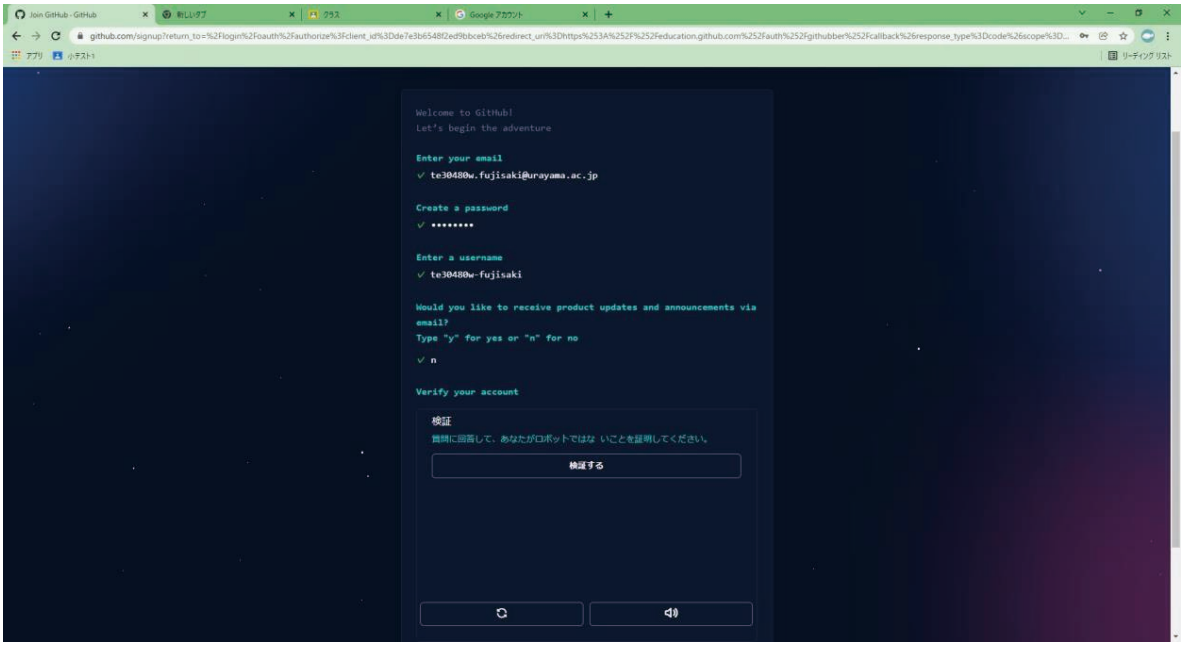


4. 登録ページが出てくるので順番に入力を行う

    1. 配布されたメールアドレス (@urayama.ac.jp のもの)
    2. 配布されたパスワード
    3. 配布されたユーザー名
    4. n を入力 (もしアップデート情報が欲しい場合は y を入力)

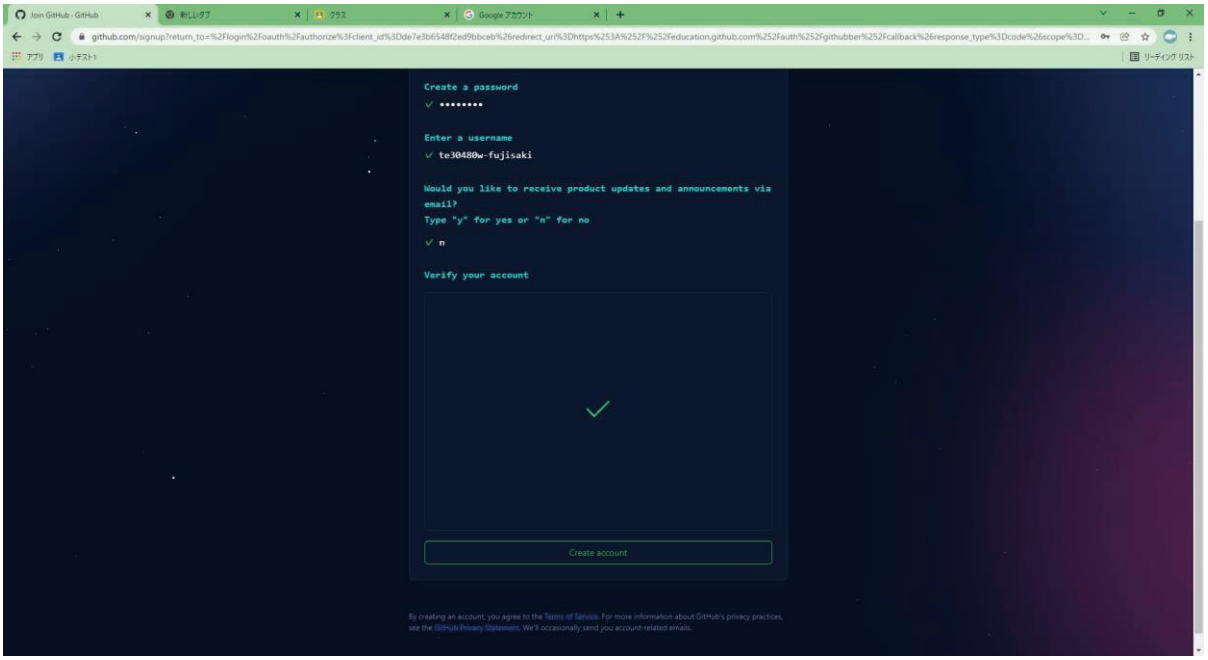


5. 検証するを実施

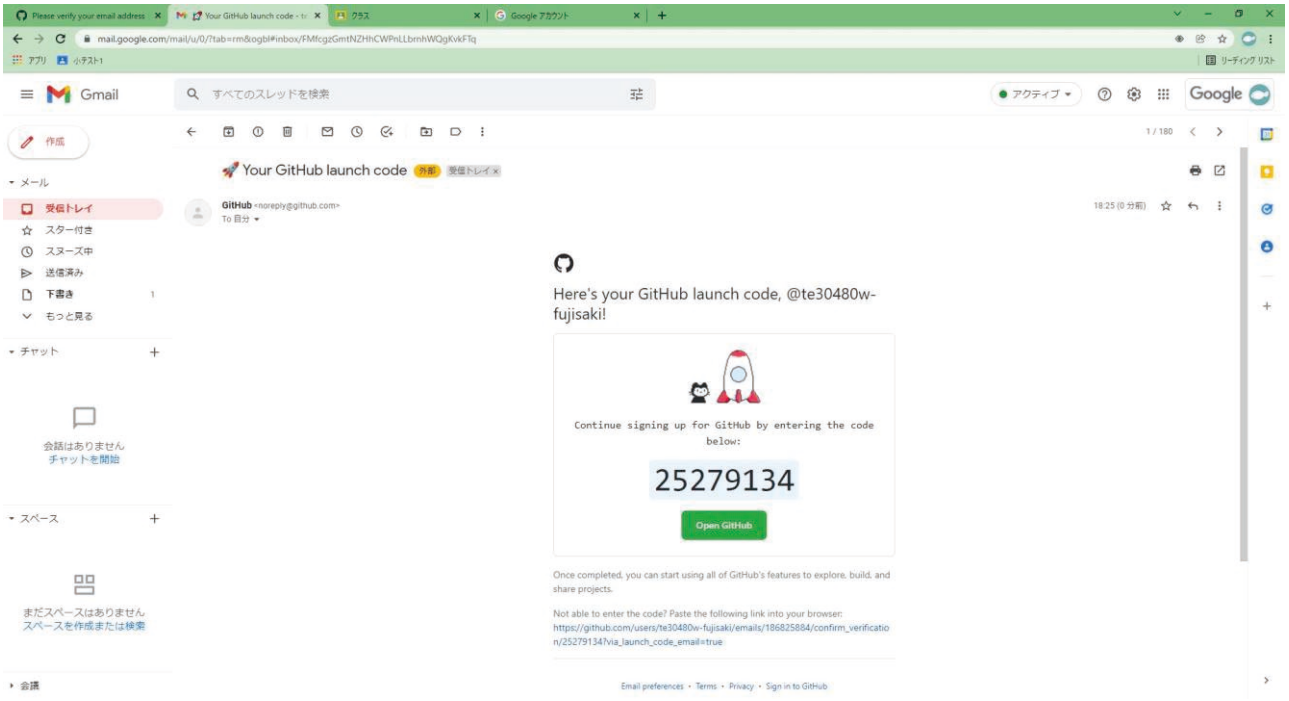


.

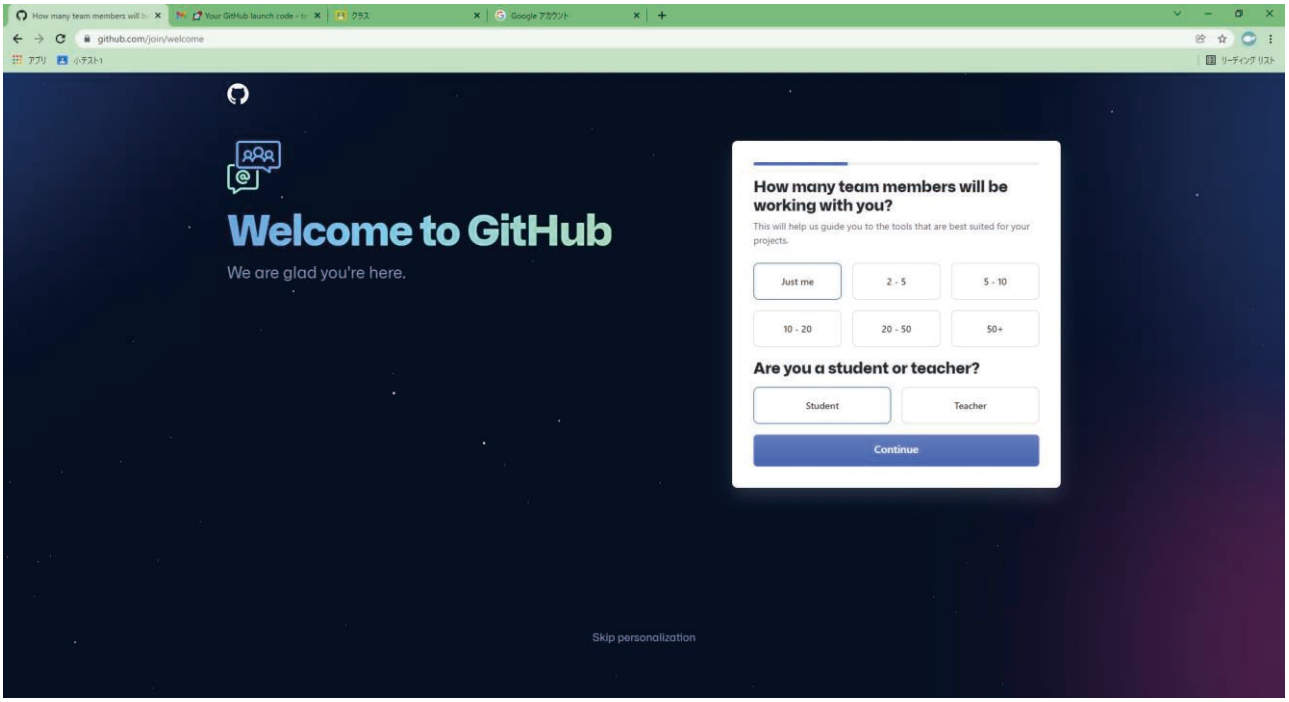
6. すべての項目を入力後、Create account をクリック



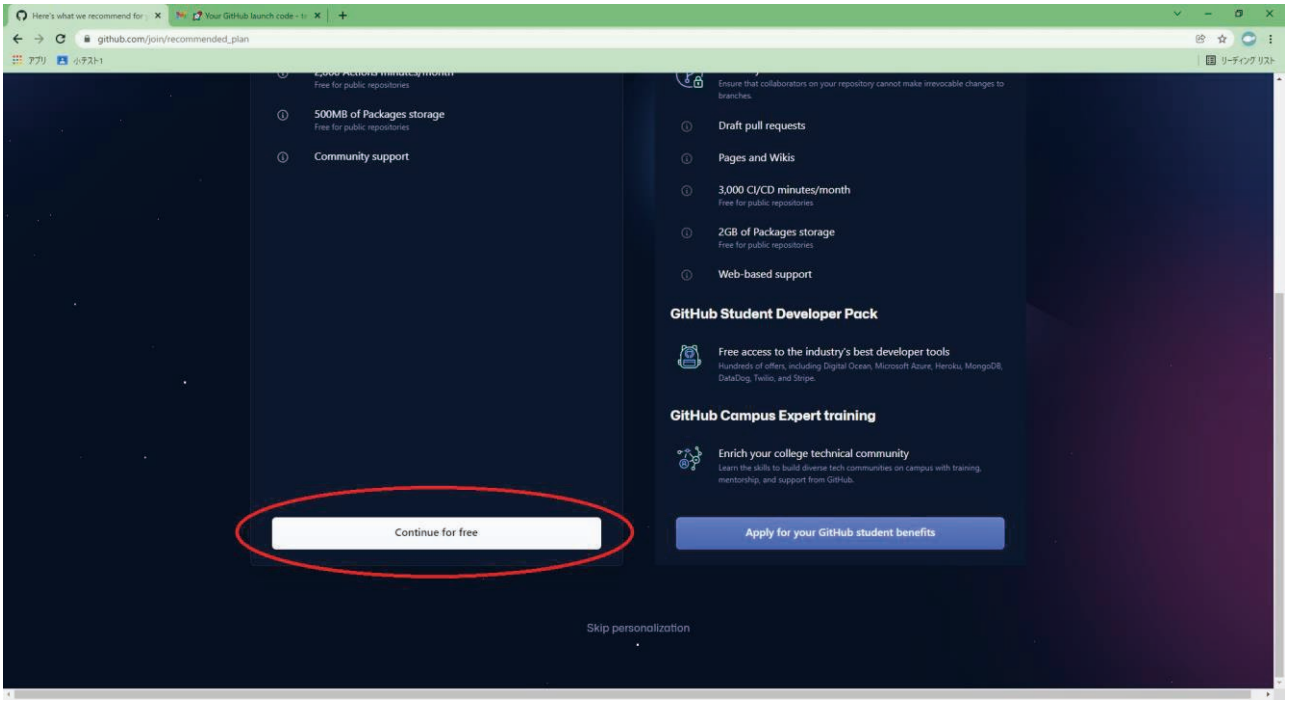
7.承認メールが届くので、配布されたメールアドレス (@urayama.ac.jp のもの)のメールを確認し数字を入力する



8.承認後、以下の画面の選択を行う



9 項目6 の画面の選択後、Continue for free をクリック



.

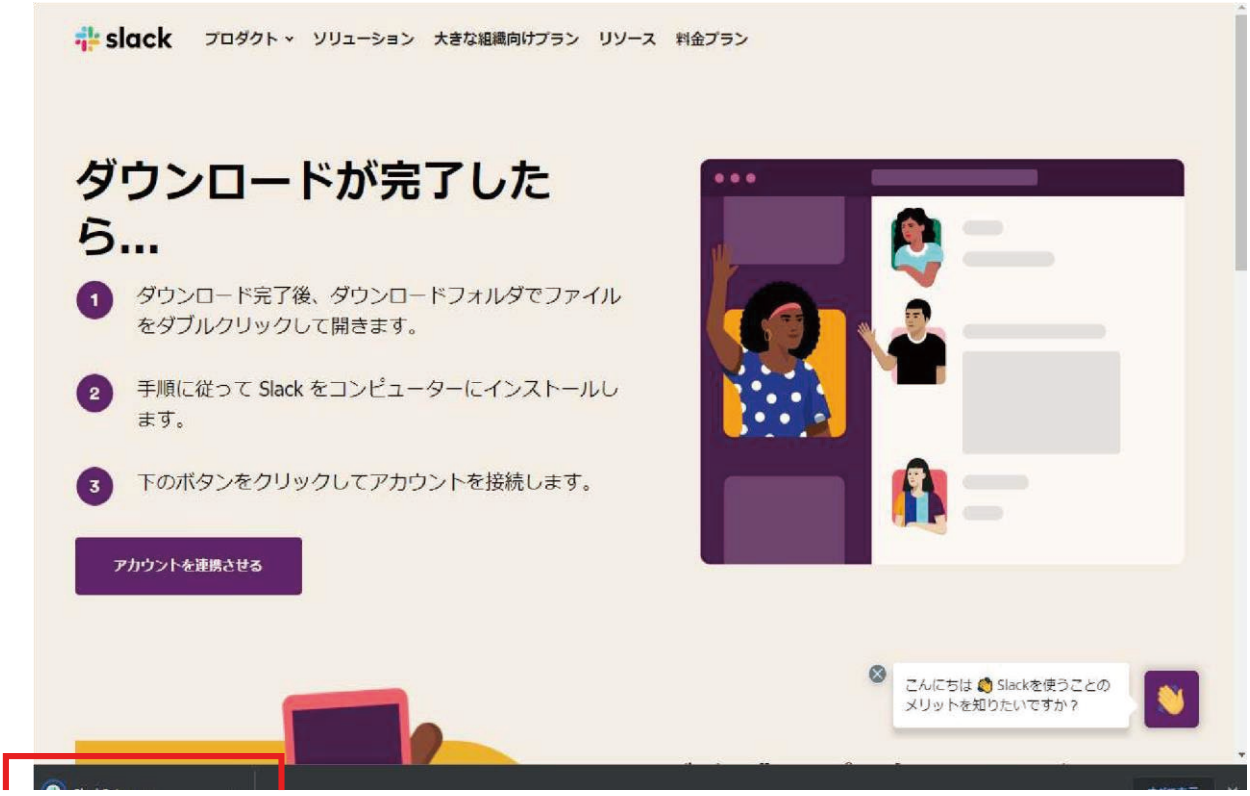
Slack の導入手順

1. 以下の URL にアクセスする

   https://slack.com/downloads/windows

2. Windows 版 Slack をダウンロードする

3. ダウンロードの完了を確認（以下の画面の赤枠のようになっていることを確認）



.

4. 上記 3. の赤枠にある SlackSetup.exe をクリックして実行する

5. アプリが立ち上がり、以下の画面が表示されることを確認する

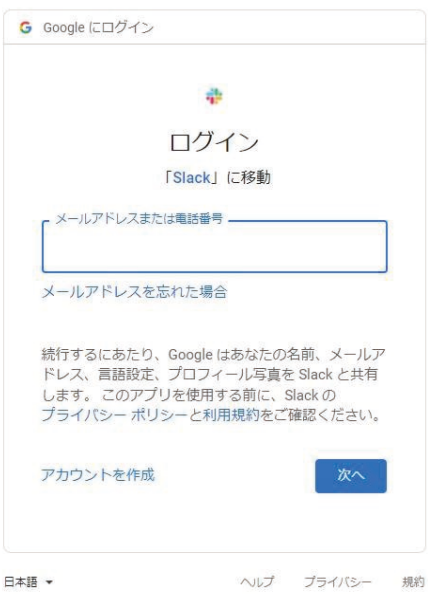
Slack への参加手順

1. 以下の URL にアクセスする

   https://join.slack.com/t/monkarmtprgtoyamateam/shared_invite/zt-1l8c7c8cj-r~Y9kXRmQalLIa3~8eF1wA

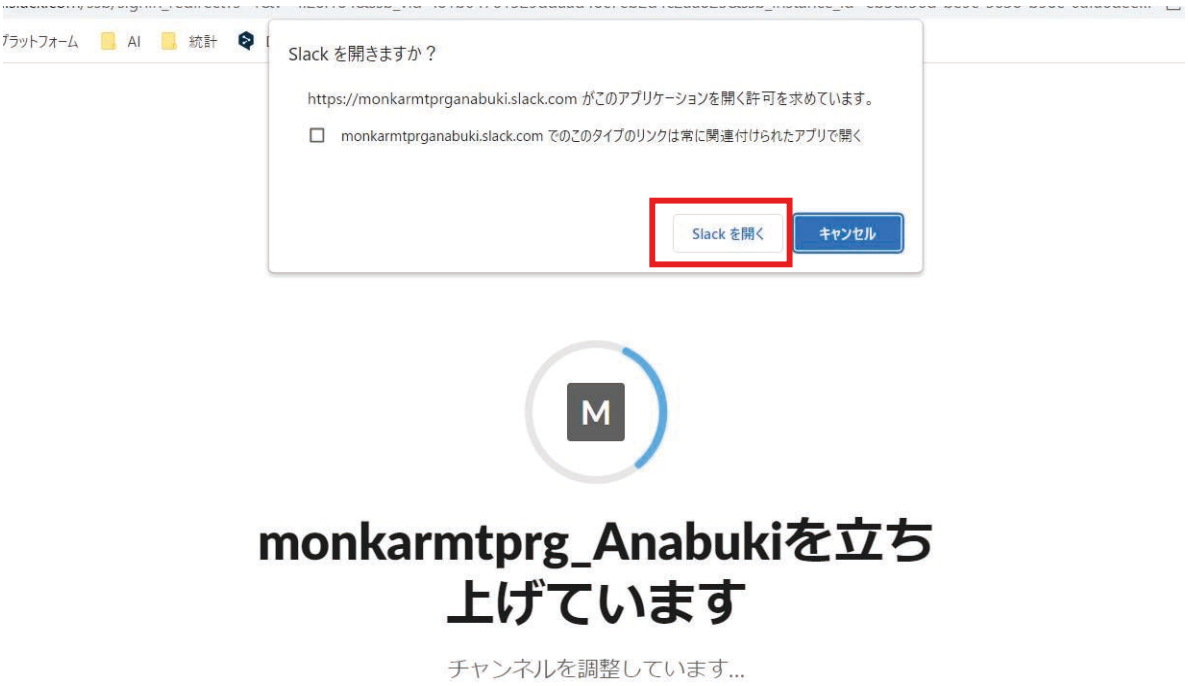
2. 以下の画面が表示されることを確認し、赤枠内の「Google で続行する」をクリック

3. 以下のような、Google の認証画面が表示された場合は配布されているメールアドレスとパスワードを入力表示されなかった場合（ログイン済み）は、次の 4. を進める



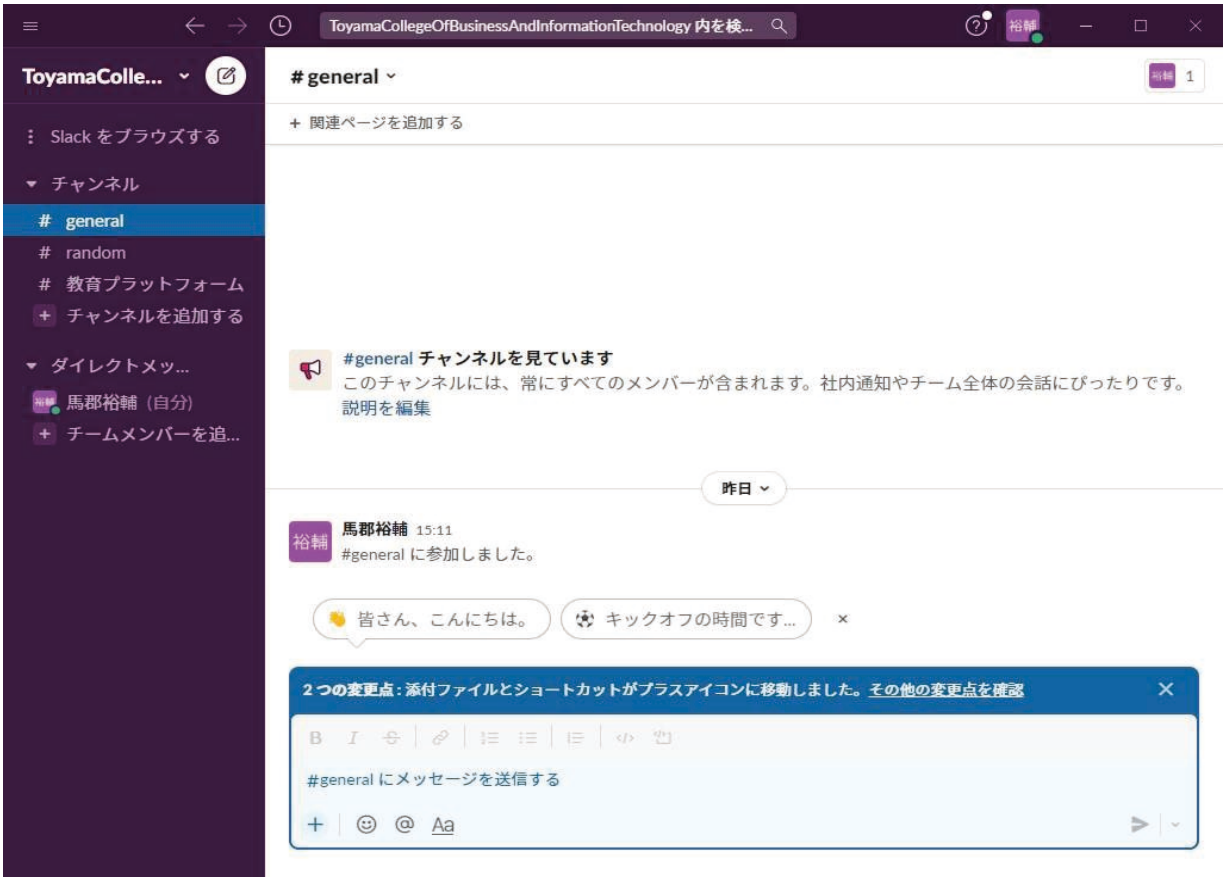
.

4. 以下のポップアップが表示された場合、赤枠の「Slack を開く」をクリック

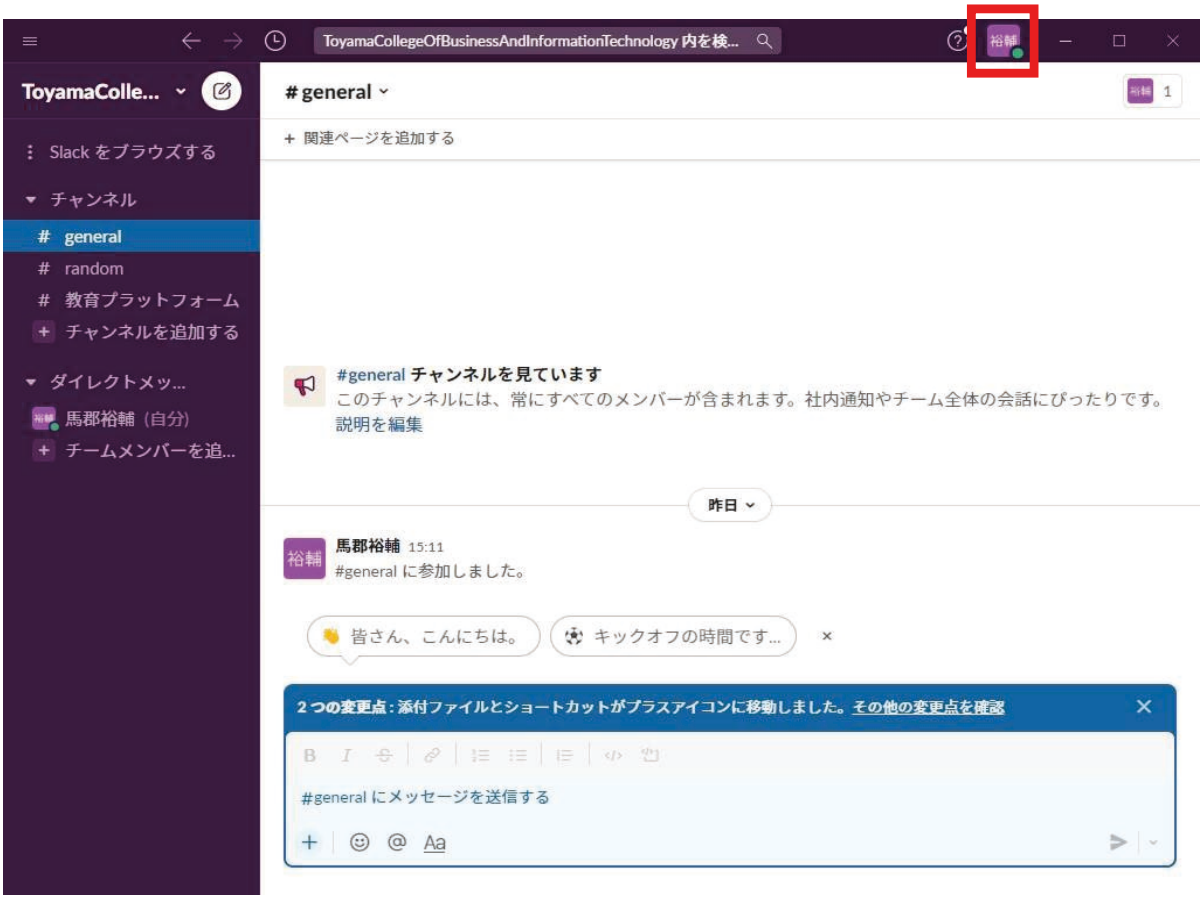


.

5. Slack アプリに以下のような画面が表示されることを確認

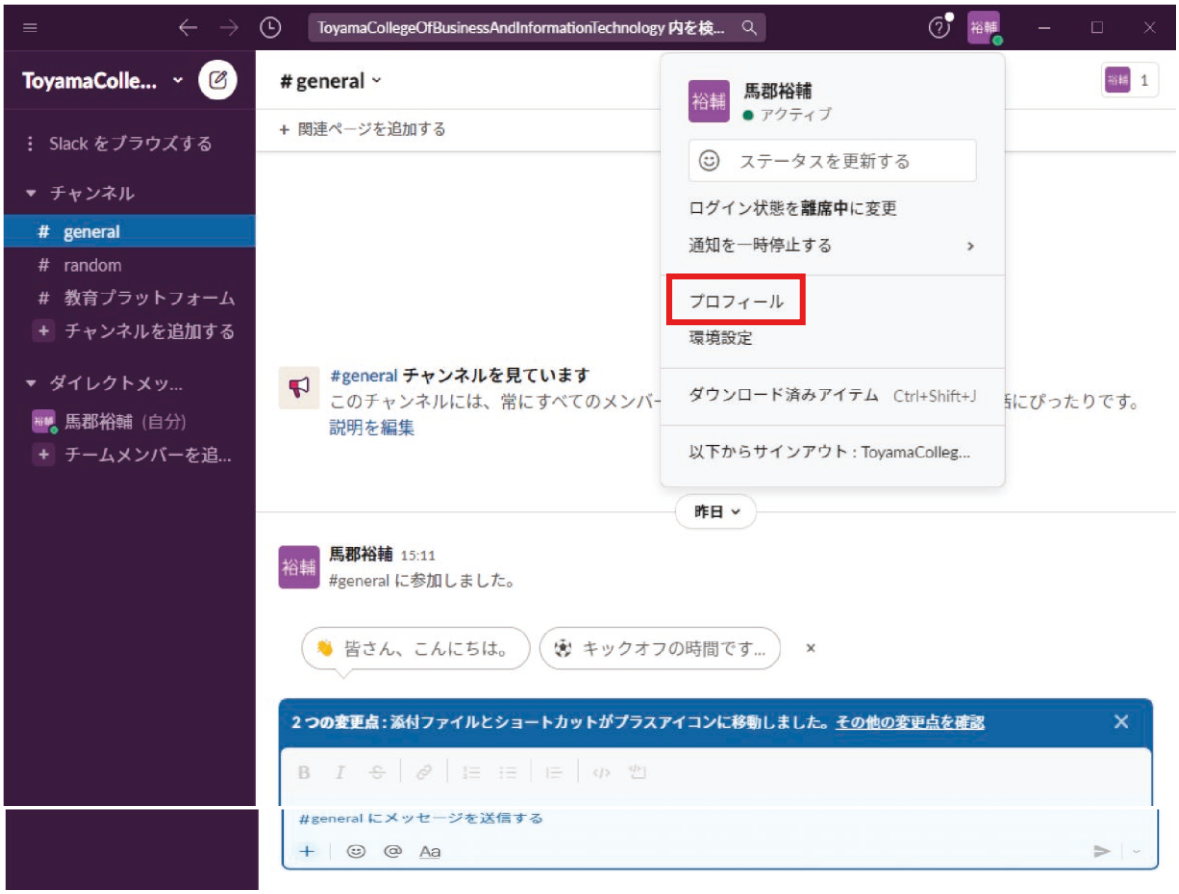


6. Slack アプリの画面から、以下の赤枠部分をクリック



.

7. 表示された項目から赤枠部分を「プロフィー



8. 画面の右側にプロフィール欄にある、赤枠内の「プロフィールを編集」をクリック

.

9. 以下のような「プロフィールを編集」画面が表示されることを確認

.

10. 赤枠内の「指名」に、任意の名前を入力（こちらに入力した内容が Slack のアカウント名として表示されます）赤枠内の「役職・担当」に、登録した github アカウント名を入力し、「変更を保存」ボタンをクリック

.

課題の開始手順について

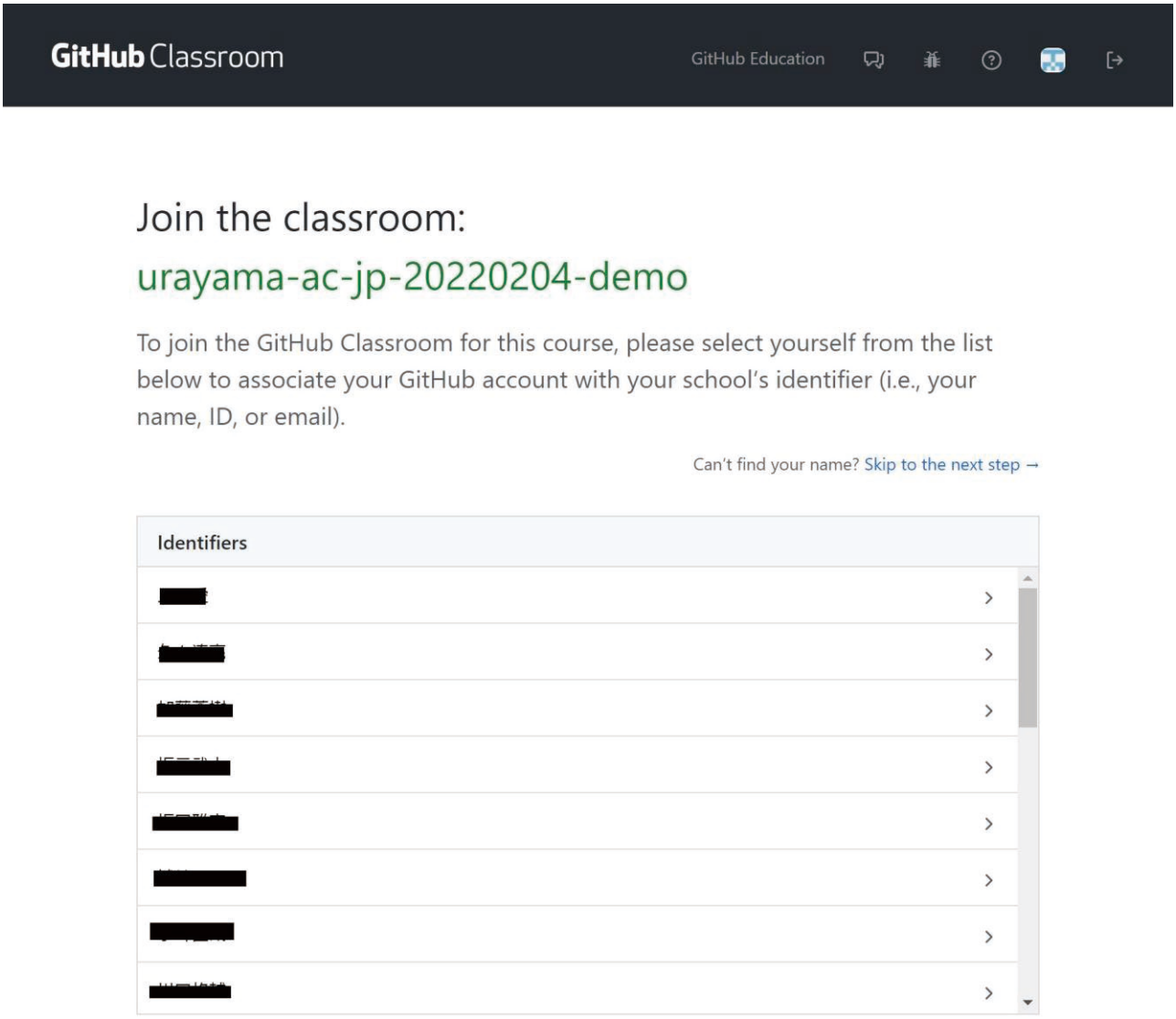
- 課題を始める際は、以下の操作を行ってください
- **なお、 2.と 3.の画面は初回のみ表示されます**

手順

1. Slack の課題チャンネルから赤枠の課題 URL をクリック

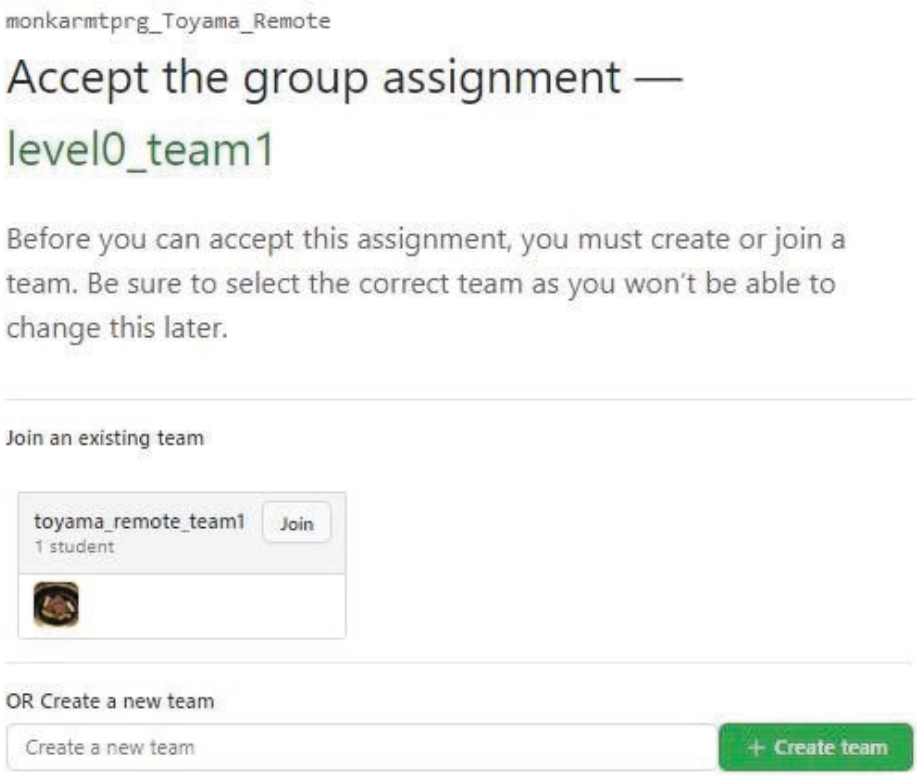
.

2 以下の画面で自分の名前を選択その際、ポップアップが表示された場合は OK を押下する
(この画面は初回の一度のみ表示)

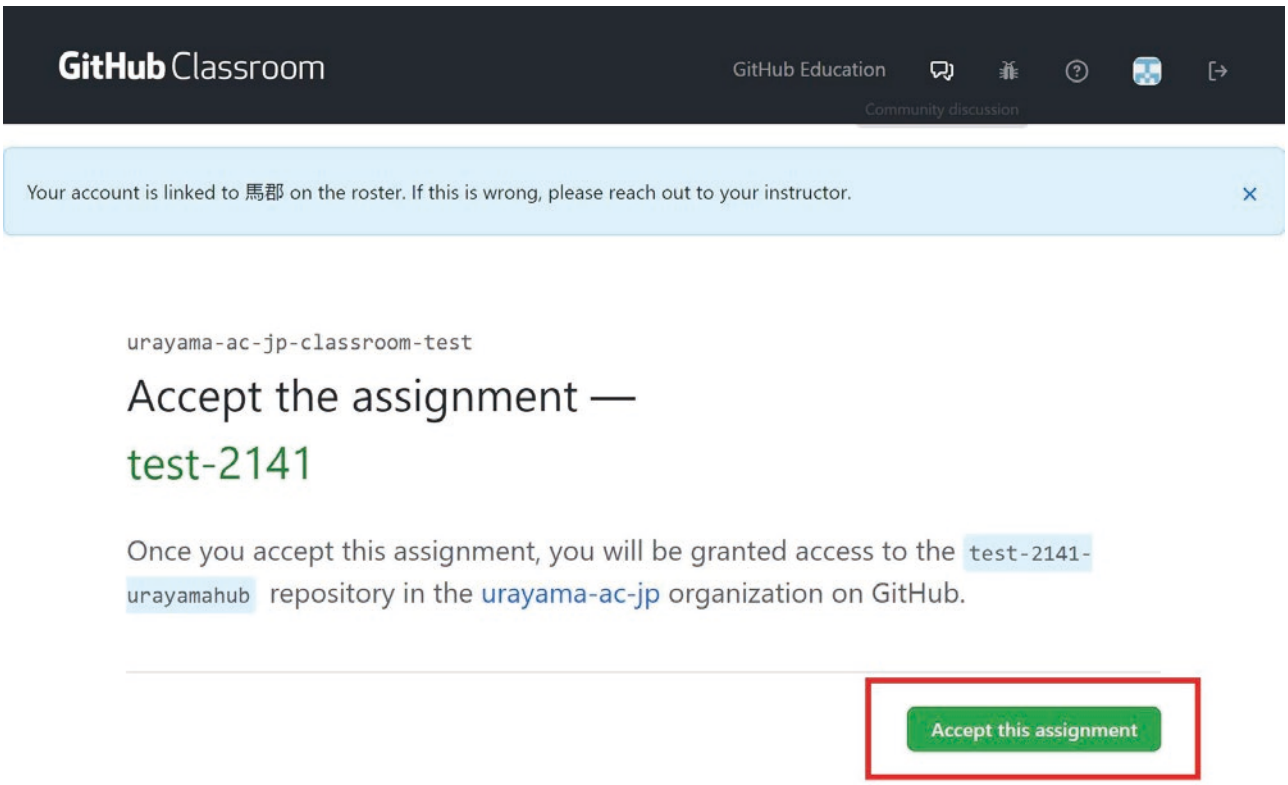


.

3.以下の画面で自分のチームを選択 (この画面は初回の一度のみ表示)

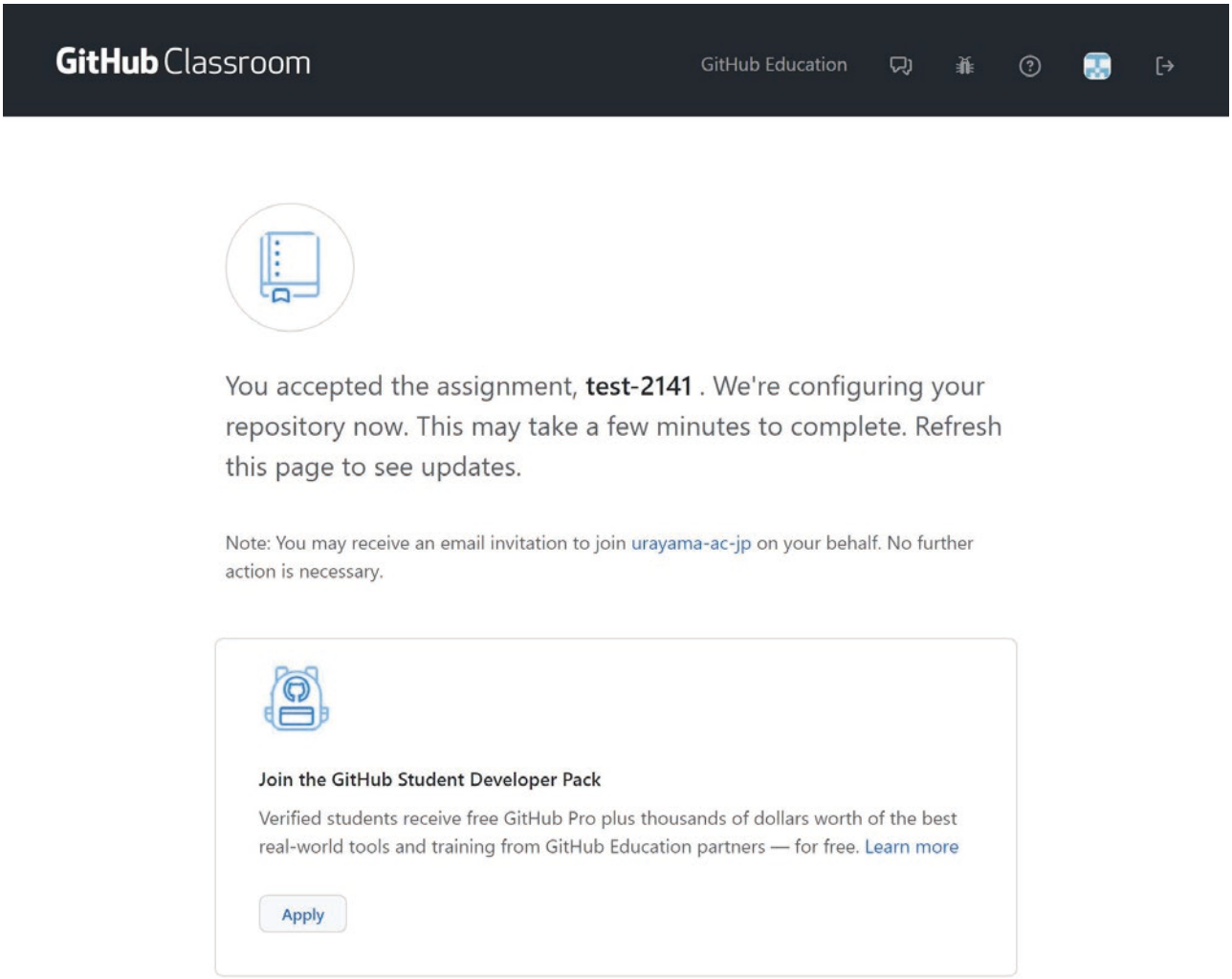


4.赤枠にあるボタン[Accept this assigment]を押下



.

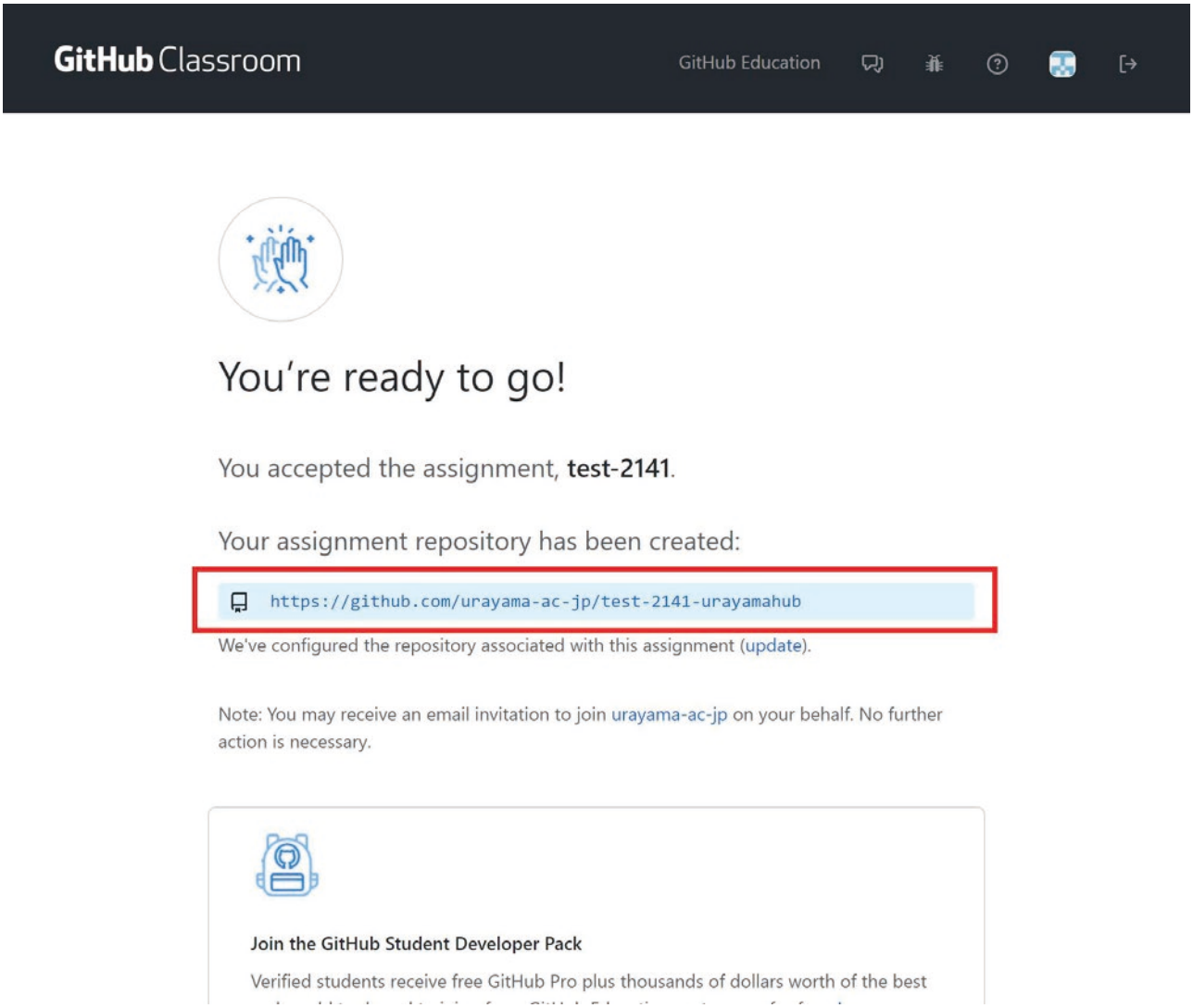
5.以下の画面が表示されたら画面の再読み込み（F5 キーを押下）を行う



.

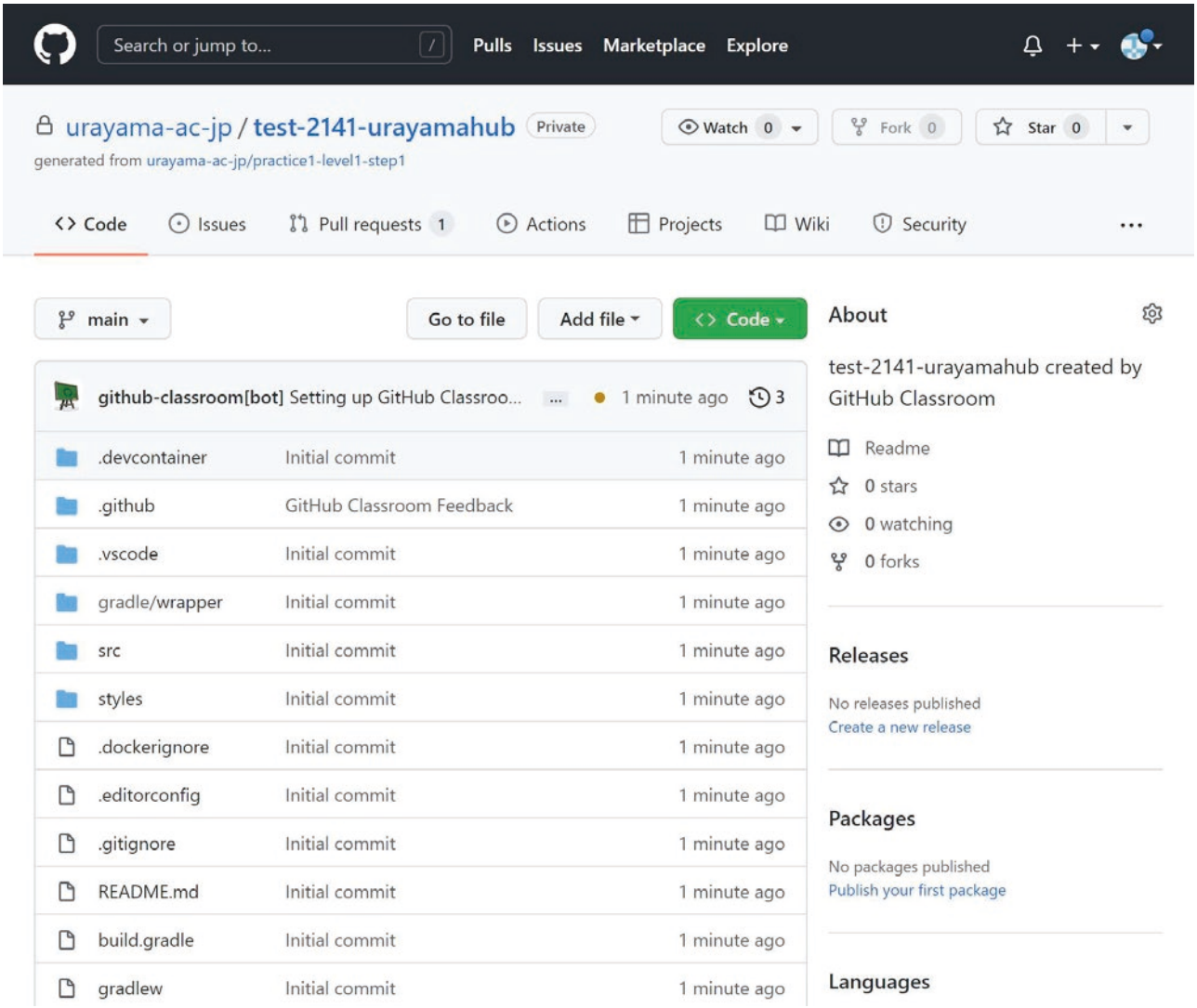
6.以下の画面が表示されるので、赤枠にあるリンクを押下

※表示されるリンク名は異なります



.

7.以下の GitHub の画面が表示されることを確認画面をスクロールして課題内容を確認してください



課題の提出手順

課題の提出には以下の作業が必要になります

Commit(コミット)

- Commit は、追加・変更したファイルを Git に登録するための操作です。

Push(プッシュ)

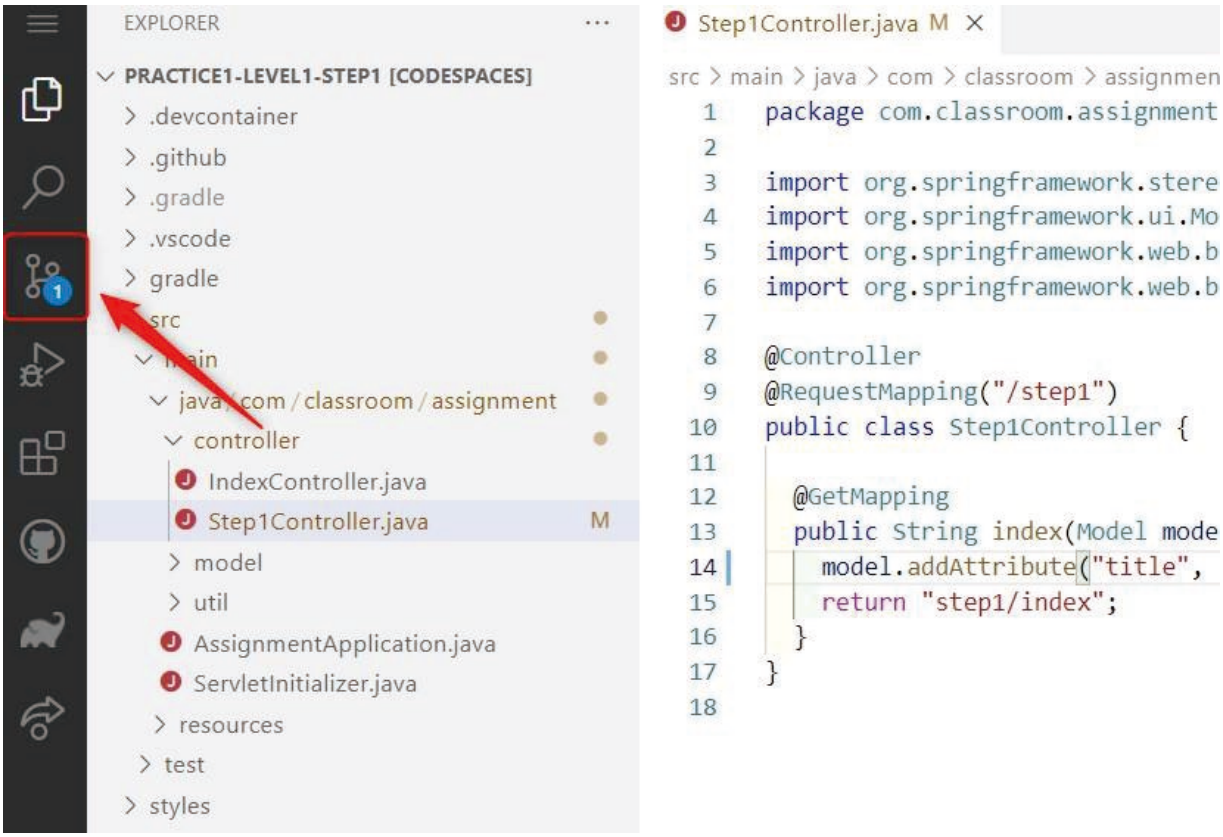
- push は、コミットした内容をローカルレポジトリからリモートリポジトリに送信する操作です。

Pull Request(プルリクエスト)

- Pull Request は、開発者のローカルレポジトリでの変更を他の開発者に通知する機能です。

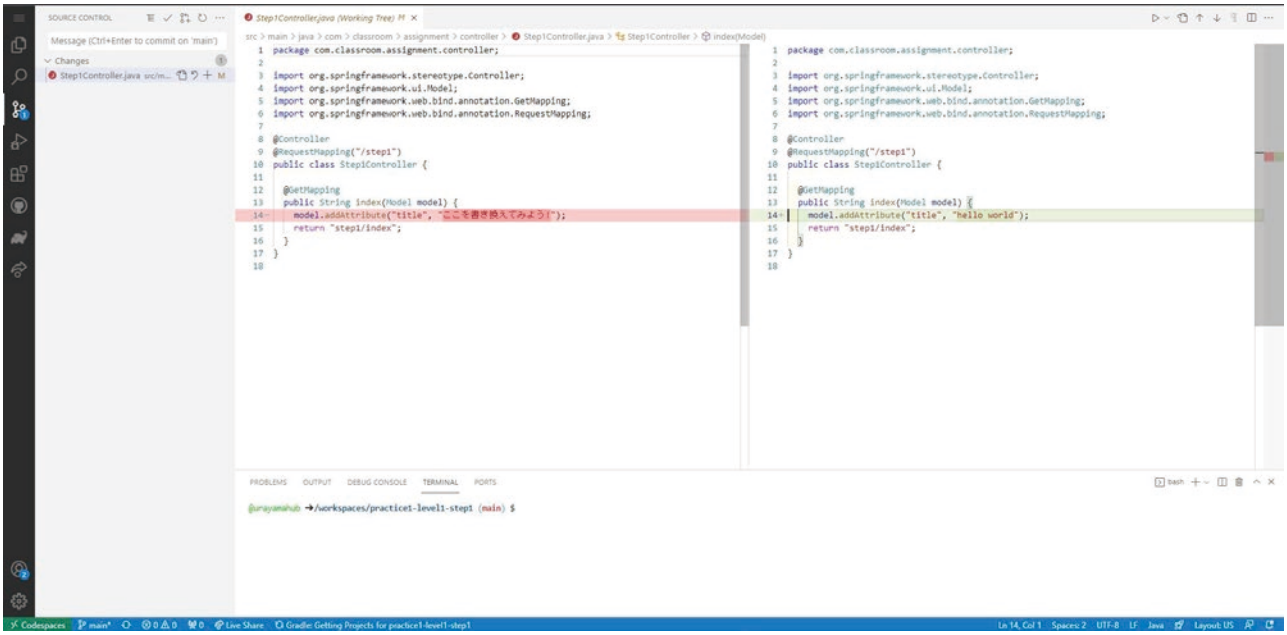
作業手順

1. 木の枝のようなマークを押します。

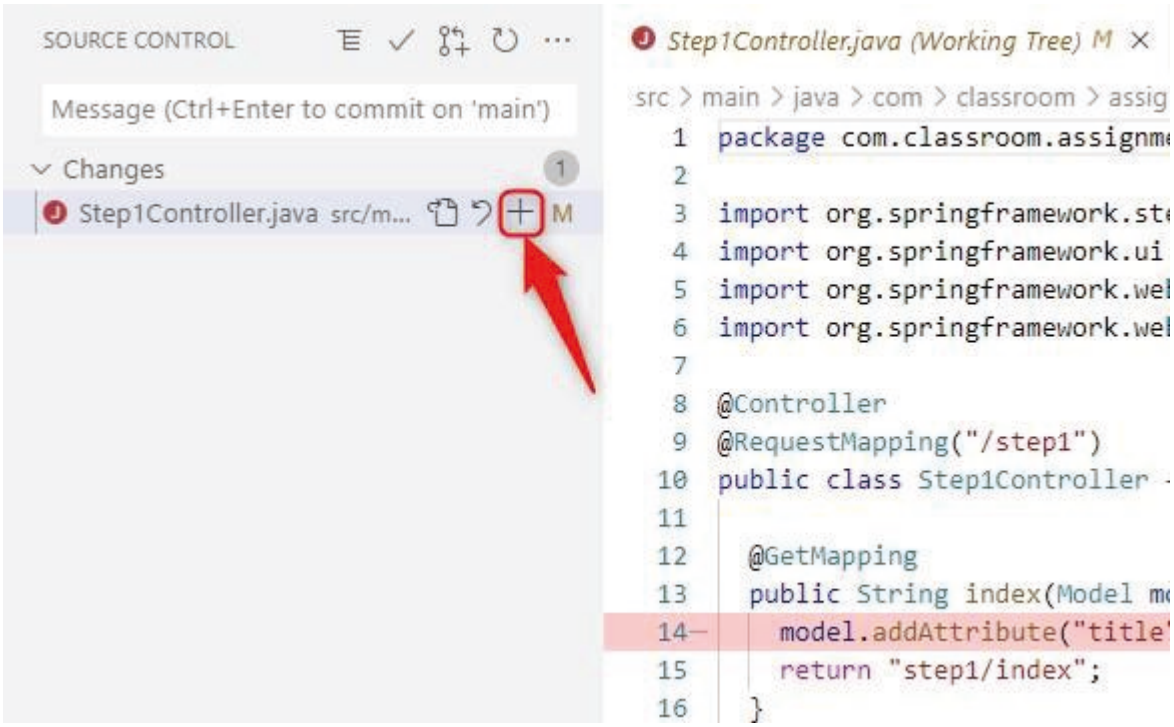


.

2. これまでに行ったファイルへの変更ないようが表示されています。 変更内容に問題がないか今一度確認しましょう。

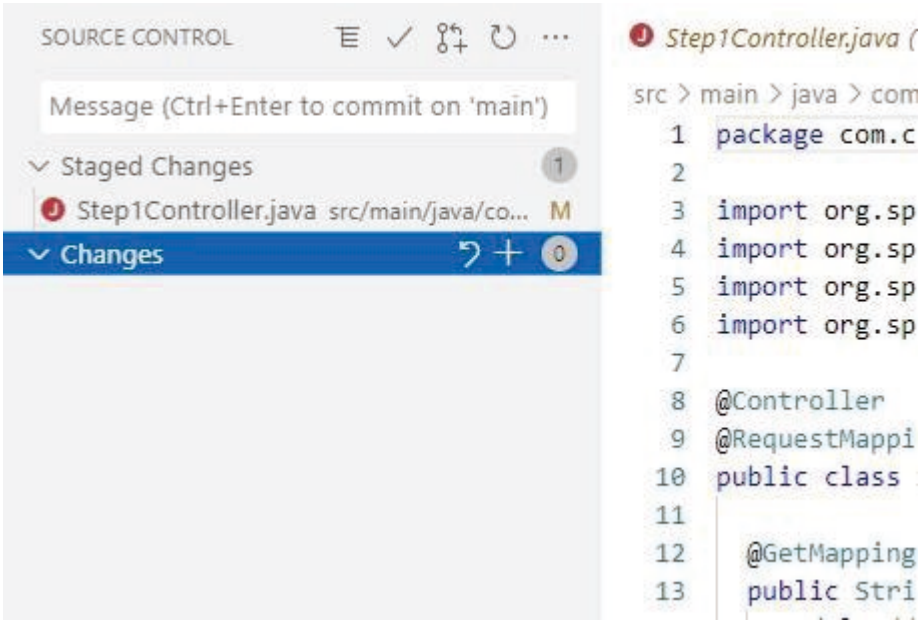


3. コミットするファイルを選択しましょう。 コミットしたいファイルの横の＋ボタンを押しましょう。



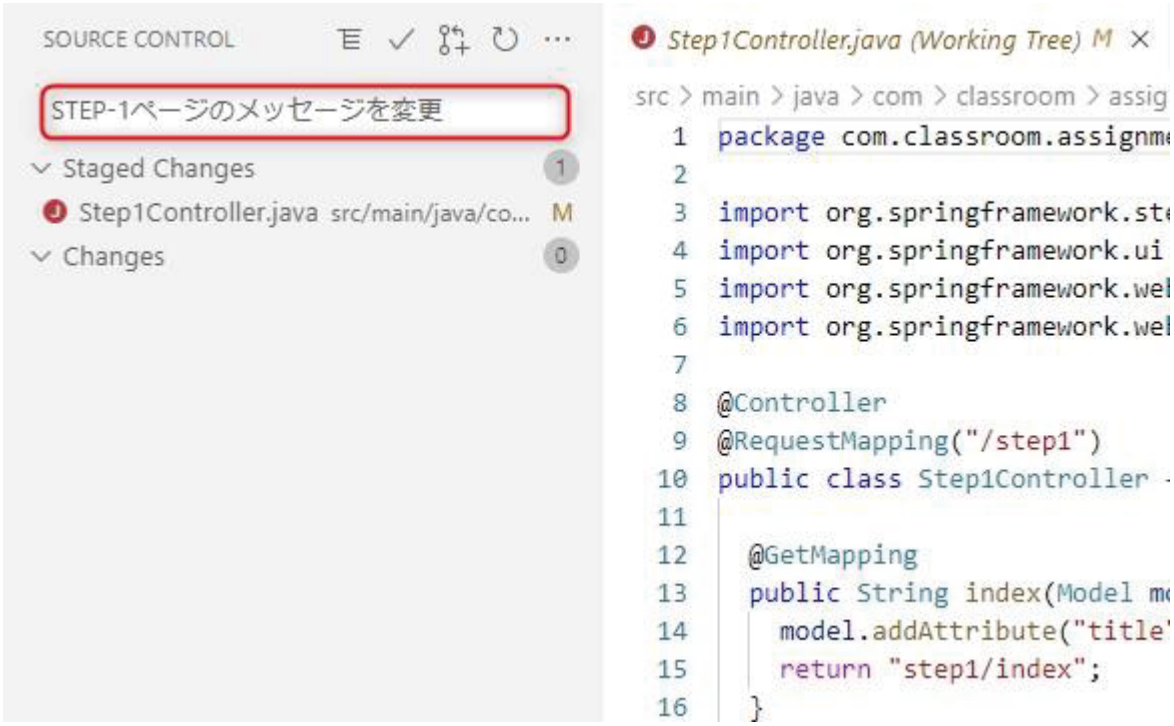
.

4 Staged Changes にコミットしたいファイルが含まれていることを確認しましょう。



5. Message 欄に Commit Message(コミットメッセージ)を入力しましょう。

-
-

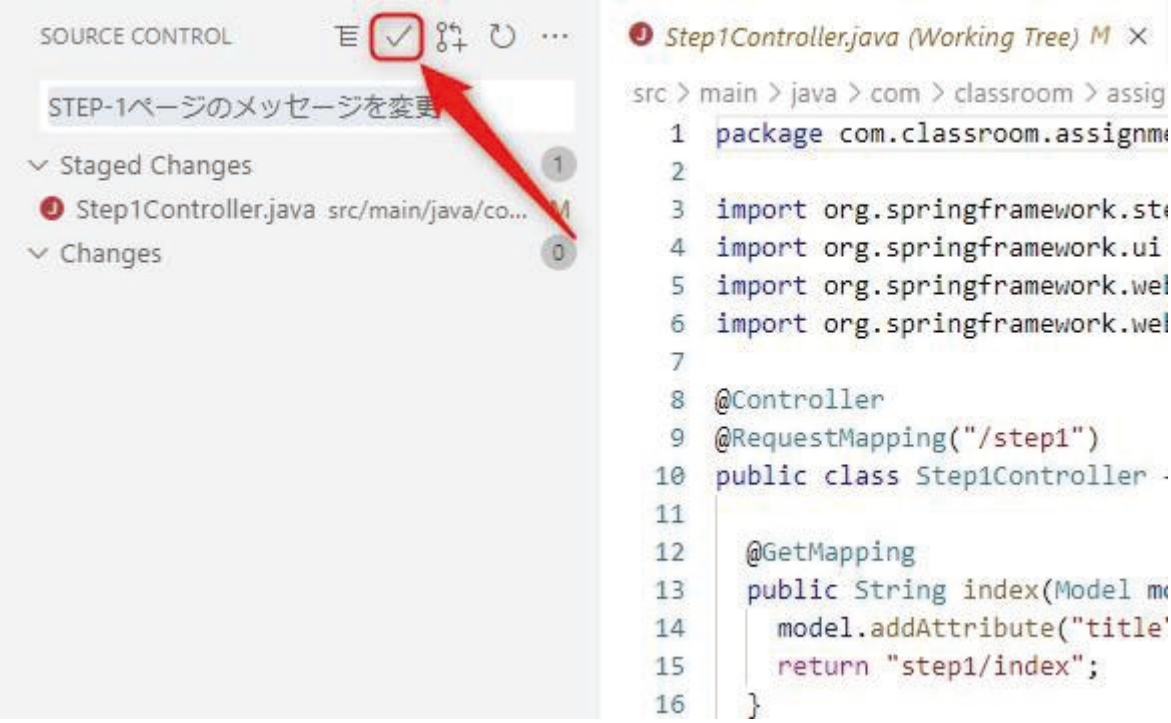


Commit Message には、今回のコミットの追加・変更内容について記載します。

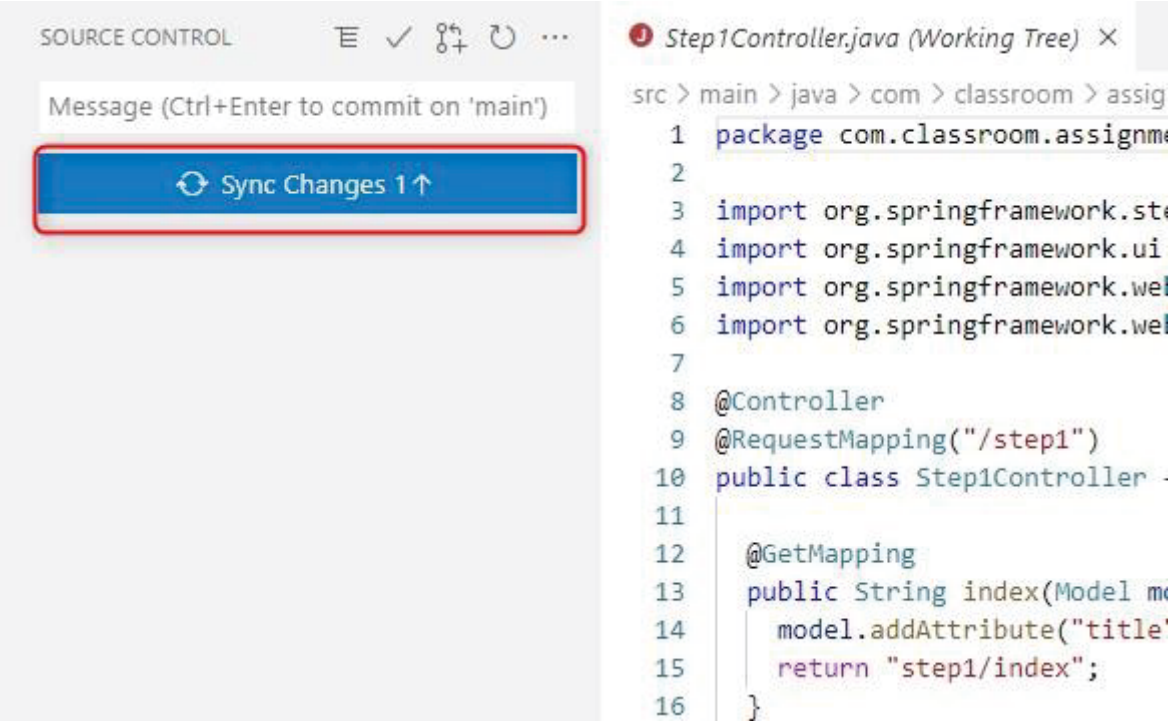
課題のテーマを記載しましょう

.

6 チェックボタンを押してコミットしましょう。

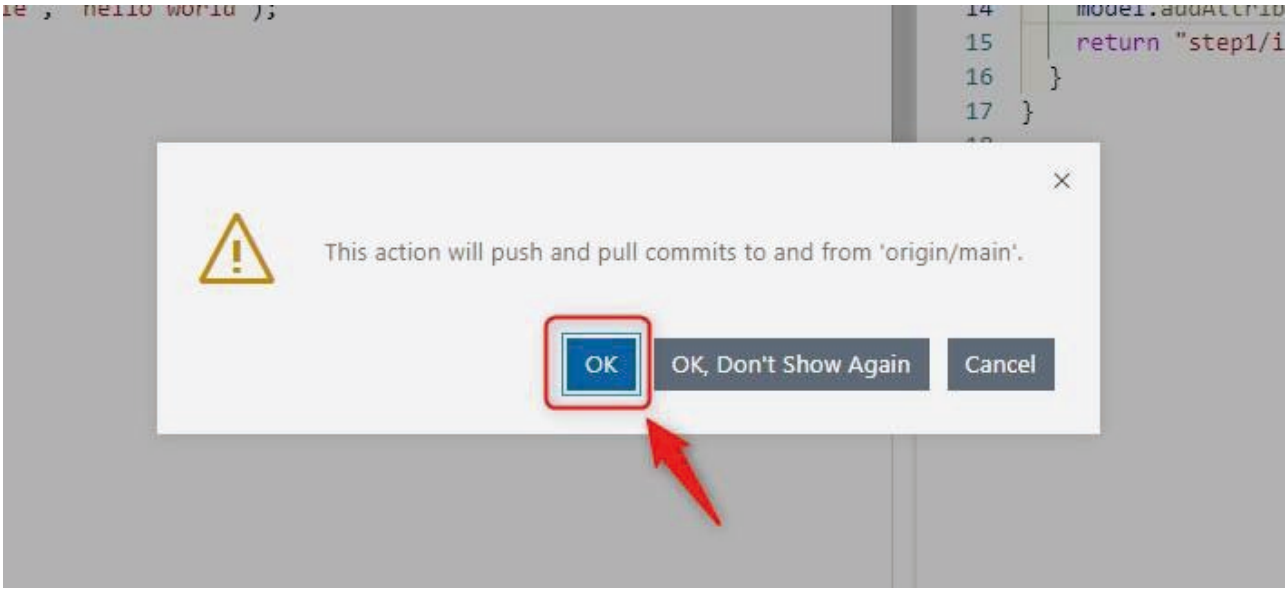


7. Sync Changes ボタンを押してコミットをプッシュしましょう。

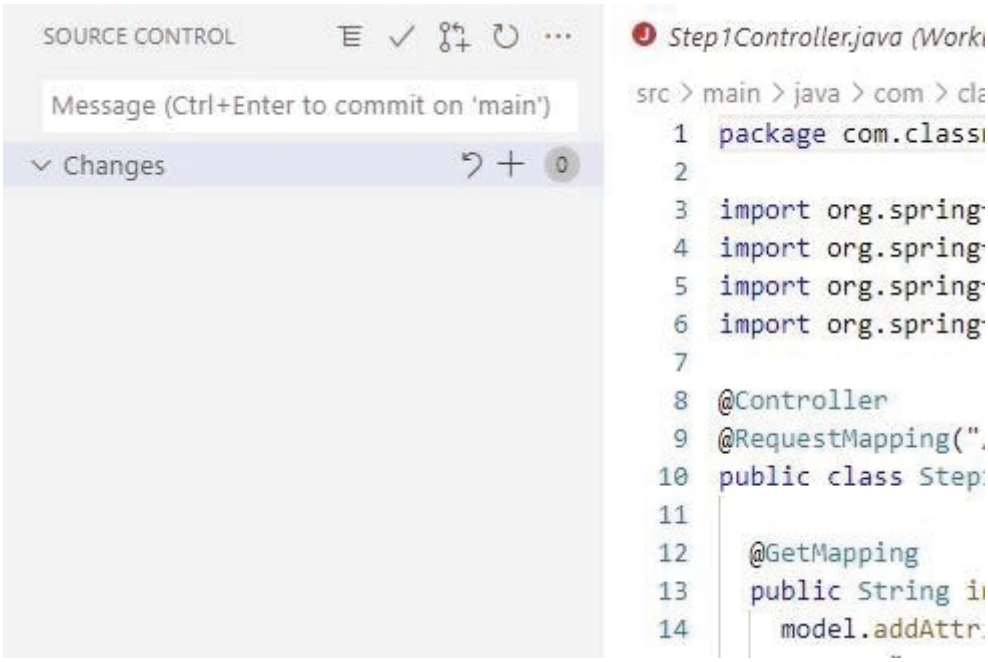


.

8 ダイアログが出ますので「OK」を押しましょう。

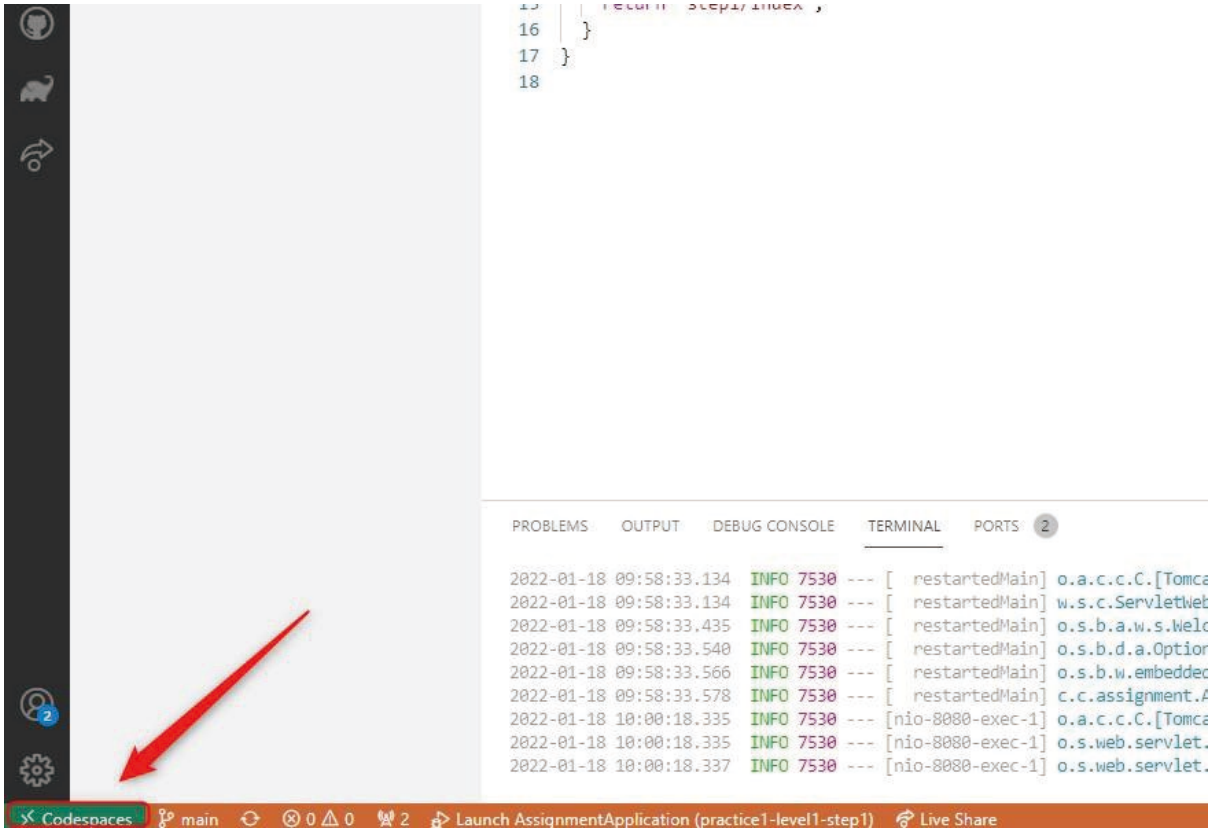


9. Sync Changes ボタンが消えていることを確認してください。

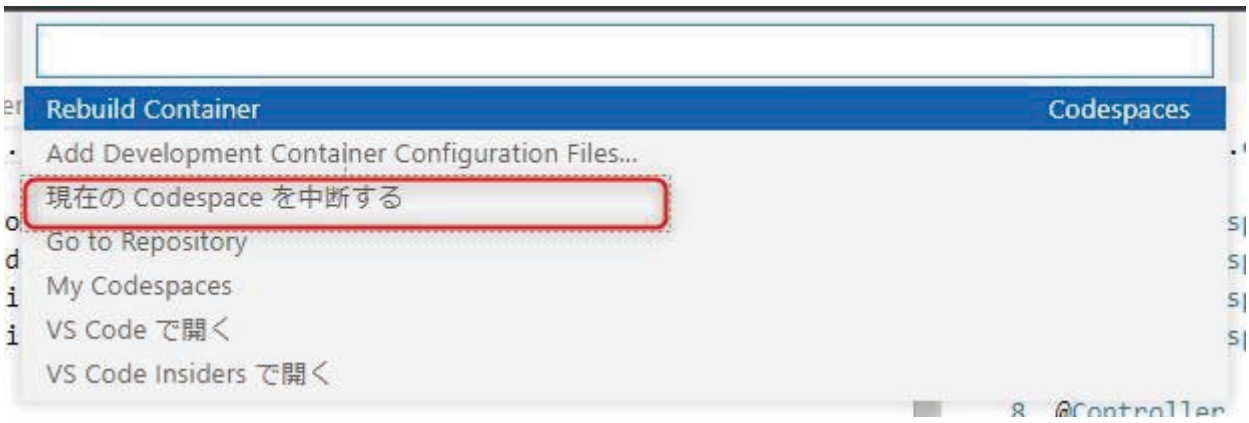


.

10 画面左下の緑のボタンを押しましょう。

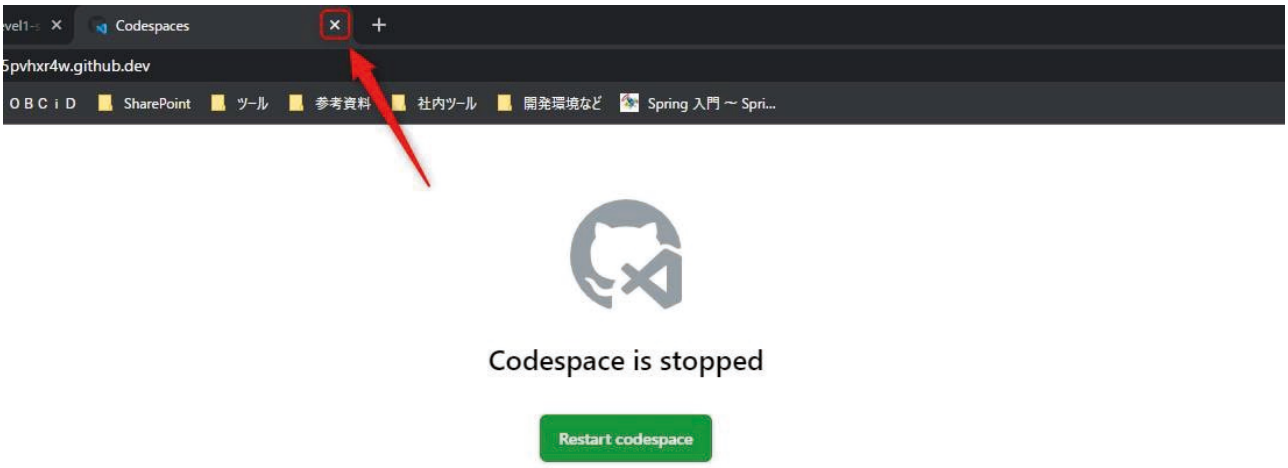


11. 画面中央上部の「現在の Codespace を中断する」を押します。

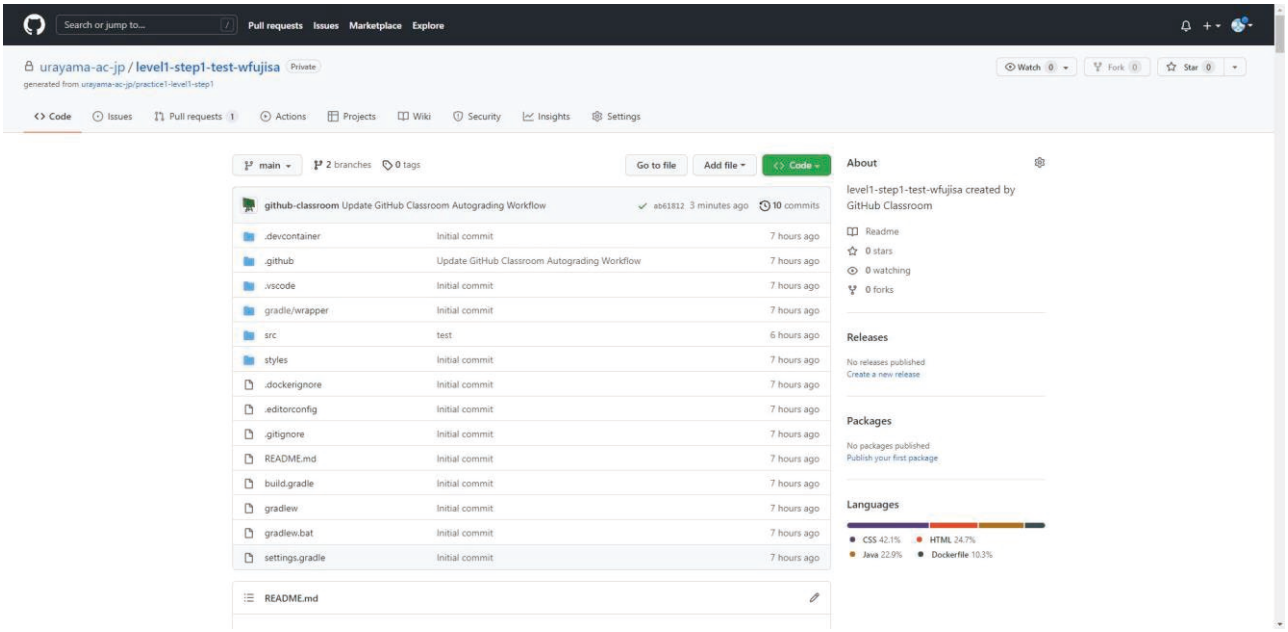


.

12 タブを×ボタンで閉じましょう。

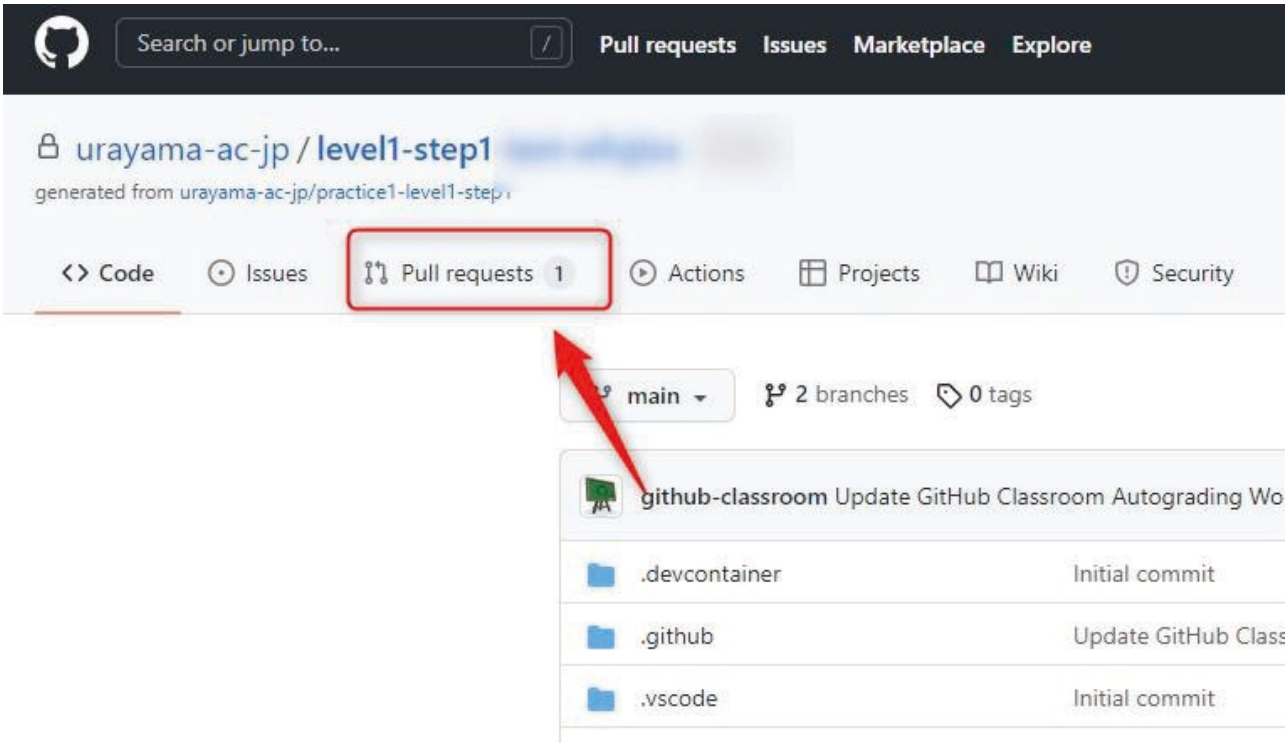


13. 画面コピーの画面に戻ります。

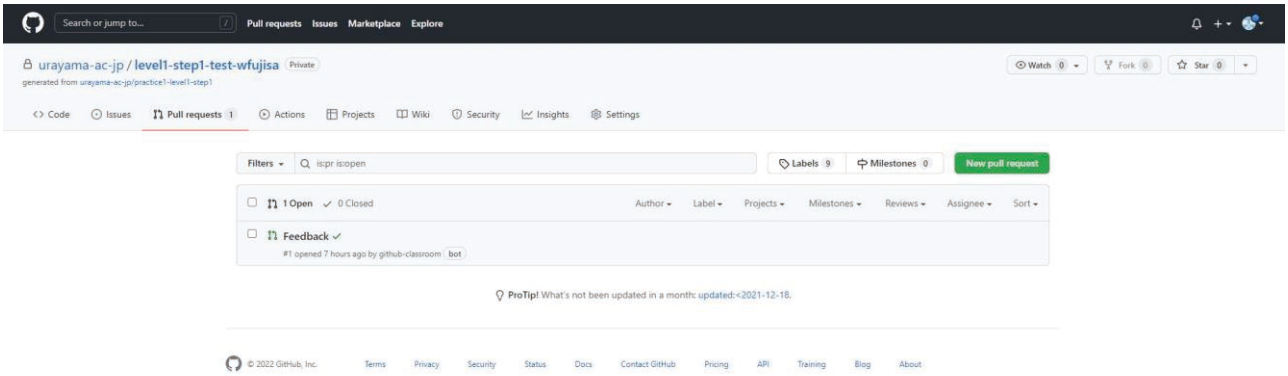


.

14 「Pull requests」ボタンを押します。



15. 講師からのフィードバックがある場合は、この画面に表示されます。



.

チーム開発方法

Live share を使ったチーム開発概要

Live Share では複数人でソースを共有し、同時に編集することができます。

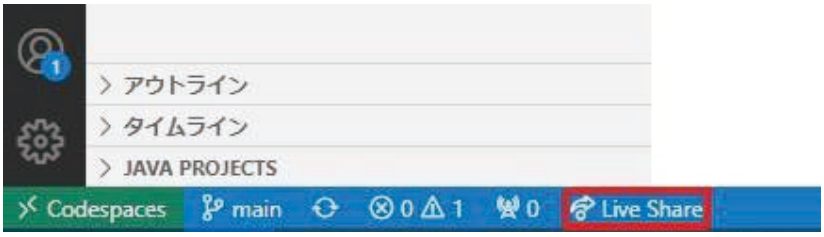
ホスト側（開発環境を立ち上げ、招待する側）とゲスト側（参加する側）に分かれ、ホストは自分の環境にゲストを招待します。

招待手順

1. ホスト：Live Share 起動
2. ホスト: url 共有
3. ゲスト: session に参加
4. ゲスト: GitHub ログイン
5. ホスト・ゲスト: Live Share 起動確認

ホスト：Live Share 起動

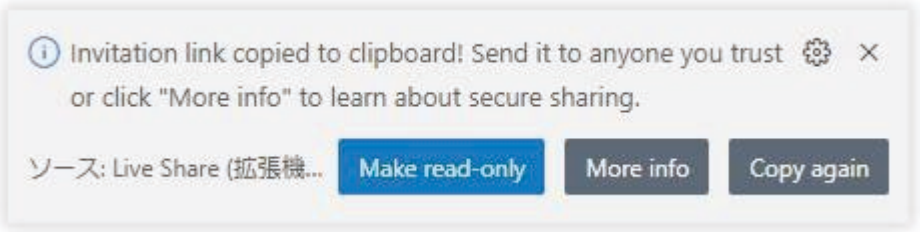
1.ホスト（招待する人）の pc で Codespace 起動後、左下の `Live Share` をクリックします。



.

2.右下に起動中のアナウンス `Starting collaboration session` が表示されています。

3.右下に `Invitaion link copied to clipboard!` ～が表示されると準備完了です。



ホスト: url 共有

1 の手順終了後、すでに pc には招待用URL がコピーされています。

slack などに貼り付け、メンバーに URL を共有してください。

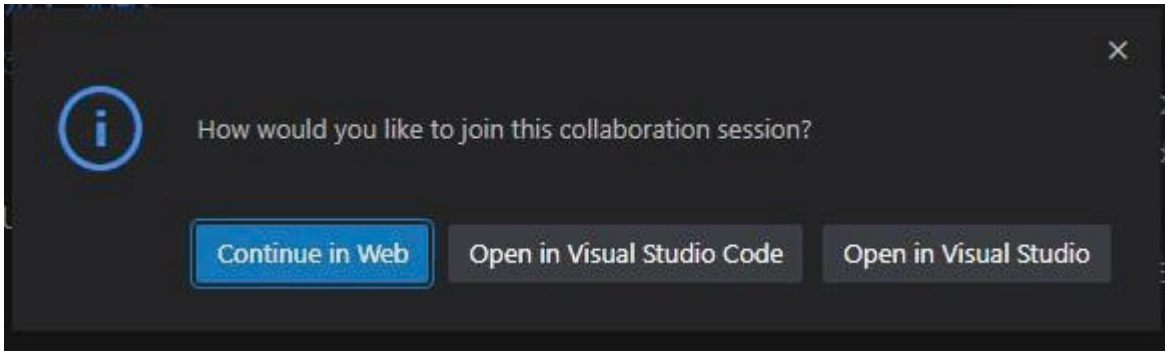
（コピーに失敗した場合は `Copy again` をクリックしてください。）

.

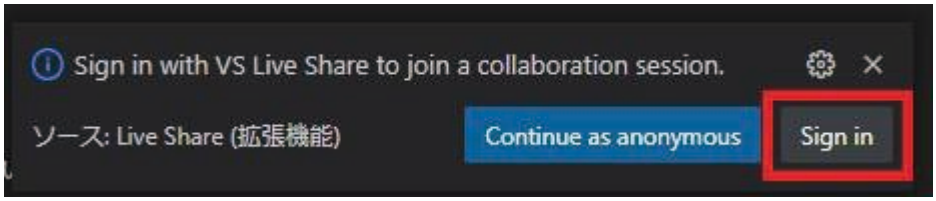
ゲスト: session に参加

ゲスト（参加メンバー）は共有された URL をクリックすると、ブラウザで VS Code が立ち上がります。

1.次のようなポップアップが表示された場合、`Continue in Web` を選択してください。



2.右下に、`Sign in with VS Live Share to a collaboration session.` が表示されるので、`Sign in` をクリックしてください。



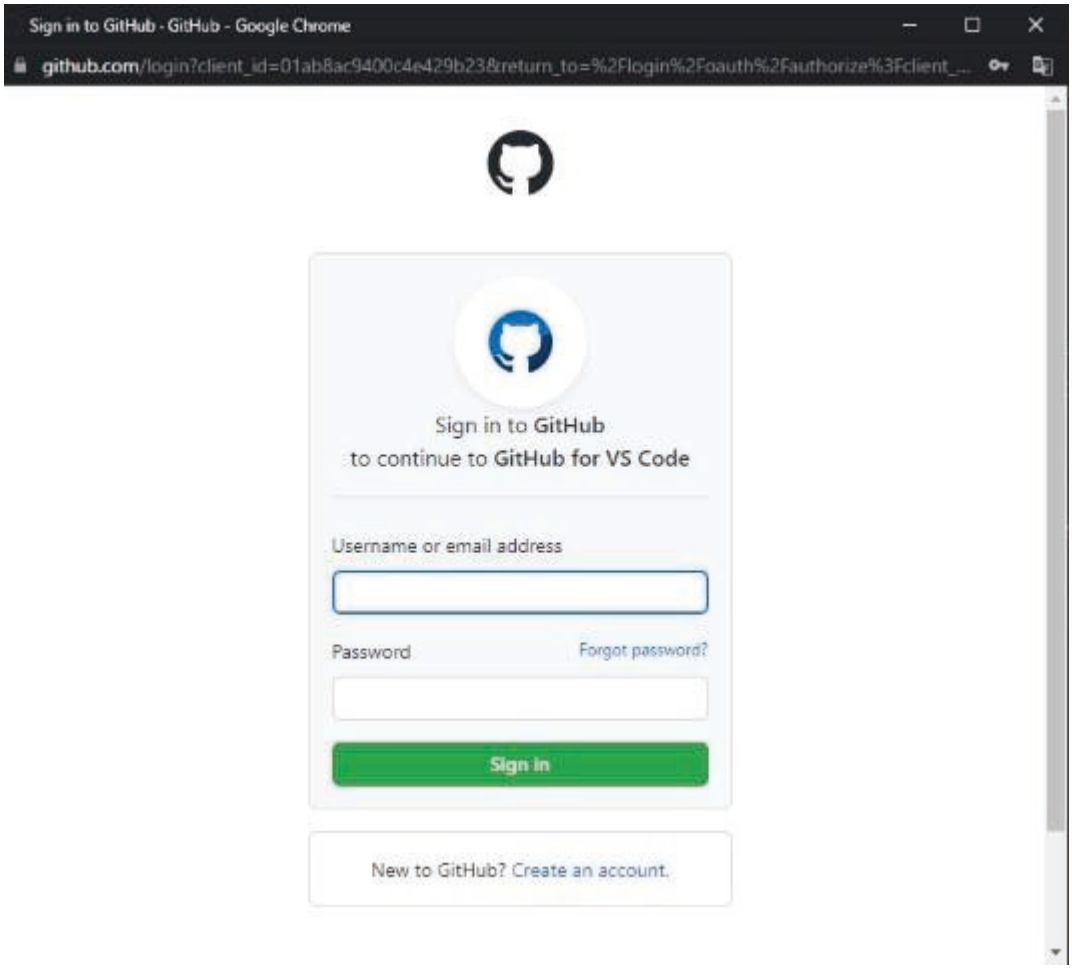
.

3.上部にサインインするサービスの選択が現れるので、GitHub を選択します。

ゲスト: GitHub ログイン

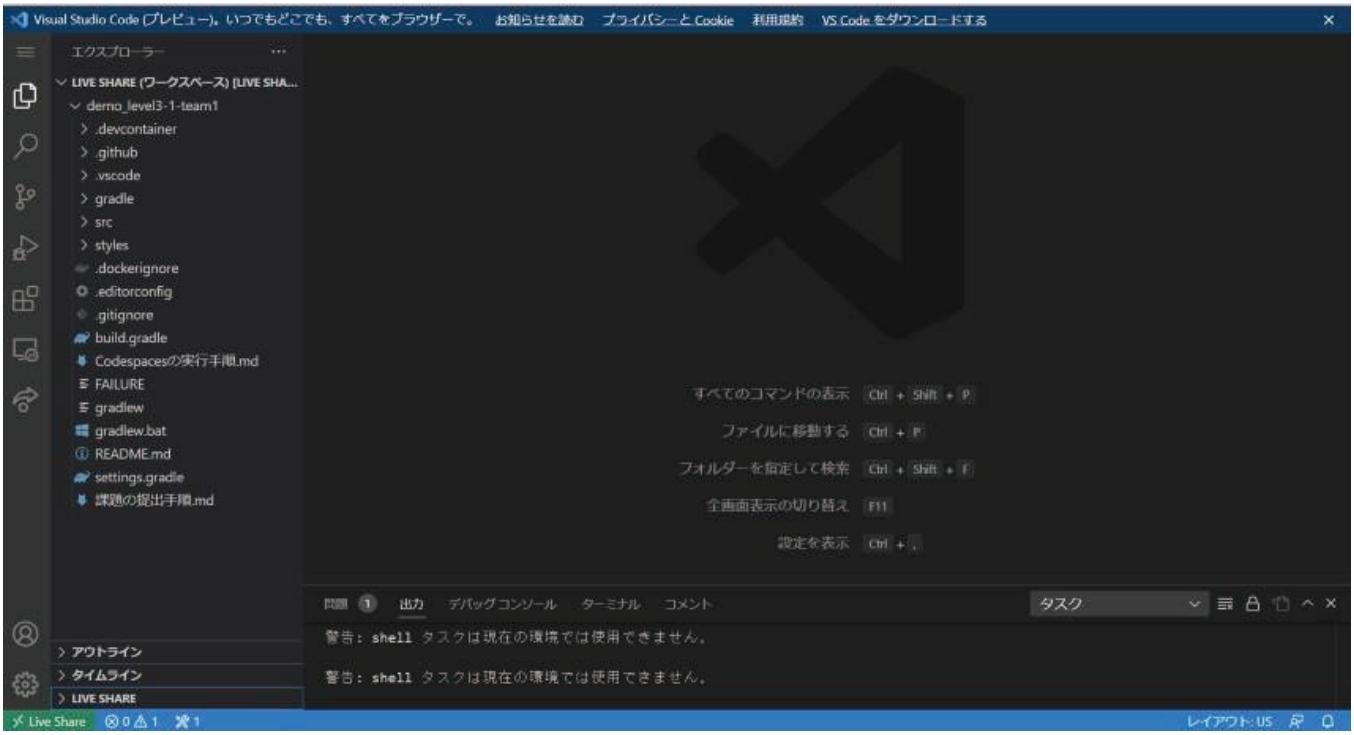
GitHub ログイン用のウィンドウが表示されます。

1.作成したアカウントを入力し、認証に進んでください。



.

2.ログイン後しばらく時間をおいた後にブラウザの VSCode の準備が完了したら終了です。



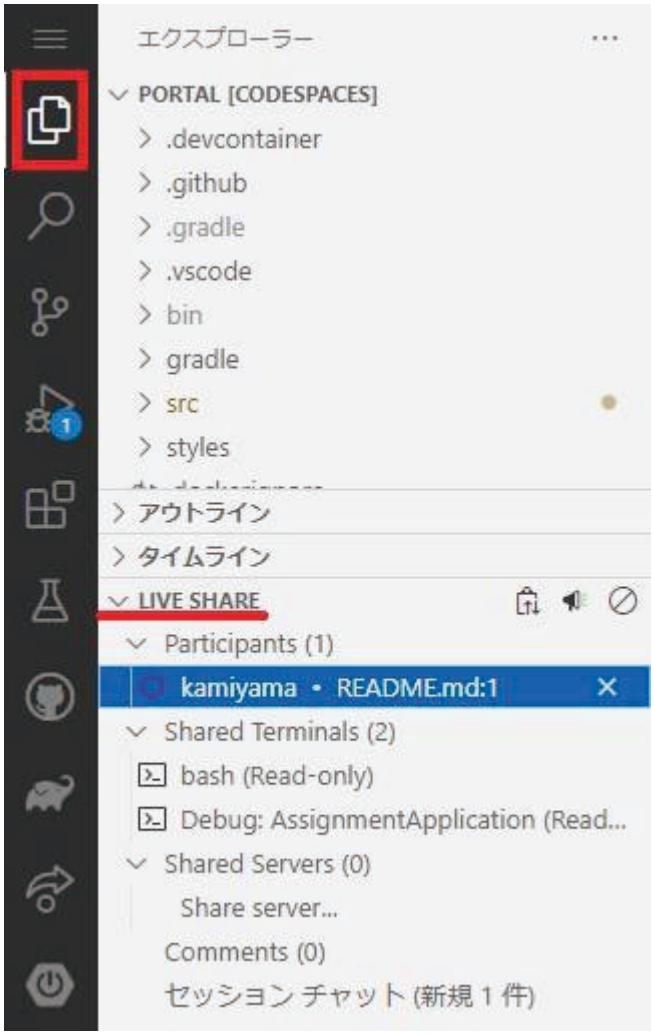
.

ホスト・ゲスト: Live Share 参加者確認

エクスプローラー（表示されていない場合は左上ファイルボタンをクリック）の下部に LIVE SHARE タブが現れます。

`Participants` に表示されるのが参加中のユーザーです。

全員参加しているか確認してみましょう。



.

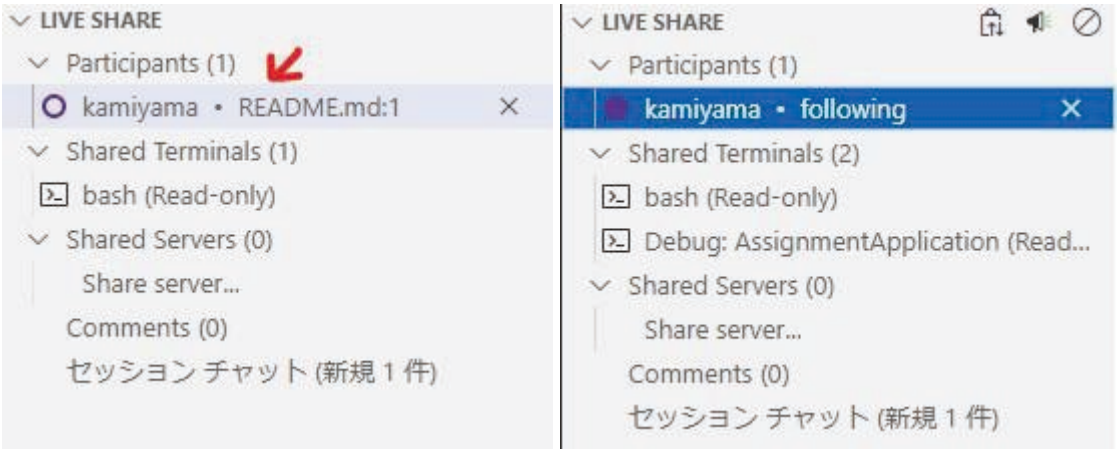
チーム開発 Tips フォロー機能□作業中のメンバーの動きを追いたい場合

（ホスト・ゲスト）

Live Share では、あるユーザーの開いているファイルや編集中や選択中の箇所を自動で追いかけることができます。

エクスプローラーの LIVE SHARE の `Participants` 内で、追いかけたいユーザー名をクリックし、名前左側の丸が塗りつぶされると追跡完了です。

解除したい場合は再度ユーザー名をクリックしてください。



実行したアプリケーションをメンバーで共有したい場合

実行したアプリケーションをメンバーの手元の pc で確認することができます。

1. ホストの人がアプリケーションを実行します。
2. エディター下部のポートを選択します。
3. ポート `8080` の上で右クリックし、ポートの表示範囲□`Public` を選択します。（画像参考）
4. 起動したアプリケーションの URL を slack などでメンバーに共有します。

□注意□アプリケーションを停止すると、共有設定も初期化されます。

お手数ですが、アプリケーション再起動の際に再度共有設定を Public に変更してください。

.

モブプロ実施方法補足コミット方法について

概要

今回のモブプロでは各ターム（各順）終了時にソースをコミット(Commit)します。

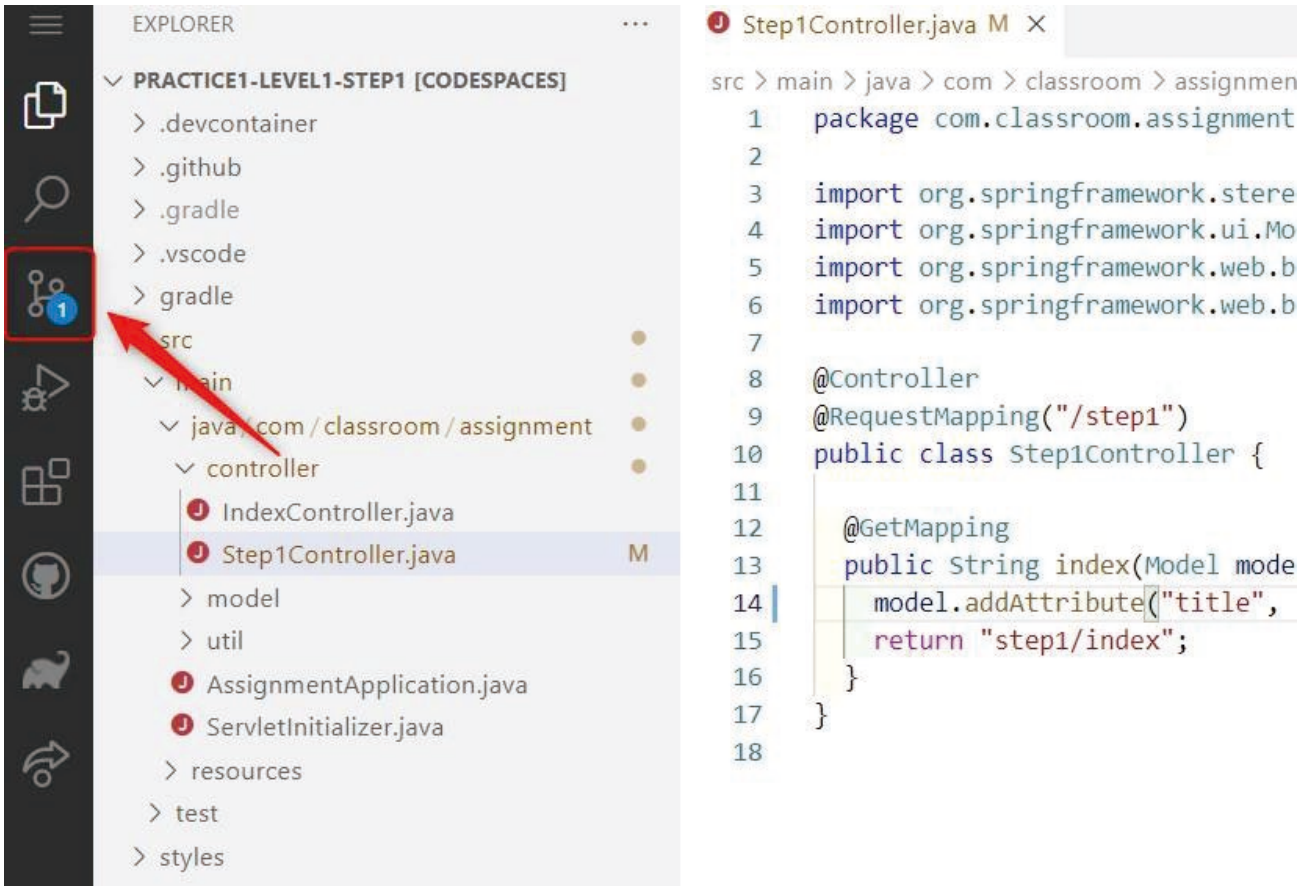
コミットは、追加・変更したファイルを Git に登録するための操作です。

コミットはホストの方が行ってください。

.

作業手順

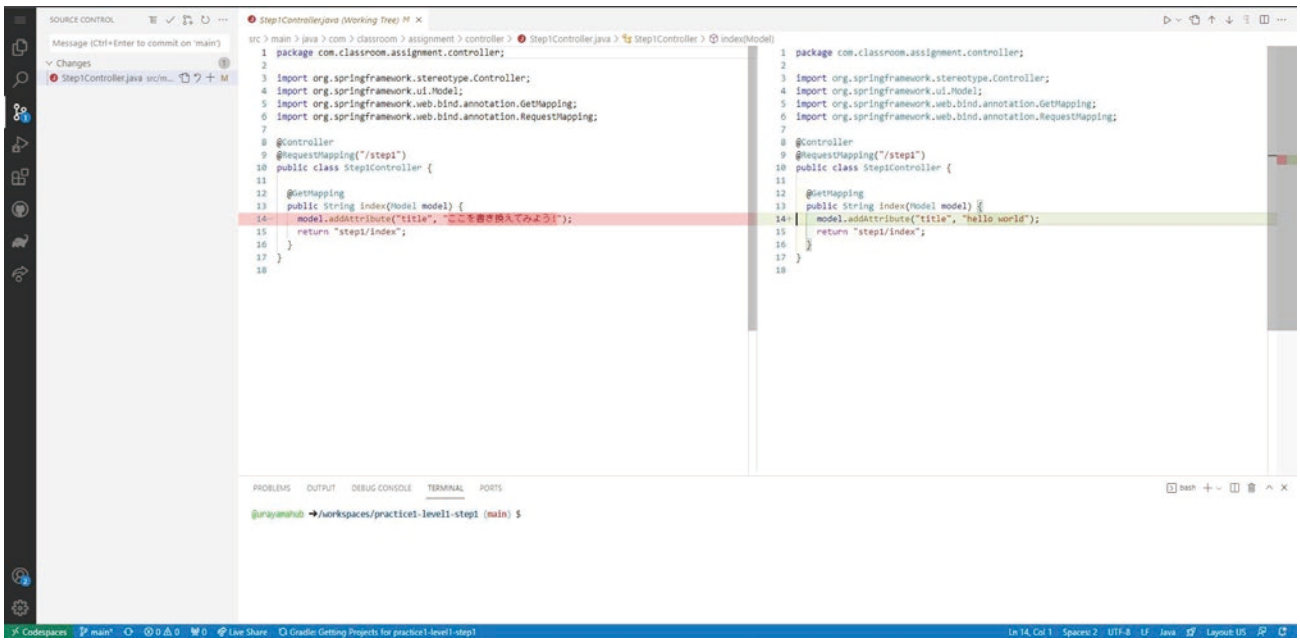
1. 木の枝のようなマークを押します。



.

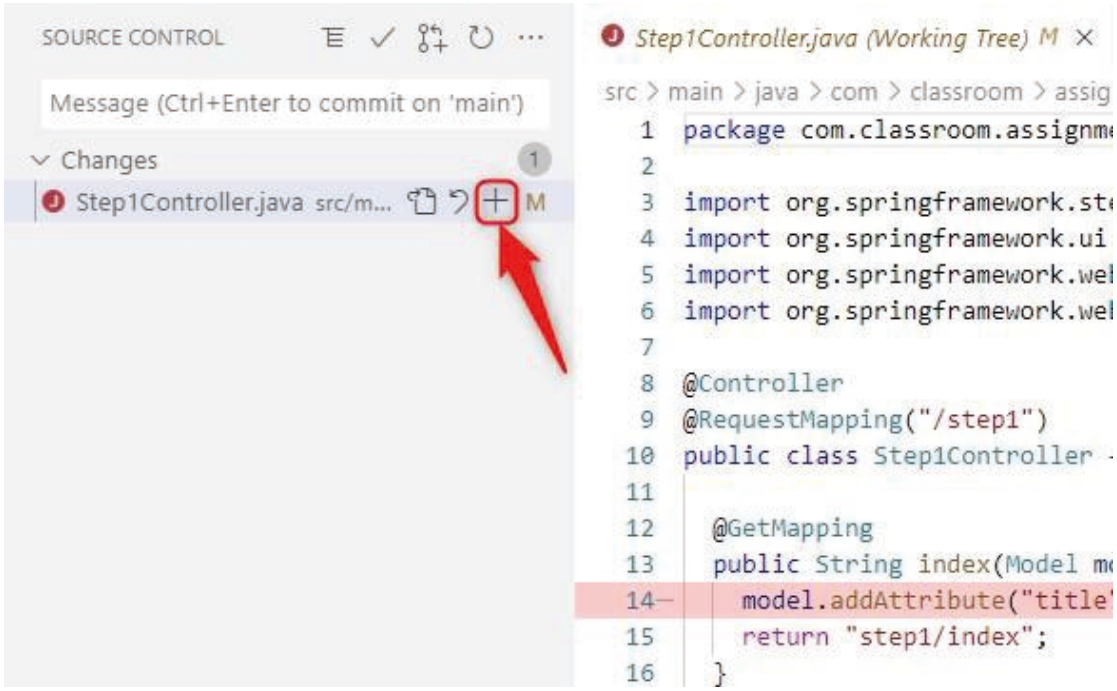
2. これまでに行ったファイルへの変更ないようが表示されています。

変更内容に問題がないか今一度確認しましょう。

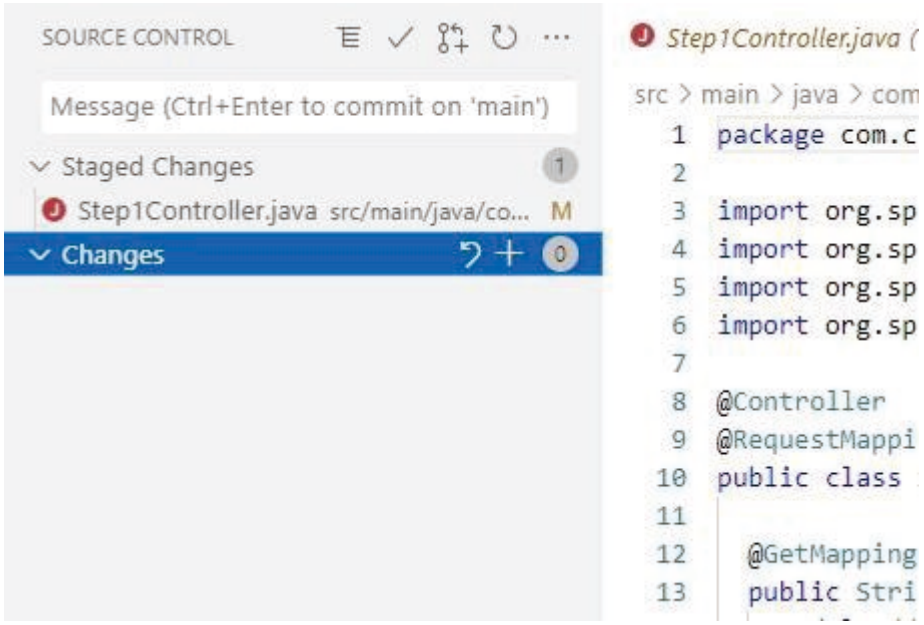


3. コミットするファイルを選択しましょう。

コミットしたいファイルの横の＋ボタンを押しましょう。



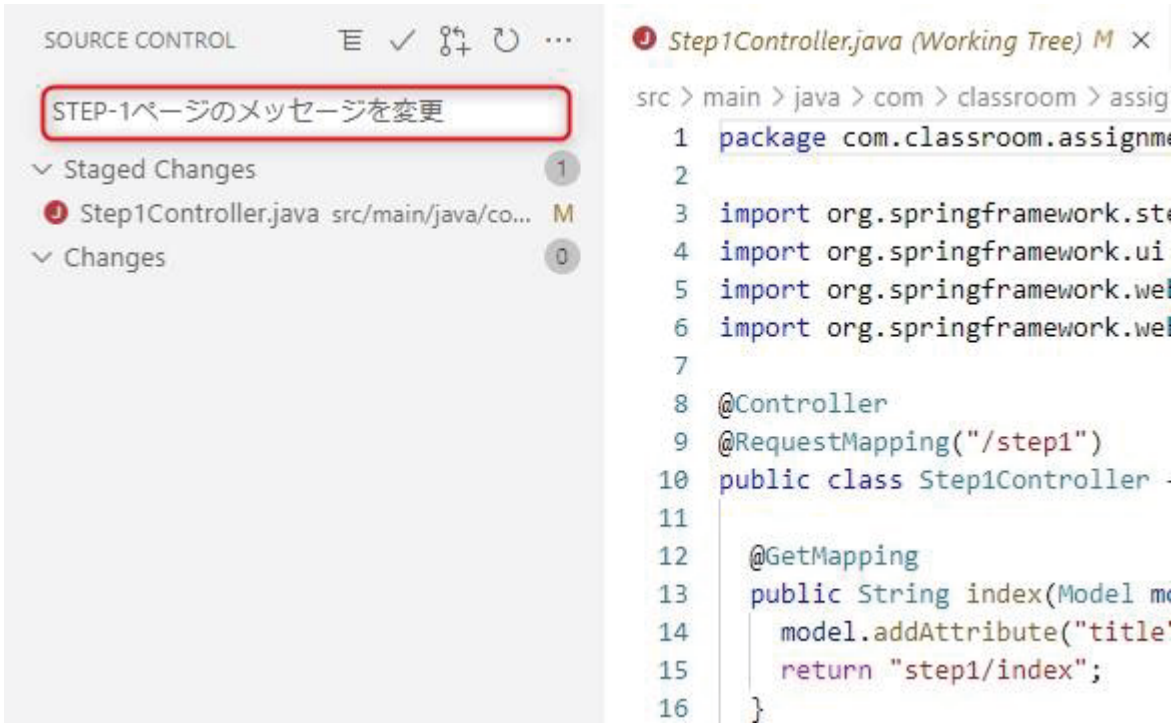
4. Staged Changes にコミットしたいファイルが含まれていることを確認しましょう。



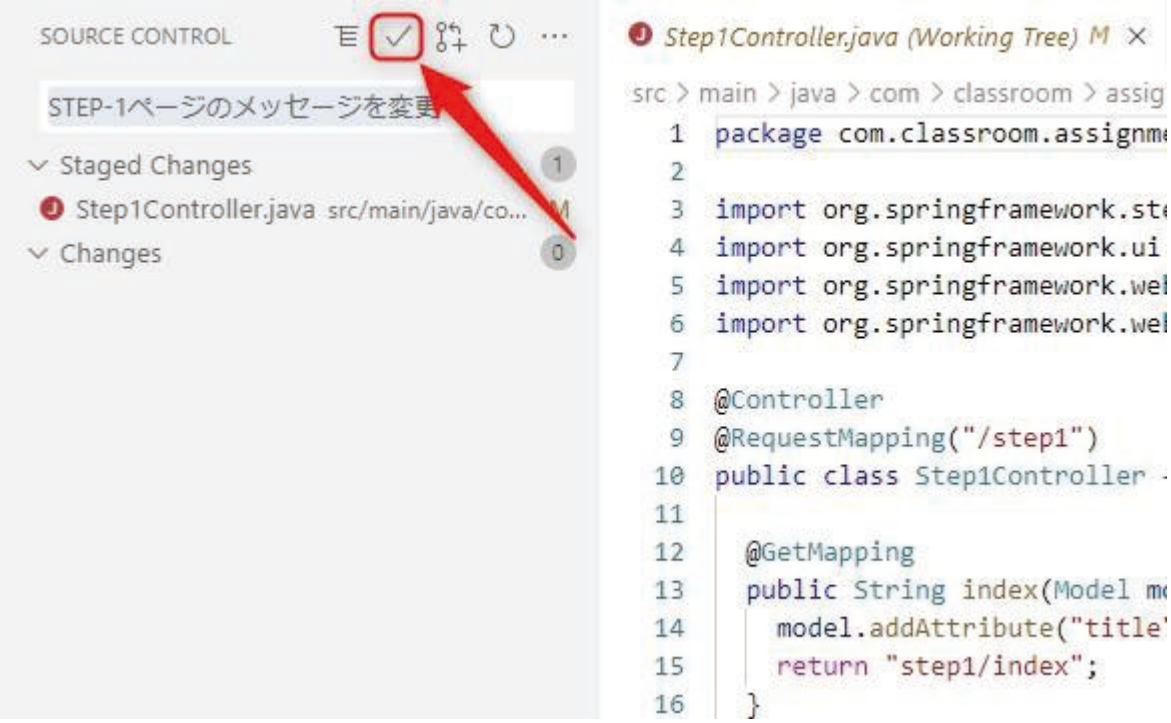
5. Message 欄に Commit Message(コミットメッセージ)を入力しましょう。

- Commit Message には、今回のコミットの追加・変更内容につ
- いて記載します。

モブプロでは、**ナビゲーターの名前**を記載してください



6. チェックボタンを押してコミットしましょう。



.

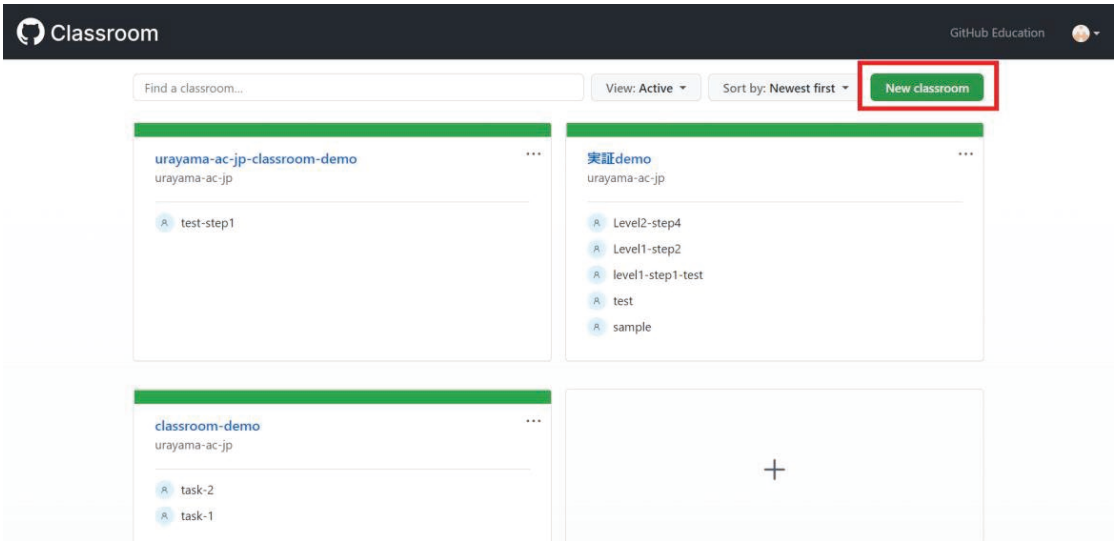
7. モブプロの順番を回し、次のタームを開始してください。

記録係の方は、コミットを行ったチェックを評価表に記入してください。

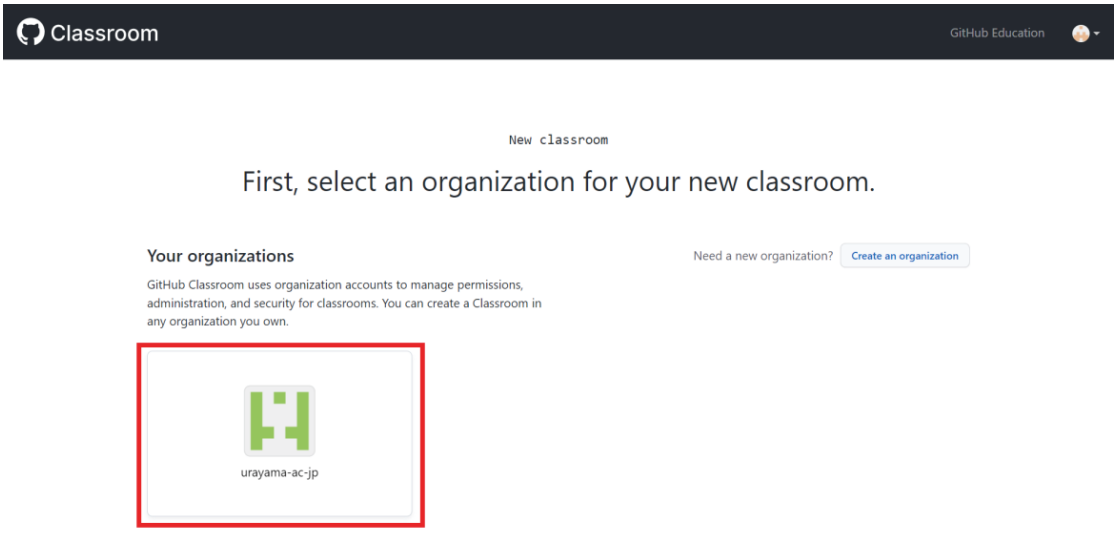
課題の作成

Github Classroom の操作

- 先生(TA)・学生・クラス・課題を設定
  - 授業用のクラスを作成

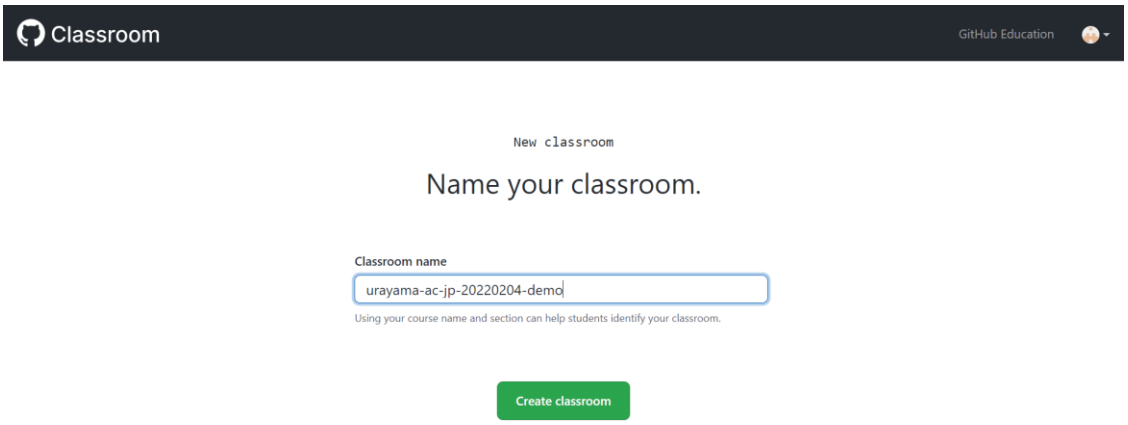


  - 所属する組織（浦山学園）を選択

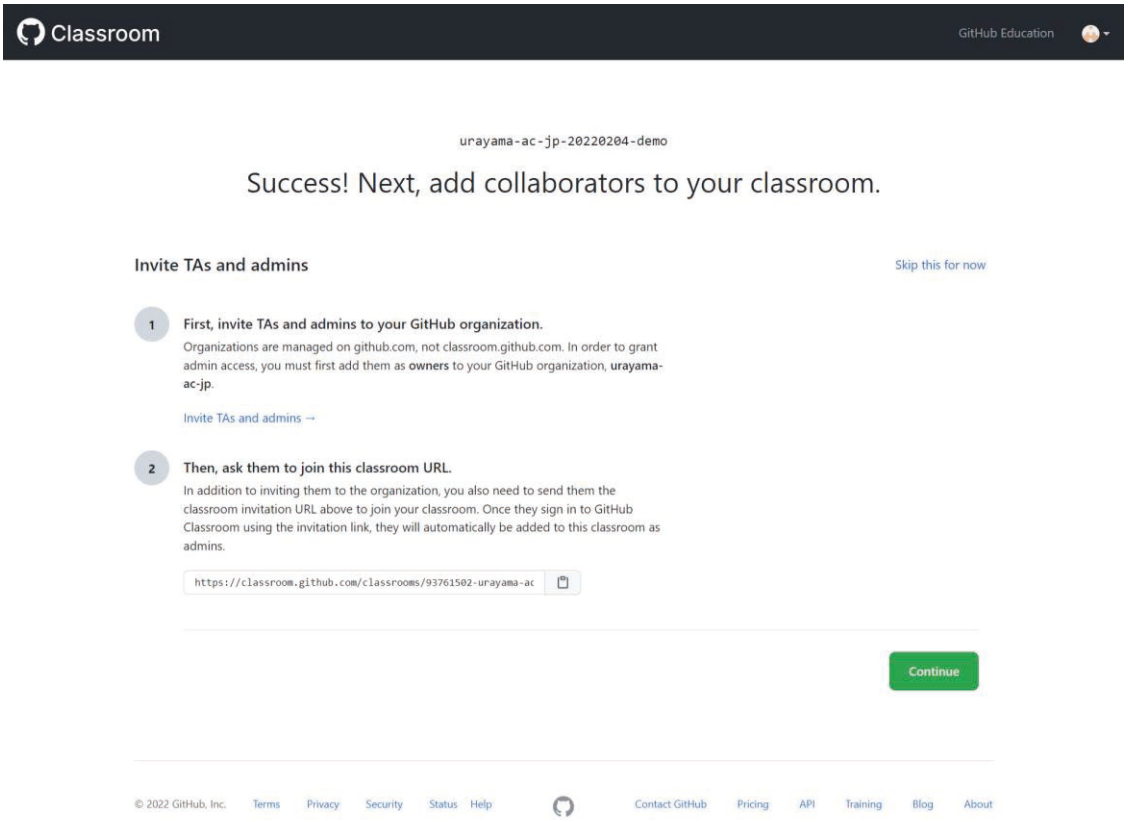


.

– 授業名を設定

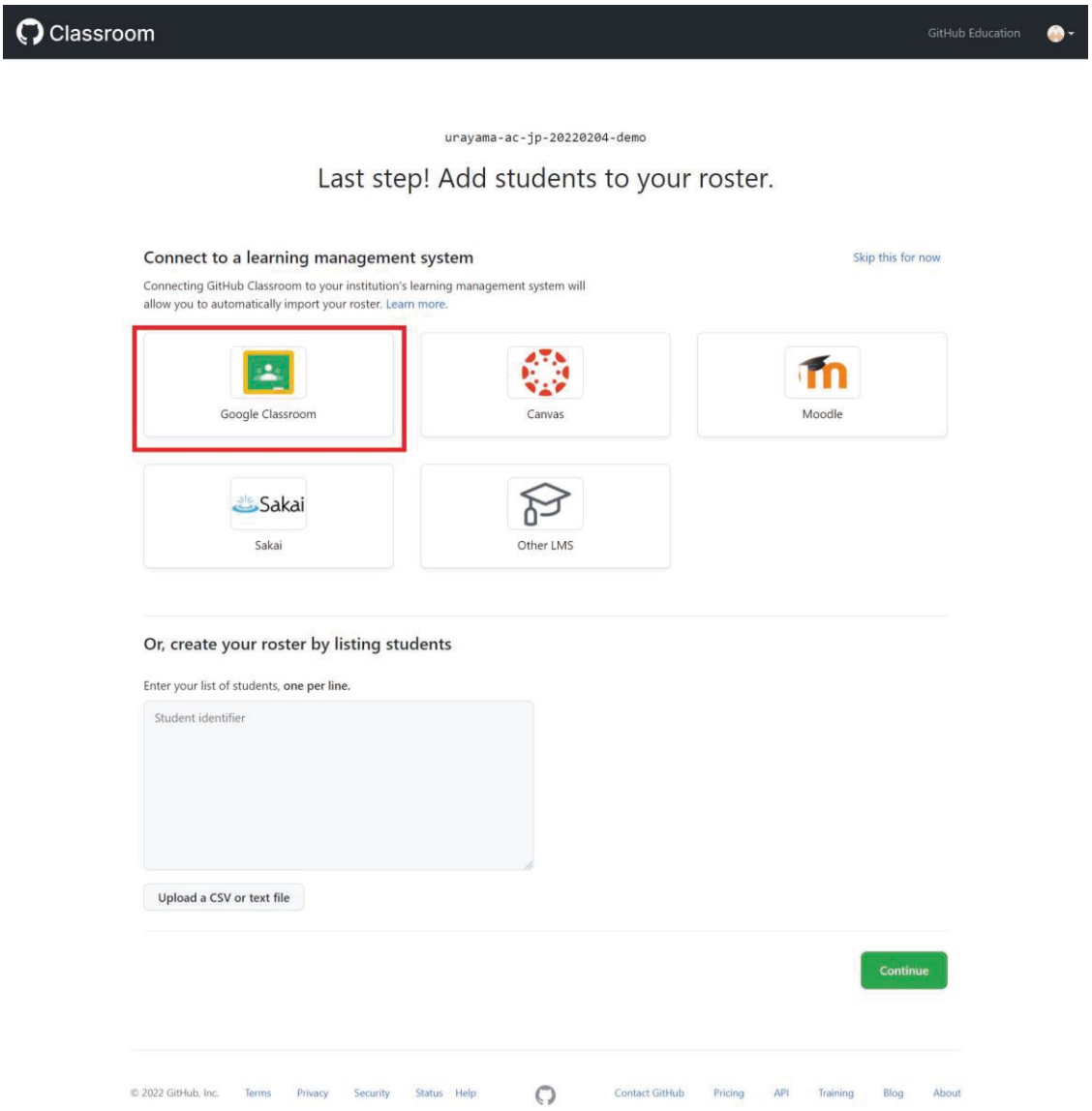


– 必要であれば、他の先生や TA の招待も可能



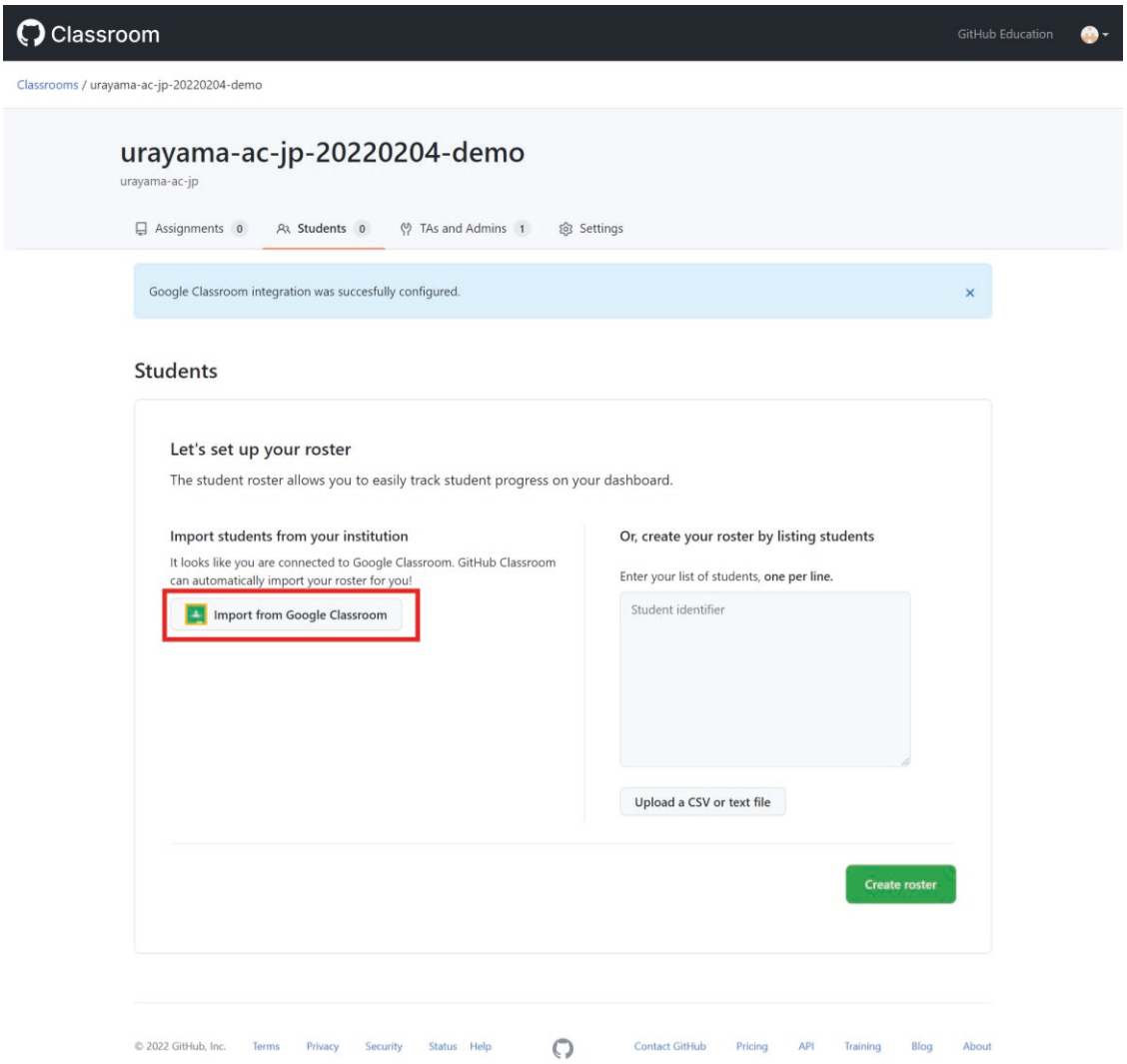
.

– 授業に紐づける学生を名寄せ（今回は GoogleClassroom から連携）



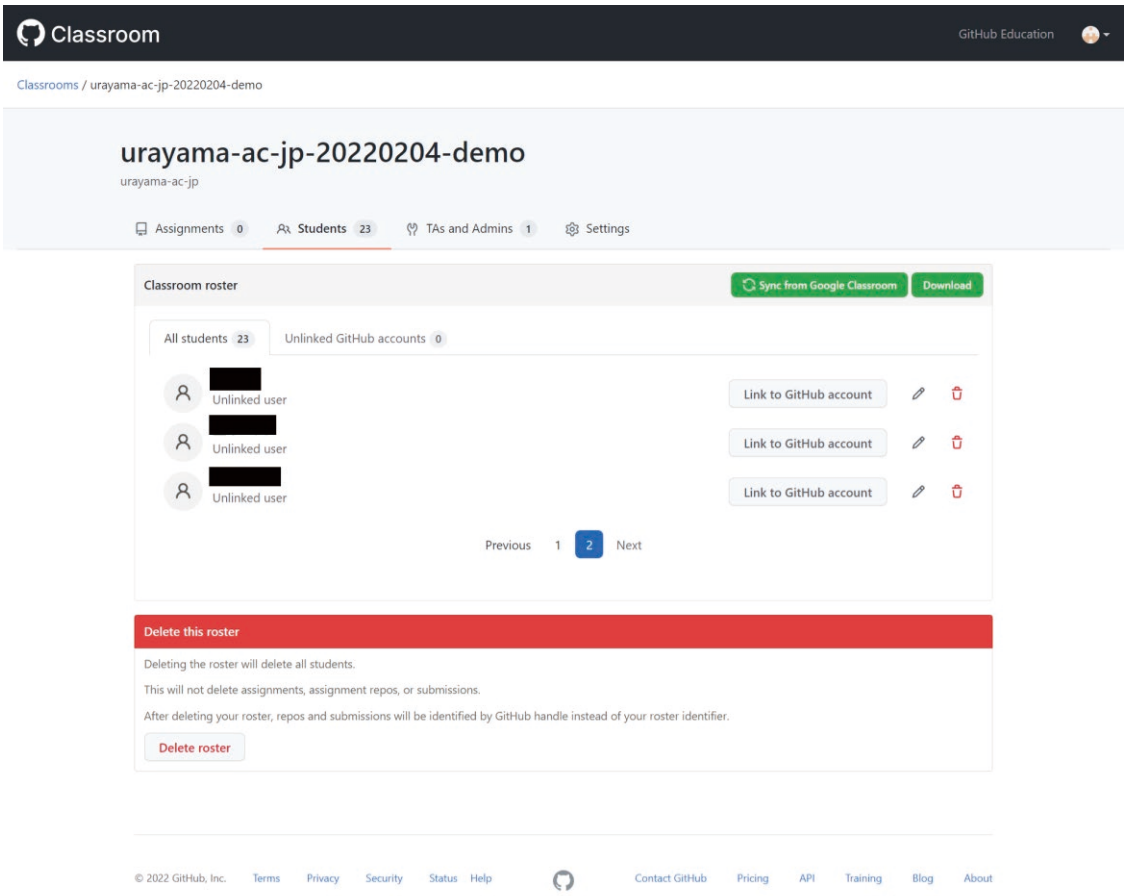
.

– 連携された情報を登録


Classroom
GitHub Education
Classrooms / urayama-ac-jp-20220204-demo
urayama-ac-jp-20220204-demo
urayama-ac-jp
Assignments 0
Students 0
TAs and Admins 1
Settings
Google Classroom integration was succesfully configured.
Students
Let's set up your roster
The student roster allows you to easily track student progress on your dashboard.
Import students from your institution
It looks like you are connected to Google Classroom. GitHub Classroom can automatically import your roster for you!
Import from Google Classroom
Or, create your roster by listing students
Enter your list of students, one per line.
Student identifier
Upload a CSV or text file
Create roster
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Help
Contact GitHub
Pricing
API
Training
Blog
About

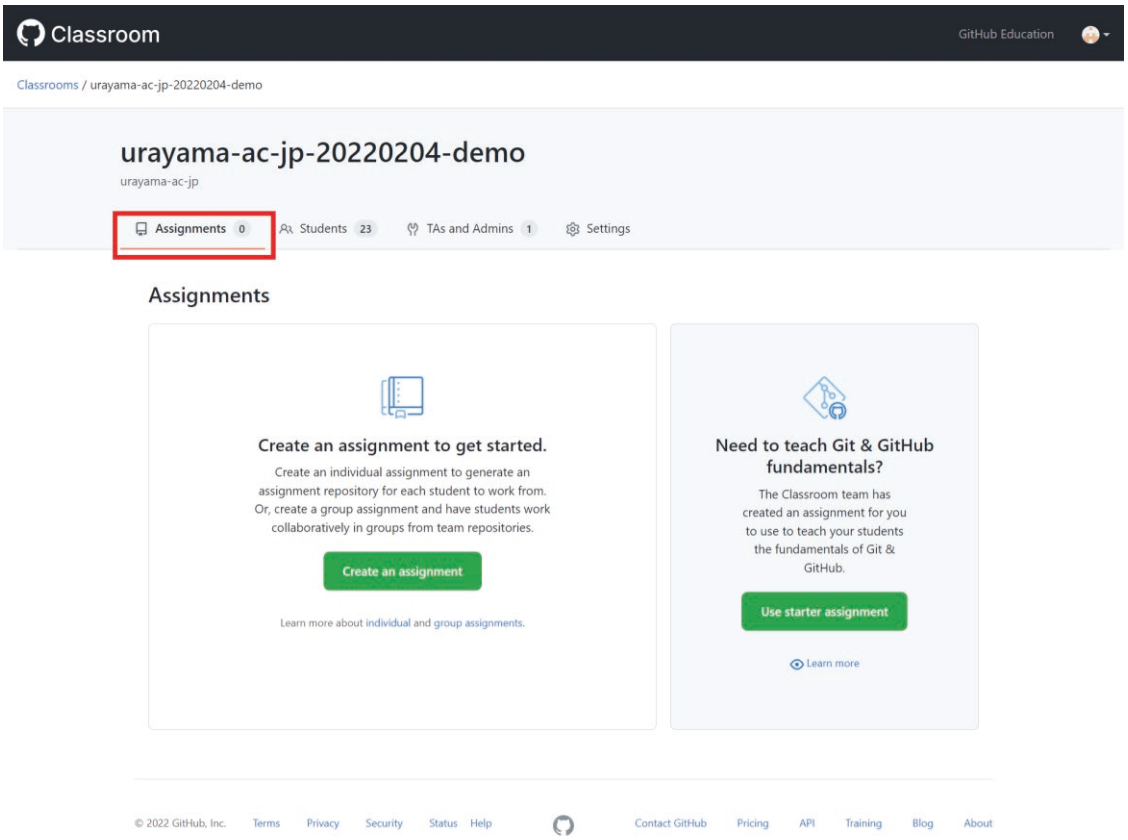
.

– クラスに名寄せされた学生の一覧を確認



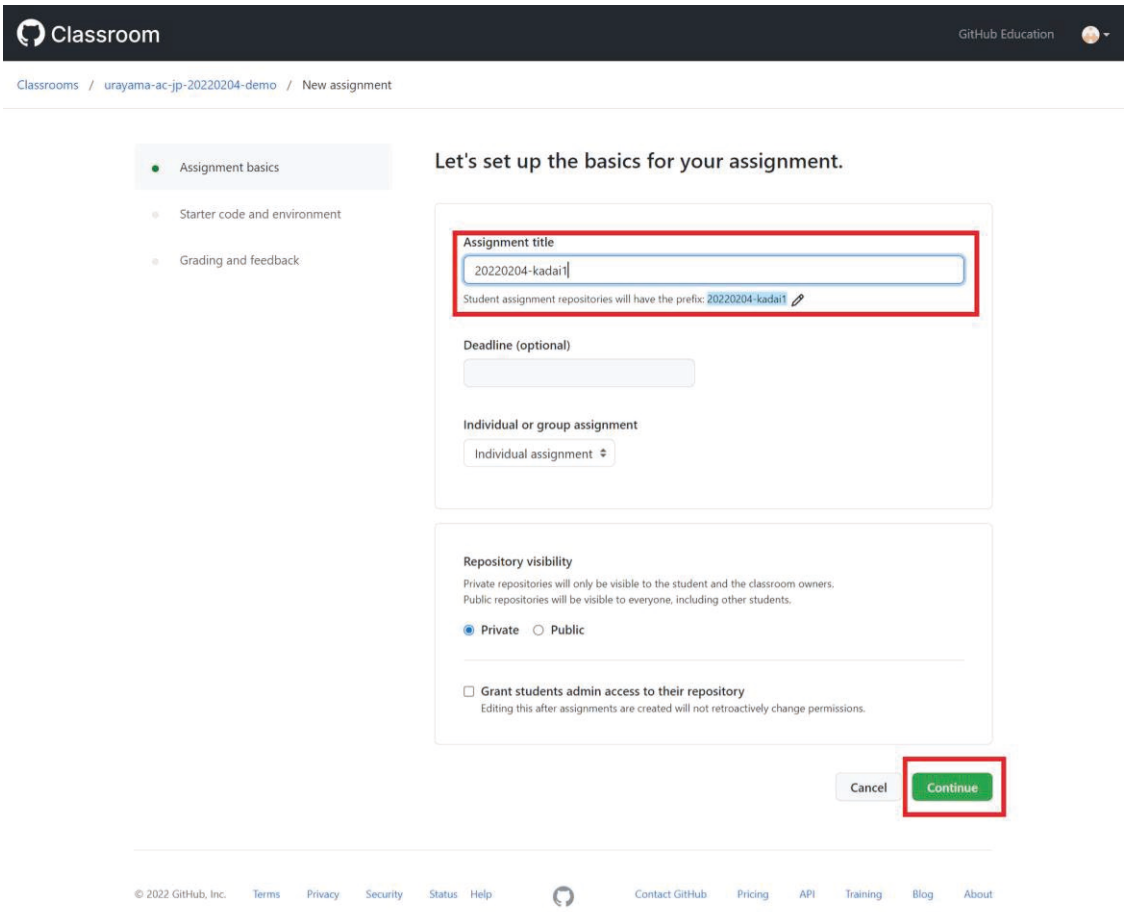
.

– 課題の登録を行っていく(課題はクラスを作った後で作成することも可能)



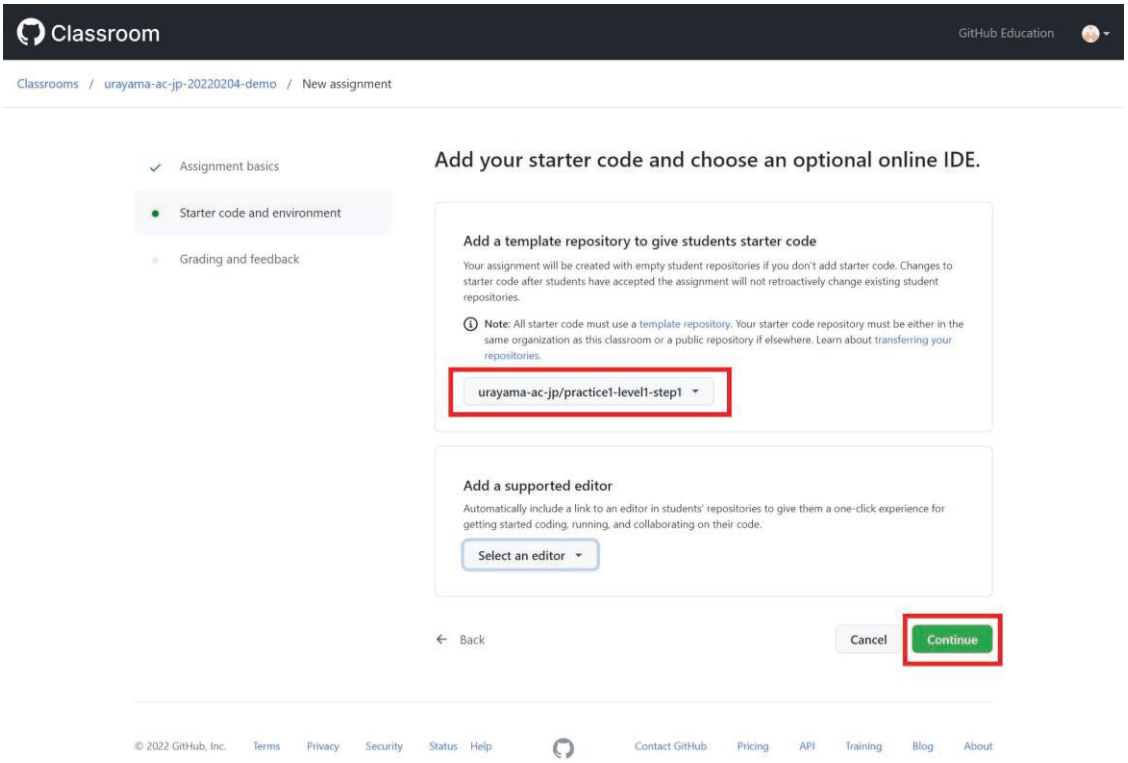
.

– 課題名を設定

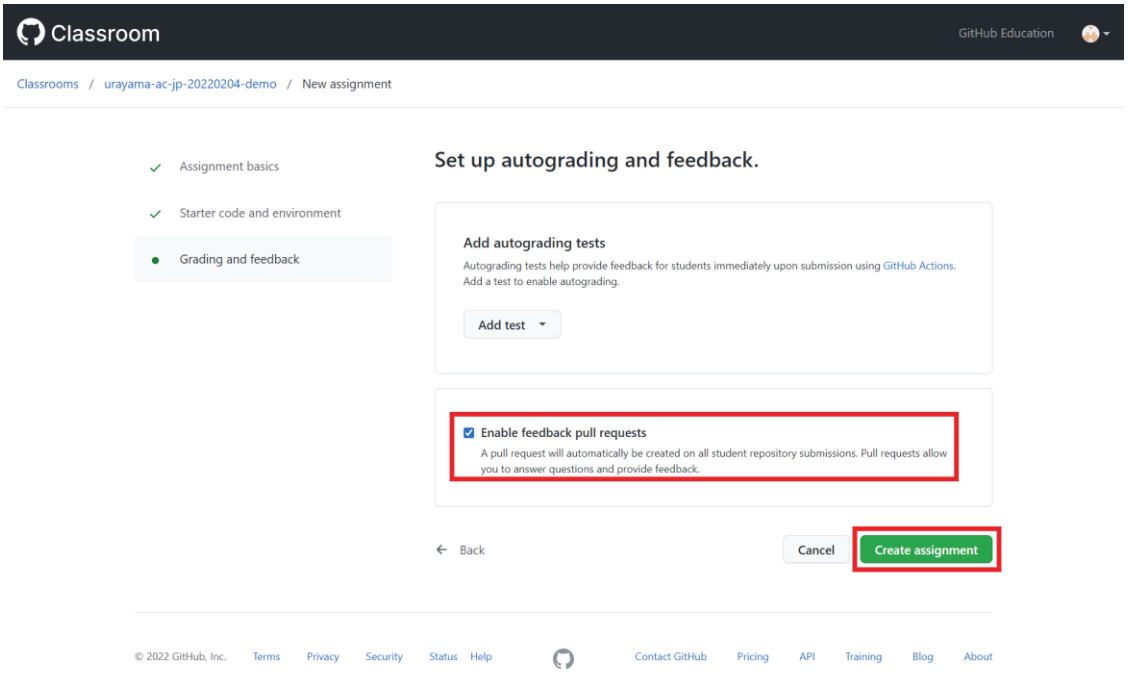

Classroom
GitHub Education
Classrooms / urayama-ac-jp-20220204-demo / New assignment
Assignment basics
Starter code and environment
Grading and feedback
Let's set up the basics for your assignment.
Assignment title
20220204-kadai1
Student assignment repositories will have the prefix: 20220204-kadai1
Deadline (optional)
Individual or group assignment
Individual assignment
Repository visibility
Private repositories will only be visible to the student and the classroom owners.
Public repositories will be visible to everyone, including other students.
Private
Public
Grant students admin access to their repository
Editing this after assignments are created will not retroactively change permissions.
Cancel
Continue
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Help
Contact GitHub
Pricing
API
Training
Blog
About

.

– 学生に配布するリポジトリを選択（ここで選択されたリポジトリがコピーされて、学生が課題を行うリポジトリが作成される）

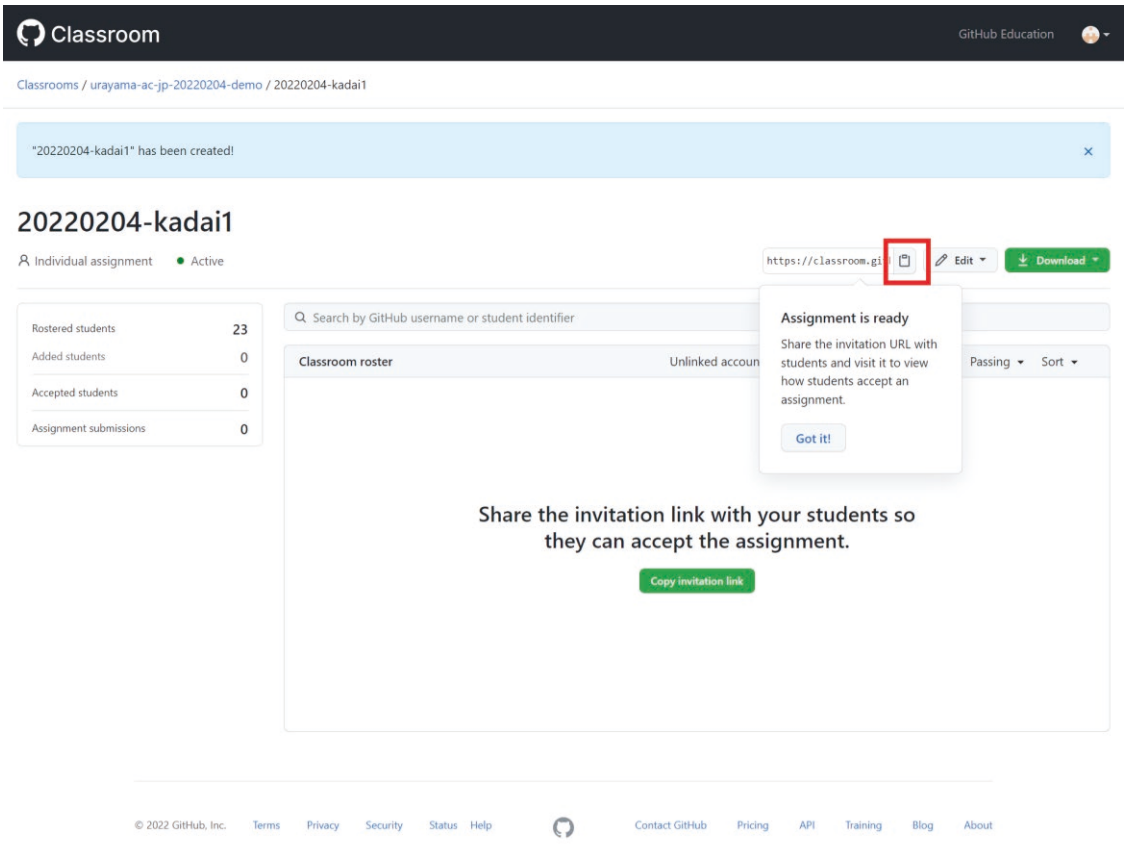


– フィードバックを行う設定を追加



.

– 課題を配布するための URL が発行されるのでコピー※別途プラットフォームを使って配布



授業の作成

Slack の操作

- 各チームごとにチャンネルを用意する

チャンネルを作成する
チャンネルとはチームがコミュニケーションを取る場所です。特定のトピックに基づいてチャンネルを作ると良いでしょう (例: #マーケティング)。
名前
チーム名
説明 (任意)
所属メンバーなどのチーム情報
このチャンネルの目的は？
プライベートチャンネルにする
チャンネルをプライベートに設定すると、招待されたメンバーのみ閲覧または参加できるようになります。
もっと詳しく
作成

就職マッチング用
ポートフォリオ作成説明書

ポートフォリオとは

- 「自分は何ができるのか」をアピールするため、
  以下を示したもの
  - 自身の技術や知識のレベル
  - これまでの経歴、実績

- 具体的には、これまでに手掛けてきた成果物を示す
  - 「作品集」「制作物集」と考えてもよい
  - 学生の皆さんは、授業で作成したプログラムやアプリを

- 対企業を意識して、ポートフォリオを作成してみましょう

ポートフォリオを公開

- IT業界では、ポートフォリオをWEB上に公開されることが多い
  - WEBサイトを自作して公開する場合、以下のような手順が必要
    - ドメインの取得
    - サーバーと契約
    - Webサイト開発
  - 手間だけでなく費用が掛かる

- そこで、今回はGitHub Pagesを利用する

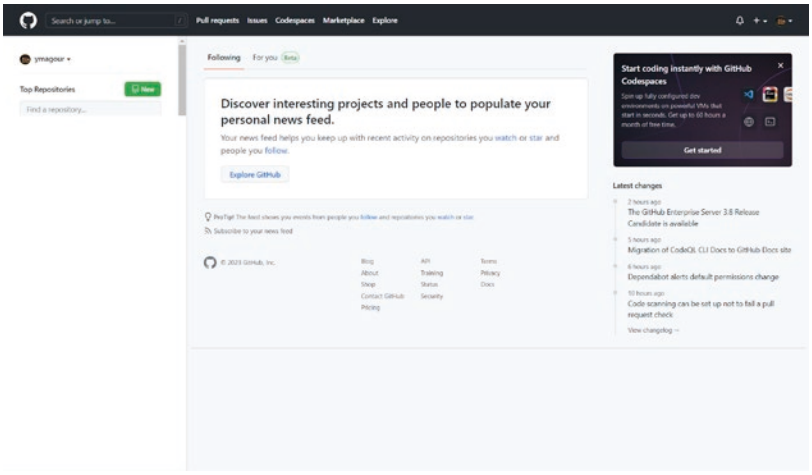
Github Pagesとは

- Github公式より
  - GitHub Pages は、GitHub のリポジトリから HTML、CSS、および JavaScript ファイル を直接取得し、任意でビルドプロセスを通じてファイルを実行し、ウェブサイトを公開できる静的なサイトホスティングサービスです。 GitHub Pages サイトの例は、GitHub Pages サンプル コレクションで確認できます。
    引用: https://docs.github.com/ja/pages/getting-started-with-github-pages/about-github-pages
  - ざっくり要約すると
    GitHubにリポジトリと呼ばれるものを作り、そこにHTMLやCSSといったファイルを置くことで、自動的にWEBサイトに公開できる

- GitHubのアカウントがあれば、Github Pagesは
  無料で利用できる
  - 今回は教育PFを使用する際に作成したGitHubアカウントを利用する

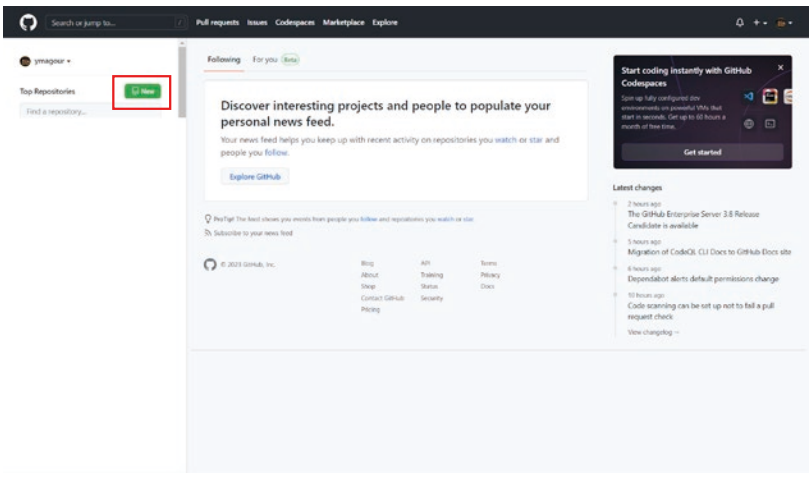
ポートフォリオサイトの作成

- https://github.com にアクセスして、ログイン



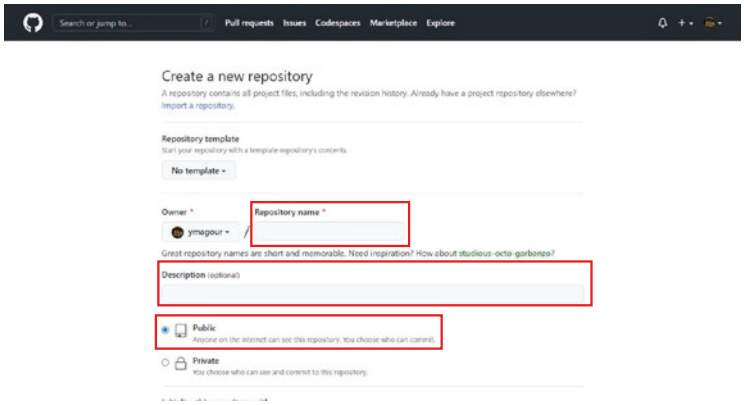
ポートフォリオサイトの作成

- [New]ボタンから新しいリポジトリを作成



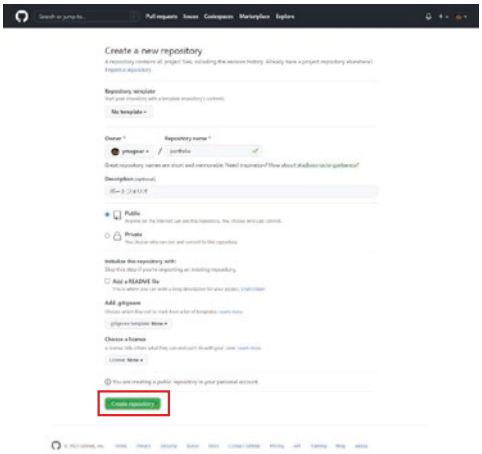
ポートフォリオサイトの作成

- 以下を設定
  - Repository name：portfolio（任意の名前でもOK）
  - Description：任意の説明文を入力
  - 公開範囲：Public



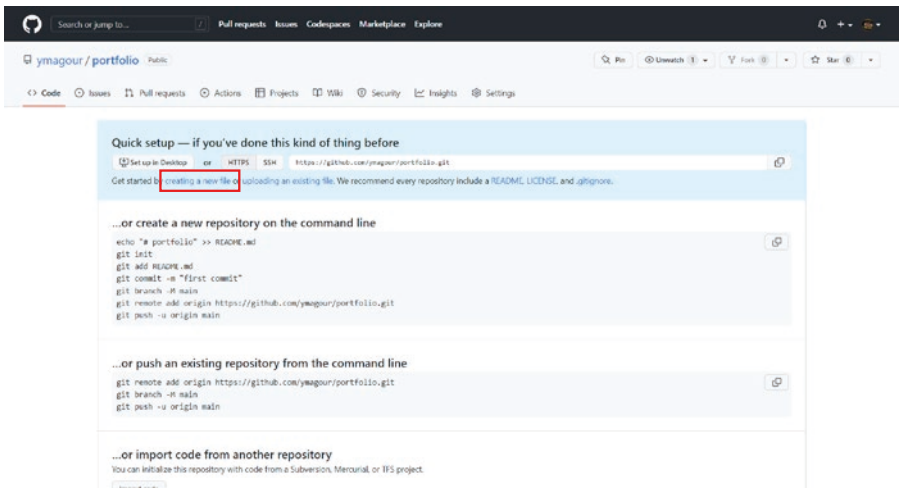
ポートフォリオサイトの作成

- [Create repository]をクリック



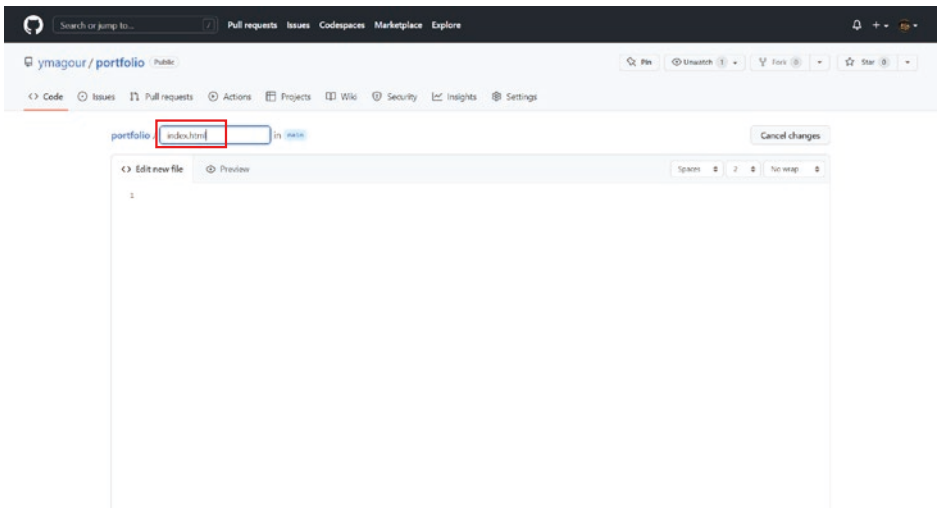
ポートフォリオサイトの作成

- [create a new file]をクリック



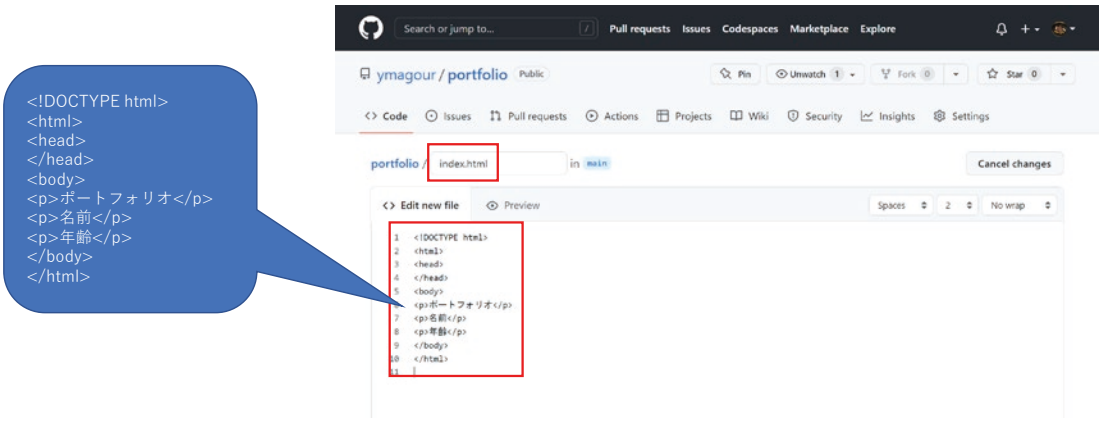
ポートフォリオサイトの作成

- index.html（サイトで一番最初に表示される画面）を作成



ポートフォリオサイトの作成

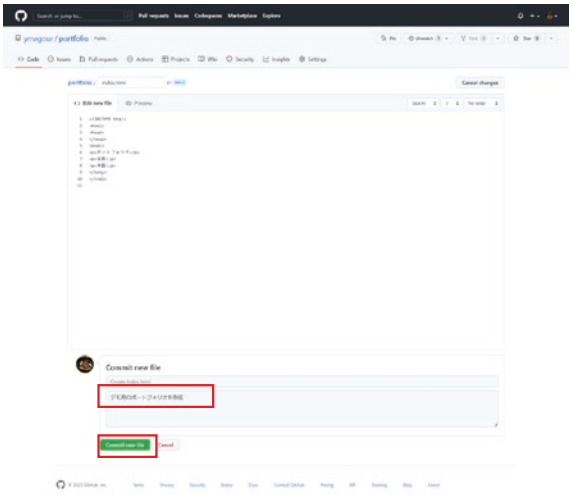
- index.html（サイトで一番最初に表示される画面）を作成
  - まずはデモ用のページを作成（吹き出しの内容を入力）



ポートフォリオサイトの作成

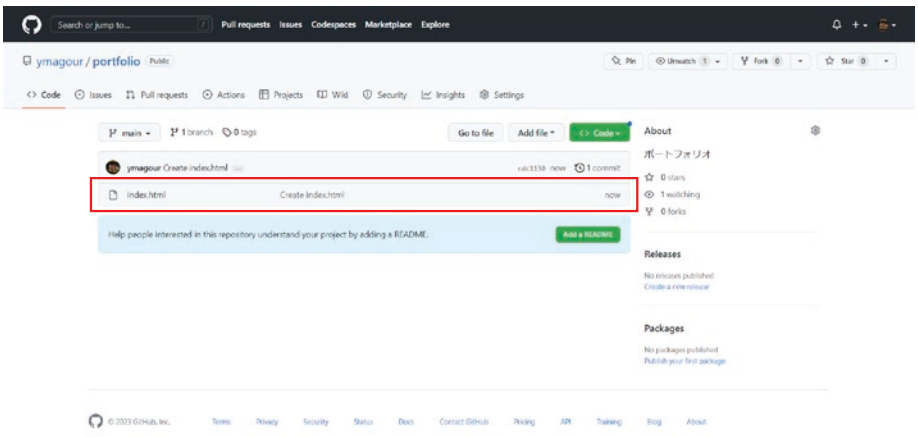
- 作成が終わったら下へスクロール

- 記入欄に作成したファイルの説明を記述して[Commit new file]をクリック



ポートフォリオサイトの作成

- 作成したindex.htmlが追加されたことを確認



実際に公開してみよう

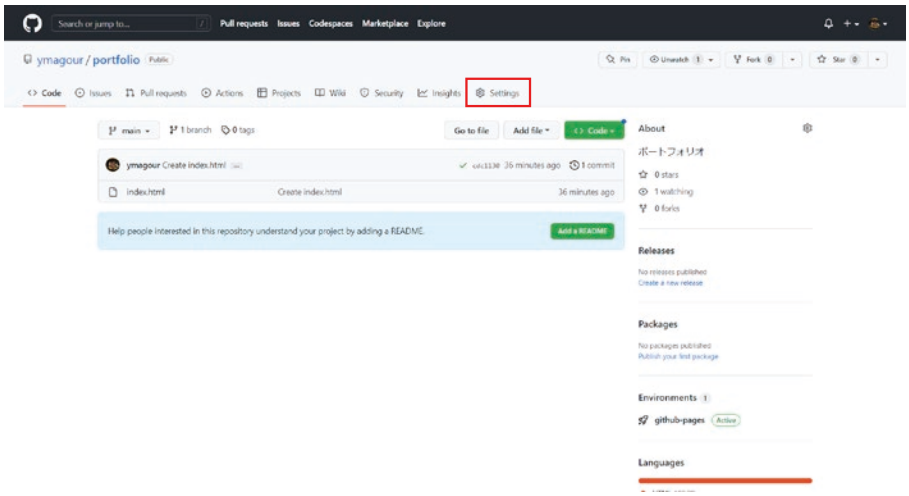
- 作成したindex.htmlを公開してみましょう

- 必要なのはGitHub Pagesの設定のみ
  - GitHub Pagesに表示させるリポジトリを選択

- 以下のURLでアクセスが可能なWEBページができる
  - https://GitHubアカウント名.github.io/リポジトリ名
    - 例： https://ymagour.github.io/portfolio/

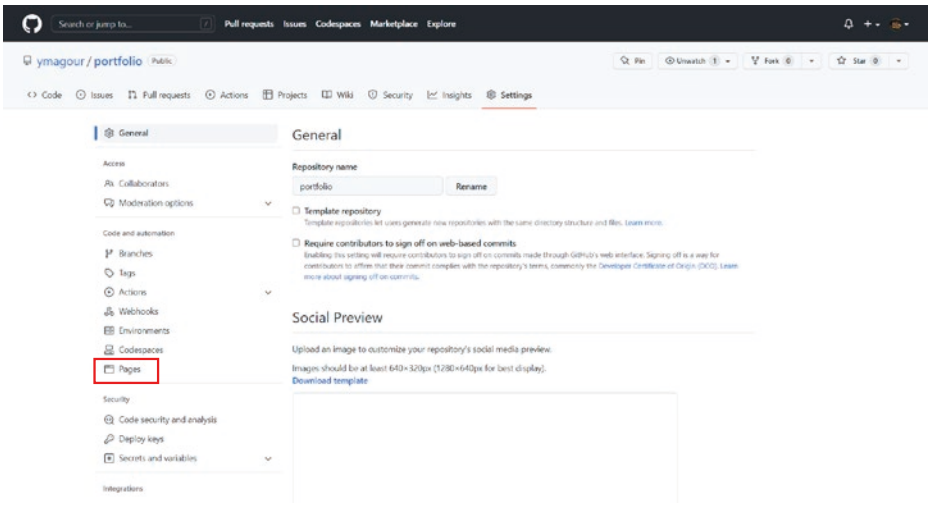
ポートフォリオを公開

- [Settings]をクリック



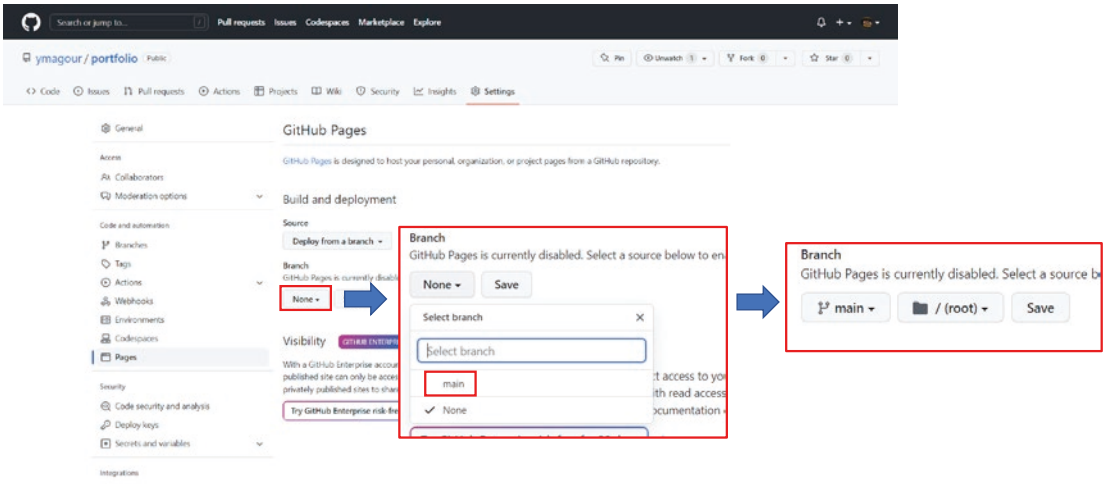
ポートフォリオを公開

- 左メニューの[Pages]をクリック



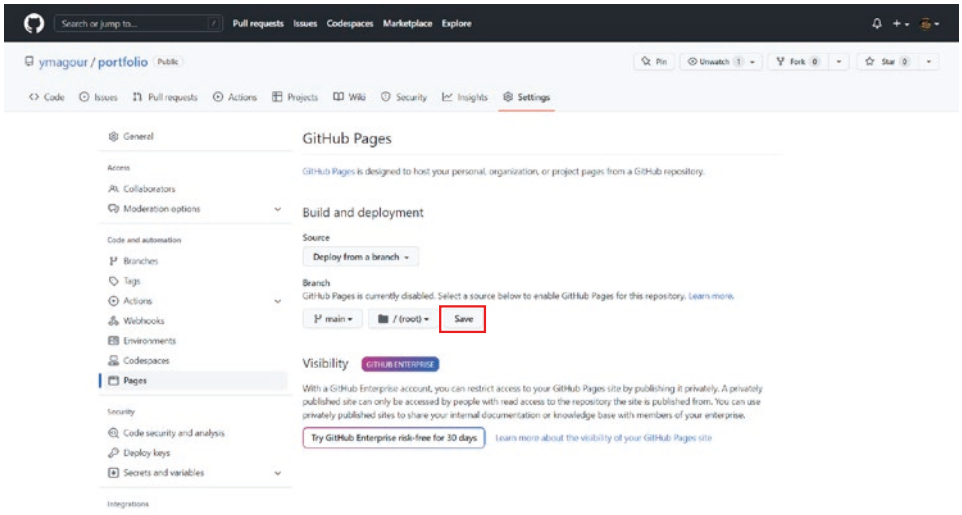
ポートフォリオを公開

- Branchの[None]となっているところを[main]に変更



ポートフォリオを公開

- [Save]をクリック

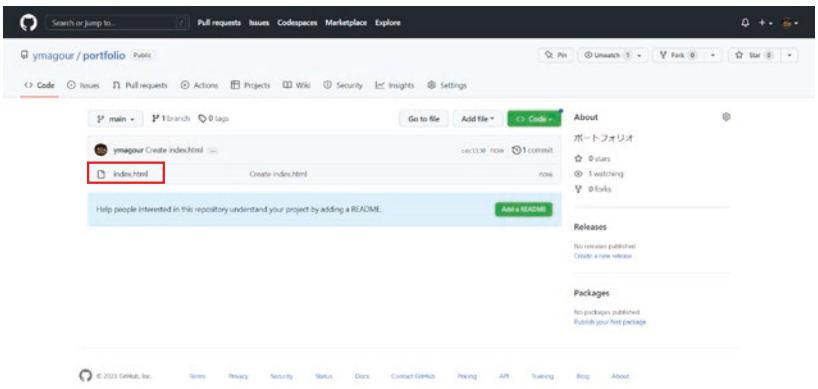


ポートフォリオを公開

- 以下にアクセスして公開されているか確認
  - https://GitHubアカウント名.github.io/リポジトリ名
    - 例： https://ymagour.github.io/portfolio/

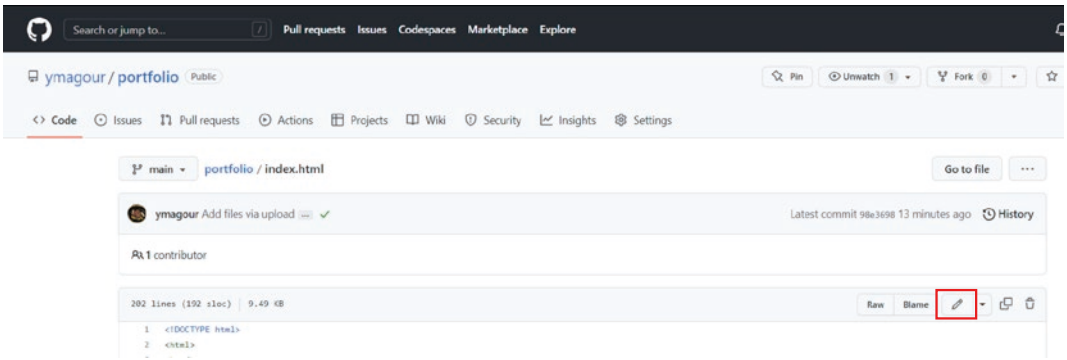
公開したページを編集

- 公開したindex.htmlの内容を変更してみましょう
  - [Index.html]をクリック



公開したページを編集

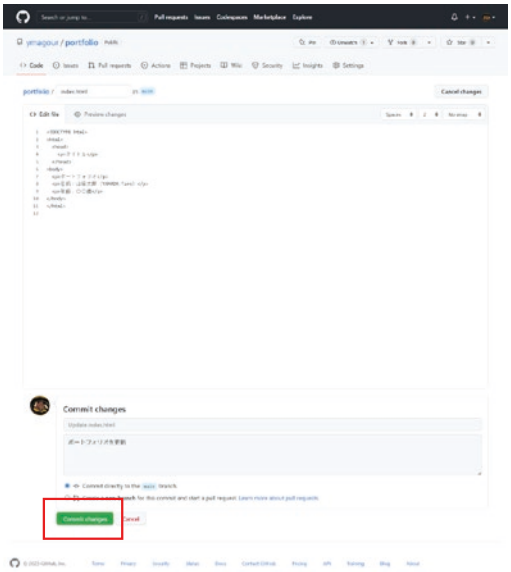
- 鉛筆（編集）マークをクリック



公開したページを編集

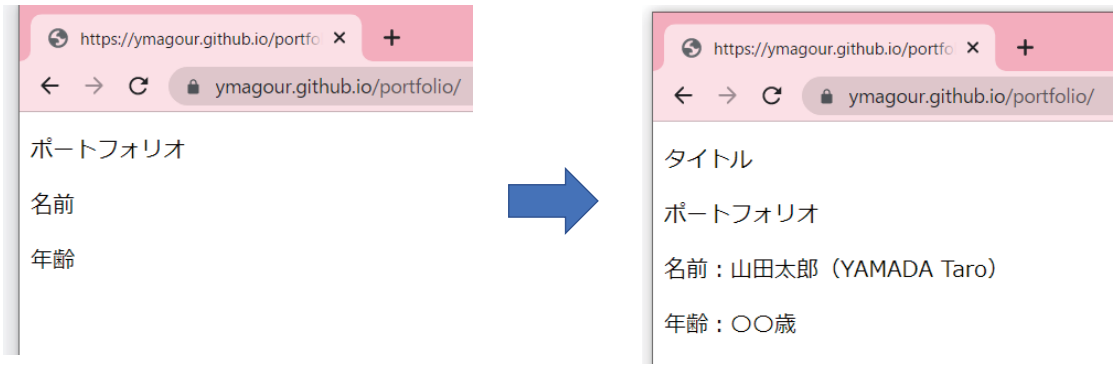
- 内容を更新
  - 今回は名前と年齢部分を変更

- [Commit Changes]の
  緑ボタンをクリック



公開したページを編集

- 再び公開したページにアクセスすると内容が変更されています
  ※変更内容が反映されるまで少し時間が必要となります

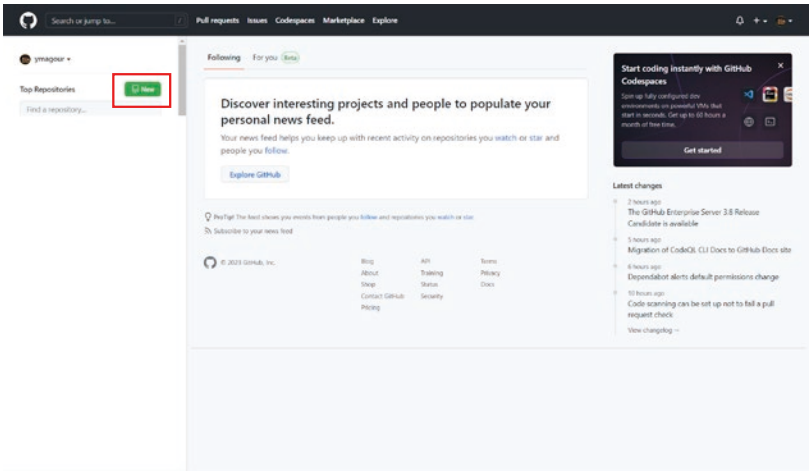


公開したページを編集

- ここまでで、GitHub Pagesを使ってWEBページを公開
- ここからは、自分のプログラムを公開できるよう準備
  - これまで授業や自作したプログラムをGitHubに登録
    - 登録先のリンクを取得
    - このリンクを共有することで、他者にプログラムを見てもらえる
  - そのリンクをindex.htmlに載せる形でソースを公開
    - IT企業の転職では、GitHubでソースを公開することで、
      どのようなプログラムを作ってきたかを見られるケースもある

自作したプログラムをGitHubに登録

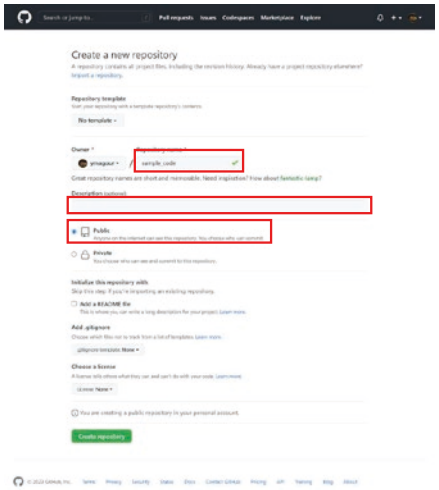
- [New]ボタンから新しいリポジトリを作成



自作したプログラムをGitHubに登録

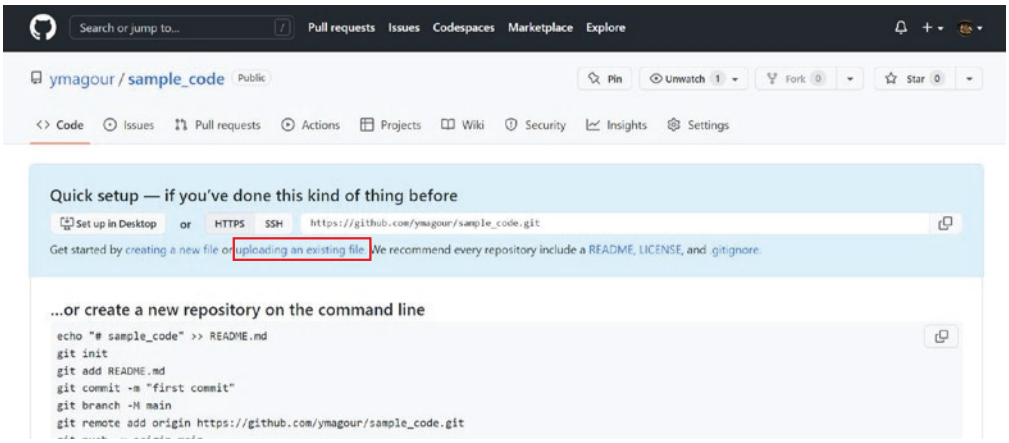
- 以下を設定
  - Repository name：sample_code（任意の名前）
  - Description：任意の説明文を入力
  - 公開範囲：Public

- [Create repository]をクリック



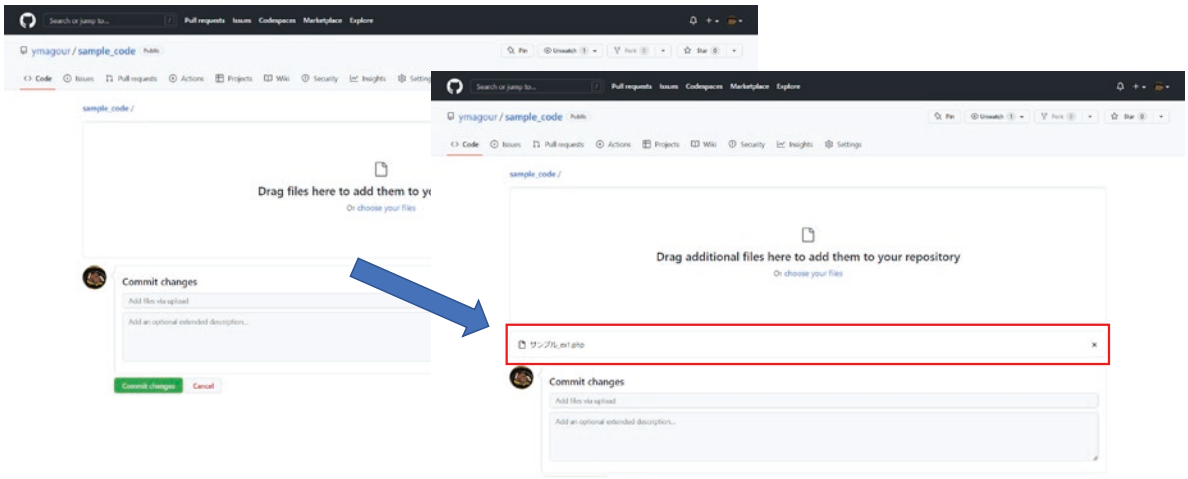
自作したプログラムをGitHubに登録

- [uploading an existing file]のリンクをクリック



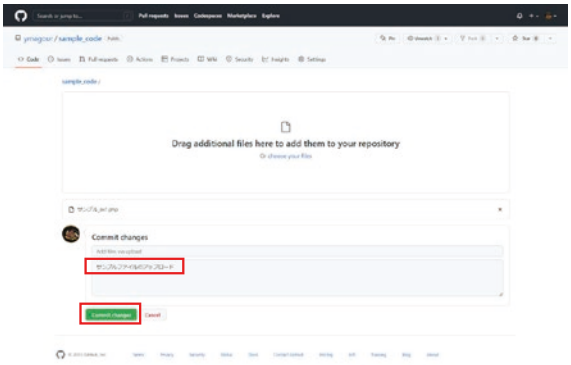
自作したプログラムをGitHubに登録

- 公開したいソースなどをアップロード



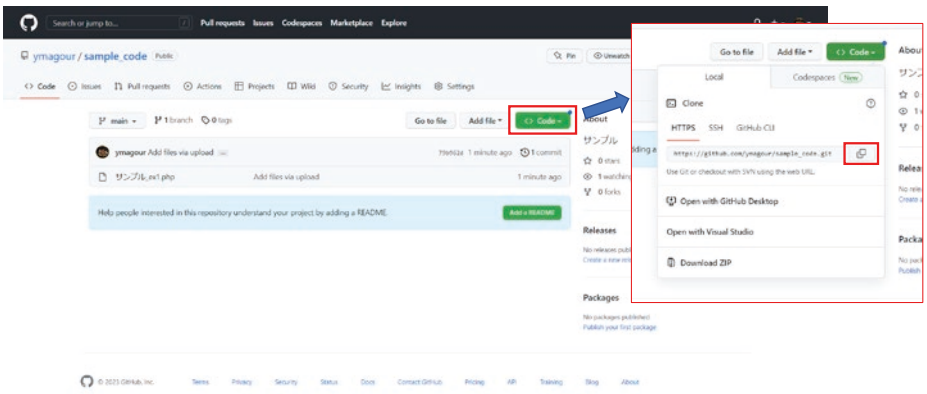
自作したプログラムをGitHubに登録

- 記入欄に作成したファイルの説明を記述
- [Commit new file]をクリック



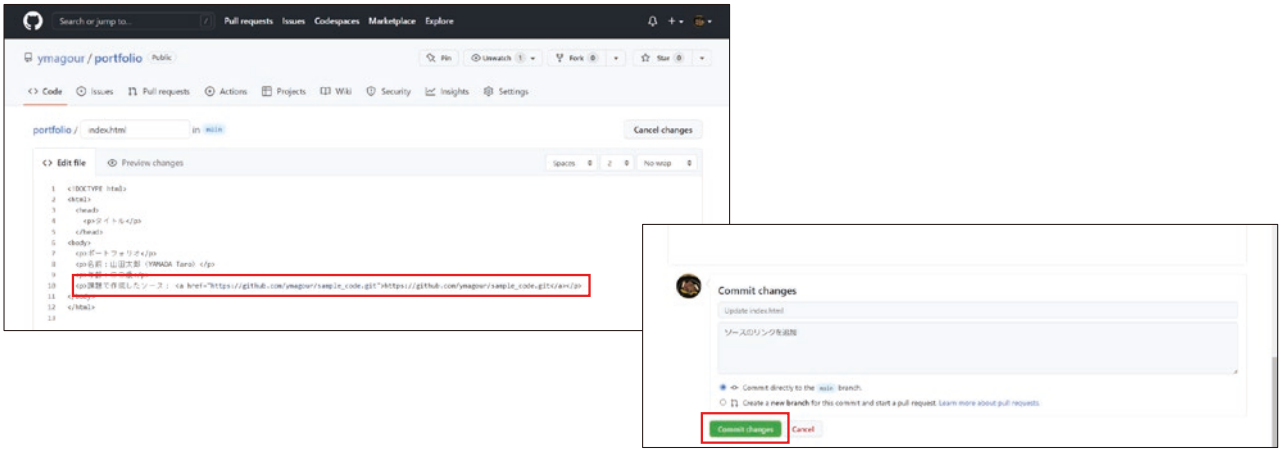
自作したプログラムをGitHubに登録

- 公開されたURLを取得



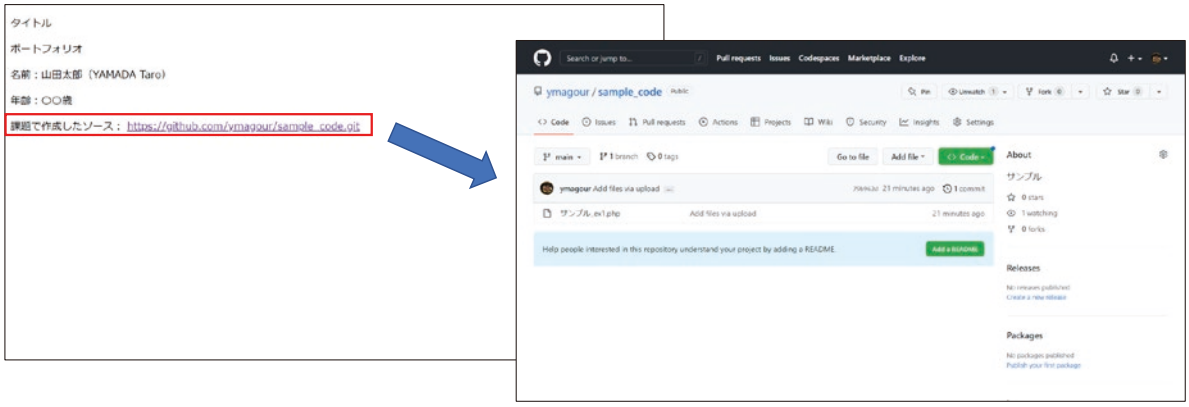
ポートフォリオにソースを追加

- ポートフォリオ用のindex.htmlに
  自作したプログラムのリンクを追加



ポートフォリオにソースを追加

- リンクのクリックし、登録したリポジトリが見えればOK



最後に

- 非常に簡単にポートフォリオを作成できる

- 今回のように、まずはindex.htmlを編集する形で
  自分のポートフォリオを作成していきましょう
  - 学校で作ったプログラムなどを作品集としてまとめてみましょう

- 見栄えをよくしたいという人はテンプレート（HTMLやCSSなど）を
  活用してみましょう（無料で使えるものも多いです）

学習管理ユニット

学習管理ユニット - チャットボット

概要

- 学習管理ユニット内のチャットボットに設定した項目について記載する
- なおチャットボットには、IBM社のWatson Assistantを使用する
- また、実証期間中にあった質問については、随時追加予定
- 以下にはWatsonに登録したQAを記載する

登録内容一覧

課題について

- 質問 : 何をすればいいのかが分かりません。
    - まずはSlackで招待されたリンクにアクセスしてみましょう。
    - アクセスした後、添付の手順書をもとに作業を進めてみてください。
    - https://files.slack.com/files-pri/T0303GCDEJY-F031VN2EBU3/_____.pdf?t=xoxe-3003556456644-3807621841623-3815601382166-03f675b00cafd30cdea1f1b49233cc21
- 質問 : レベル4の課題がわかりません。
    - まずは、他の課題のソースを参考にしましょう。
    - 他のレベルのソースをコピー&ペースト + ファイル名のリネームを行い、ソースを用意しましょう。
    - そして、画面から何が送られてくるか、また何を画面に返したらいいかを確認しましょう。
- 質問 : レベル4が終わりました、追加の課題はありますか？
    - 課題はレベル4までになります。
    - それ以降について、自分の欲しい（必要）と思う機能を自身で追加してみましょう。
- 質問 : Getter/Setter(ゲッター/セッター)の作成でソースアクションを選択したのですが、利用可能なソースアクションがありませんと出てソースアクションの選択ができません。
    - 環境構築時の読み込み等がうまくいっていない可能性があります。
    - 一度、プログラムを実行させてうまくいかなかった読み込みを再実行させてみましょう
    - 左メニューの「実行とデバッグ」から実行（デバックなし）してみましょう

開発環境について

- 質問 : codespacesが起動しません
    - 既にcodespacesが複数作成されていることが予想されます。
    - 以下の手順で不要なcodespacesを削除しましょう。
    - 削除手順(https://drive.google.com/file/d/1vfdQB0dNzaA0rqQfiWu0OOaCHAXLhxyi/view)
- 質問 : 手順にあるプログラミングを開始できるまでに時間がかかります。これは毎回ですか？

    - 各課題の初回起動時にプログラミングを行う環境を作っています。そのため、新しい課題を始める際は時間がかかります。
    - ただ、課題を中断、再開する際は、短い時間で起動されます。

- 質問 : codespacesの手順にあるインストールのダイアログが表示されません

    - ダイアログの確認をスキップして次項の手順（以下）を実施してください。
    - 7. 虫と再生ボタンのアイコンを押します
    - 8.Launch AssignmentApplicationが選択されていることを確認し再生ボタンを押します。
    - 初回はダイアログが表示されます、YESを選択してください。
    - 9. Open in Browserを押します。(表示されるまでしばらく時間が掛かります。)
    - 上記を実施後、
    - 10. Webページが表示されます。
    - が、表示されれば問題ないです。
    - もし、表示されなかった場合は、Slackにてお知らせください。

プラットフォームの利用について

- 質問 : 課題のソースはダウンロードできますか。

    - はい、ダウンロードは可能です。

- 質問 : ログインしたあとに手順書とは違う画面がでます。

    - 一度、GitHubのアカウントをログアウト→再ログインして手順をやり直してみてください。
    - 緑色の「GitHubクラスルームを承認する」ボタンのある画面が表示されると思いますので、その緑色のボタンを押下してください。

- 質問 : 課題の期限を教えてください。

    - 期限は実証期間が終了する金曜までとなります。

- 質問 : githubの画面に赤い×が表示されていますが、これはどうしたら直りますか？

    - 課題を提出後(コミット&プッシュ)にエラーがなければ緑のチェックマークに変わります。

チーム開発資料

アジェンダ

- 実習概要
- チーム開発手法
- チーム開発ルール
- 評価方法
- チーム開発実施方法
  - 対面
  - 遠隔（Live Share使用方法）

実習概要

- 概要
  - 対面と遠隔のそれぞれで、教育PFを使ったチーム開発をモブプロで行い、同等の結果が得られるかの検証を行う

- 実習内容
  - チーム単位でSpringBootを使ったプログラム作成を行う演習課題を実施
    - 演習課題は、実証実験と同じ（学生向けportal画面の開発）ものを使用
  - 演習課題は**モブプロ**(※)にて行う
  - 対象の富山の学生（1年生）40名
  - 対面と遠隔で20名ずつに分け、その中で1チーム4人を編成
    （対面：5チーム、遠隔：5チーム）
- スケジュール
  - 遠隔
    - 12/12(PM1コマ 教室C301)：準備、課題の説明、各自でLevel0の内容を確認、Live Shareを使ったモブプロ体験
    - 12/13：各自でLevel1と2を実施
    - 12/14-16：3時間 level3（モブプロ）
  - 対面
    - 12/13(AM2コマ 教室C301)：準備、説明、各自でLevel0-2を実施
    - 12/14(AM2コマ教室C301)：3時間 Level3 （モブプロ）

チーム開発手法

今回のチーム開発ではモブプロといった手法をベースに開発を進める。

モブプログラミング（モブプロ）：

モブプログラミングとは１つのプログラムを複数人でコミュニケーションを取りながら実装を進め、知識をチーム全体で共有しながらモブ（開発の議論や実装を考える人）とタイピスト（モブの指示でコードを書く人）に分かれて行う開発手法。

また、2人で実施する場合をペアプログラミング（ペアプロ）という。

チーム開発ルール

- 今回はチーム(4人)ごとに必要な役職（プログラマー、ナビゲーター、記録係、サポーター）を割り振り、役職のローテーションを行いながら、時間を区切ってコーディングを行う。
- 10分で交代
  （課題の切り替え、環境構築の時間等は除く。時間についてはチーム内で変更可能）

チーム開発ルール（役職）

- ドライバー（プログラマー）
  - ナビゲーターの指示を受けてコードを書き進める。
- ナビゲーター
  - プログラマーに指示を出す役割。コーディングの進行をメインで進めていく。
- サポーター
  - ナビゲーターがつまるところなど適宜アドバイスを行う。
- 記録係
  - 各順の進捗や議論時間についての記録を行います。主にナビゲーターの評価となる。
  - 合わせてタイムキーパーも行います。10分計り、時間が来たら声掛けを行う。

チーム開発ルール（各ターム手順）

- 各課題初回のみ：
  遠隔：ホスト（リーダー）がCodespaseを立ち上げ、Live Shareでソースを共有する。
  対面：リーダーがCodespaseを立ち上げ、共有のディスプレイに表示する。

- ナビゲーターが主に指示を出しながら、ドライバー（プログラマー）にプログラミングを進めてもらう。
- 記録係はタイムキーパーとして時間を測る。
- 必要に応じてドライバー（プログラマー）の画面共有を行うとよい。

↓

- 時間終了後、記録係が声掛け。
- ホスト（リーダー）はコミットを行う。
- 記録係は評価シートに記録を行う。

↓

- 役職を回し、次のタームに進む。

評価表への記載方法

- チーム開発評価スプレッドシートを各チームに配布。
- 日付ごとに１シート使用。（シートをコピーして使用。）
- 日付、メンバー名と初回の役割分担を入力する。
  - 役割のローテーションは１巡目を入力すると、２巡目以降は自動で割り振りされる。
- 各順ごとに、記録係の人が各順の実施時間や評価項目に結果を記載する。

			1巡目
	時間（○分）		10分
役割	ナビゲーター（メインの指示を出す人）		田中
	ドライバー（プログラムを書く人）		鈴木
	サポーター		山本
	サポーター・記録係		渡辺
記録項目	ソースに関する議論がなされていた時間(割合)		90%
	ホストはコミットを行ったか		○
	課題の進捗	開始地点	level3-1
		終了地点	level3-1
	その他コメントがあれば記入		

評価法記載方法

日付	2022年12月12日						
		1巡目	2巡目	3巡目	4巡目	5巡目	6巡目
	時間（〇分）	10分	10分	10分	10分		
役割	ナビゲーター（メインの指示を出す人）	田中	渡辺	山本	鈴木	田中	渡辺
	ドライバー（プログラムを書く人）	鈴木	田中	渡辺	山本	鈴木	田中
	サポーター	山本	鈴木	田中	渡辺	山本	鈴木
	サポーター・記録係	渡辺	山本	鈴木	田中	渡辺	山本
記録項目	ソースに関する議論がなされていた時間(割合)	90%	100%	100%	80%		
	ホストはコミットを行ったか	○	○	○	○		
	課題の進捗 開始地点	level3-1	level3-1	level3-2	level3-3		
	課題の進捗 終了地点	level3-1	level3-2	level3-3	level3-3		
	その他コメントがあれば記入						

チーム開発実施方法

- 対面
  - 各チームごとに1枚ディスプレイを用意。リーダの作業PCの内容を映し出しながら、全員で同時に確認できるようにする。
  - 各タームごとに席替えを行い、PCを使いまわす形で操作を行う。

- 遠隔
  - Visual Studio Code の Live Share という機能用いて、リーダの作業PCの内容を共有および同時操作できる状態で実施する。（Live Shareの使用方法は次ページから説明）

チーム開発実施方法（遠隔：Live Ahare）

Live Shareを使ったチーム開発

概要

- Live Shareでは複数人でソースを共有し、同時に編集することができる。
- ホスト側（開発環境を立ち上げ、招待する側）とゲスト側（参加する側）に分かれ、ホストは自分の環境にゲストを招待する。

招待手順

1. ホスト：Live Share起動
2. ホスト: url共有
3. ゲスト: sessionに参加
4. ゲスト: GitHubログイン
5. ホスト・ゲスト: Live Share起動確認

チーム開発実施方法（遠隔：Live Ahare）

Live Shareを使ったチーム開発

概要

- Live Shareでは複数人でソースを共有し、同時に編集することができる。
- ホスト側（開発環境を立ち上げ、招待する側）とゲスト側（参加する側）に分かれ、ホストは自分の環境にゲストを招待する。

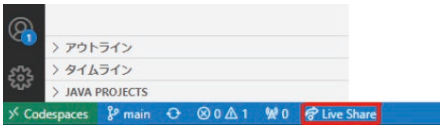
招待手順

1. ホスト：Live Share起動
2. ホスト: url共有
3. ゲスト: sessionに参加
4. ゲスト: GitHubログイン
5. ホスト・ゲスト: Live Share起動確認

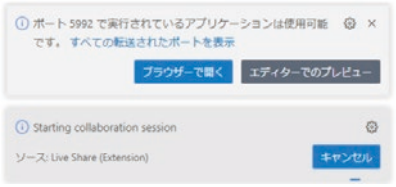
チーム開発実施方法（遠隔：Live Ahare）

1. ホスト：Live Share起動

- ホスト（招待する人）のpcでCodespace起動後、左下のLive Shareをクリック。



- 右下に起動中のアナウンスStarting collaboration sessionが表示。



- 右下にInvitation link copied to clipboard! ～が表示されると準備完了。

チーム開発実施方法（遠隔：Live Ahare）

2. ホスト: url共有

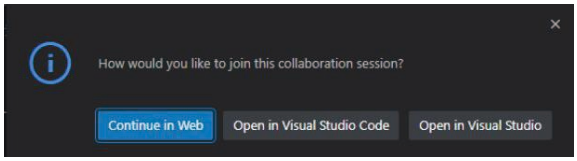
1の手順終了後、すでにpcには招待用URLがコピーされているため、slackなどに貼り付け、メンバーにURLを共有する。

（コピーに失敗した場合はCopy againをクリックするとよい。）

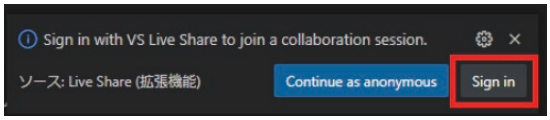
チーム開発実施方法（遠隔：Live Ahare）

3. ゲスト: sessionに参加

- ゲスト（参加メンバー）は共有されたURLをクリックすると、ブラウザでVS Codeが立ち上がる。次のようなポップアップが表示された場合、Continue in Webを選択すること。



- 右下に、Sign in with VS Live Share to a collaboration session.が表示されるので、Sign in をクリックする。

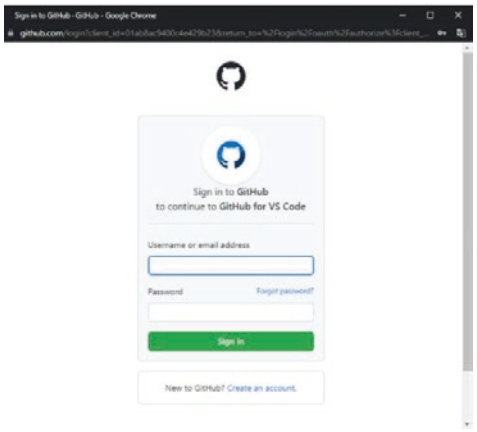


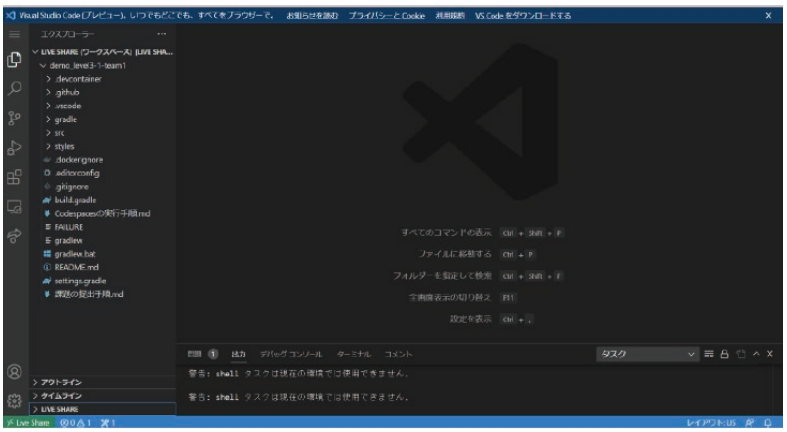
- 上部にサインインするサービスの選択が現れるので、GitHubを選択します。

チーム開発実施方法（遠隔：Live Ahare）

4. ゲスト: GitHubログイン

GitHubログイン用のウィンドウが表示され、作成したアカウントを入力し、認証に進む。
ログイン後、しばらく時間をおいた後にブラウザのVSCodeの準備が完了となる。

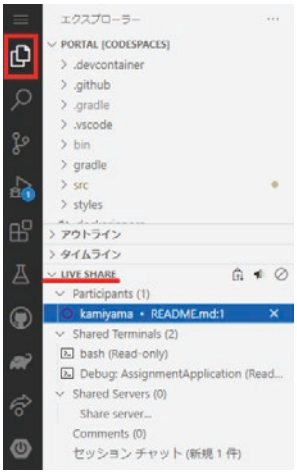




チーム開発実施方法（遠隔：Live Ahare）

5. ホスト・ゲスト: Live Share参加者確認

エクスプローラー（表示されていない場合は左上ファイルボタンをクリック）の下部にLIVE SHAREタブが現れる。
Participantsに表示されるのが参加中のユーザーのため、
全員参加しているか確認する。



チーム開発方法

今回のチーム開発ではモブプロ・ペアプロといった手法を用いて開発を進めます。
詳細： ***https://qiita.com/mzmz__02/items/26a34e3e3f2cb870cfa2***

◎ルール
・チーム(4人)ごとに必要な役職（プログラマー、ナビゲーター、記録係、サポーター）を割り振り
・役職のローテーションを行いながら、時間を区切ってコーディングを行います。
・10分で交代（課題の切り替え、環境構築の時間等は除く。時間についてはチーム内で変更可能）

◎役職
・ドライバー（プログラマー）
***　ナビゲーターの指示を受けてコードを書き進めていきます。***
・ナビゲーター
***　プログラマーに指示を出す役割。コーディングの進行をメインで進めていくことになります。***
・サポーター
***　ナビゲーターがつまるところなど適宜アドバイスをします。***
・記録係
***　各順の進捗や議論時間についての記録を行います。主にナビゲーターの評価になります。***
***　合わせてタイムキーパーも行います。10分計り、時間が来たら声掛けをお願いします。***

◎各タームでの手順
各課題初回のみ：ホスト（リーダー）がCodespasesを立ち上げ、Live Shareでソースを共有する。（チーム開発方法の資料参照）

ナビゲーターが主に指示を出しながら、**ドライバー**（プログラマー）にプログラミングを進めてもらう。
記録係はタイムキーパーとして時間を図る
必要に応じて**ドライバー（プログラマー）**の画面共有を行うとよい
↓
時間終了後、**記録係**が声掛け
ホスト（リーダー）はコミットを行う。（チーム開発方法の資料参照）
記録係は評価シートに記録を行う。
↓
役職を回し、次のタームに進む

チーム開発評価シート

日付ごとに１シート使用します。（コピーして使ってください。シート名は日付を記入します。）表の黄色いセル内の項目を埋めていってください。
役割のローテーションは１巡目を入力すると、２巡目以降は自動で割り振りされます。
各順ごとに、記録係の人が各順の時間や評価項目に記載をお願いします。

日付	2022年12月12日

			１巡目	２巡目	３巡目	４巡目	５巡目	６巡目	７巡目	８巡目	９巡目	１０巡目	１１巡目	１２巡目	１３巡目	１４巡目	１５巡目	１６巡目	１７巡目	１８巡目	１９巡目	２０巡目
	時間（〇分）																					
役割	ナビゲーター（メインの指示を出す人）			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	ドライバー（プログラムを書く人）			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	サポーター			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	サポーター・記録係			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
記録項目	ソースに関する議論がなされていた時間(割合)																					
	ホストはコミットを行ったか																					
	課題の進捗	開始地点																				
		終了地点																				
	その他コメントがあれば記入																					

チーム開発評価シート

（使用例）

日付ごとに１シート使用します。（コピーして使ってください。シート名は日付を記入します。）表の黄色いセル内の項目を埋めていってください。
役割のローテーションは１巡目を入力すると、２巡目以降は自動で割り振りされます。
各順ごとに、記録係の人が各順の時間や評価項目に記載をお願いします。

日付	2022年12月12日

			１巡目	２巡目	３巡目	４巡目	５巡目
	時間（〇分）		10分	10分	10分	10分	
役割	ナビゲーター（メインの指示を出す人）		田中	渡辺	山本	鈴木	田中
	ドライバー（プログラムを書く人）		鈴木	田中	渡辺	山本	鈴木
	サポーター		山本	鈴木	田中	渡辺	山本
	サポーター・記録係		渡辺	山本	鈴木	田中	渡辺
記録項目	ソースに関する議論がなされていた時間(割合）		90%	100%	100%	80%	
	ホストはコミットを行ったか		○	○			
	課題の進捗	開始地点	level3-1	level3-1	level3-2	level3-3	
		終了地点	level3-1	level3-2	level3-3	level3-3	
	その他コメントがあれば記入						

実証講座

１．演習ユニットと学習管理ユニットの実証講座

（１）目的

令和 4 年度に開発した教材およびプラットフォームを検証する。

（２）実施方法

3 校の専門学校で Java 言語の基礎を習得している学生を対象として検証した。実証前に、委員 2 名程度を対象校に派遣して、学習環境確認および学生に対する説明会を開催した。学生は各自の好きな時間に学習した。学習時間は 3〜5 日程度とした。

学校	学生	期間
A 専門学校	情報系学生 26 名	10 月 24 日〜10 月 27 日
B 専門学校	情報系学生 16 名	11 月 14 日〜11 月 18 日
C 専門学校	情報系学生 44 名	11 月 28 日〜12 月 2 日

（３）実施結果

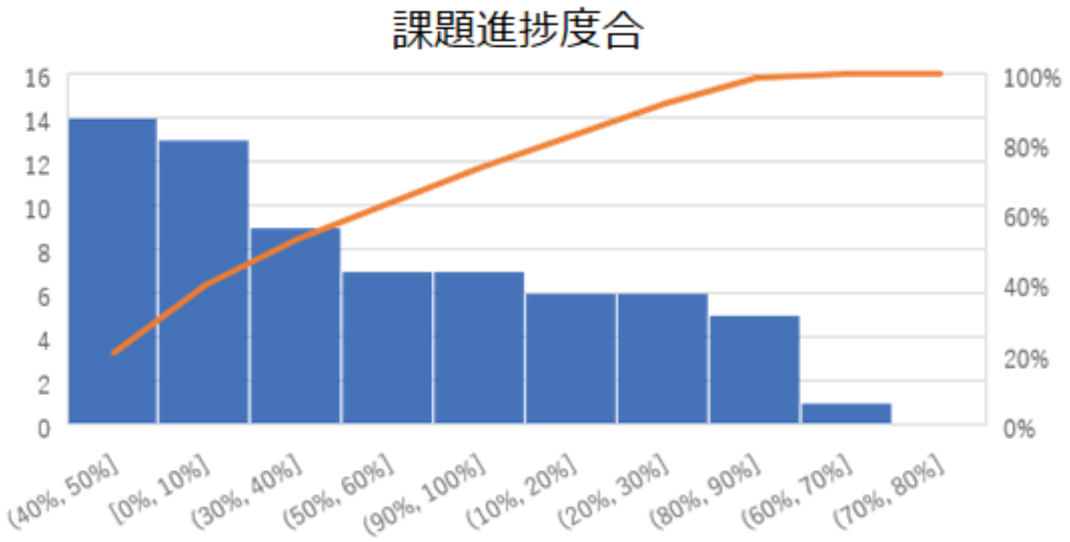
課題の進捗結果

（ａ）3 校全体

・対象者

実証実験用に払い出したアカウントで有効 かつ、実証実験後のアンケートに回答があった学生を対象とした。

A 専門学校 26 名、B 専門学校 15 名、C 専門学校 27 名

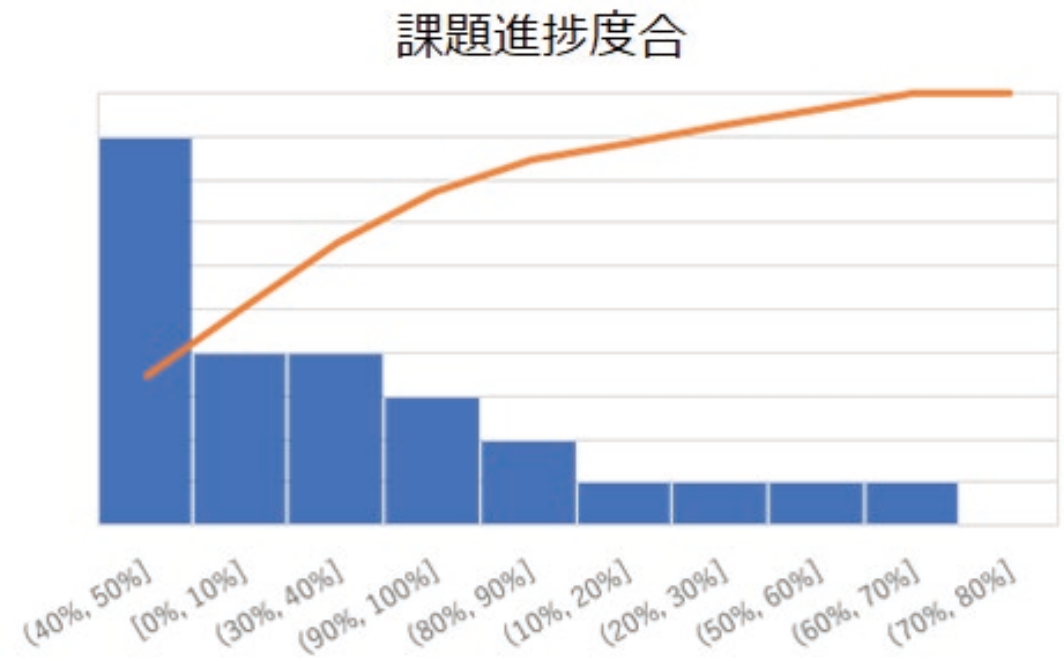


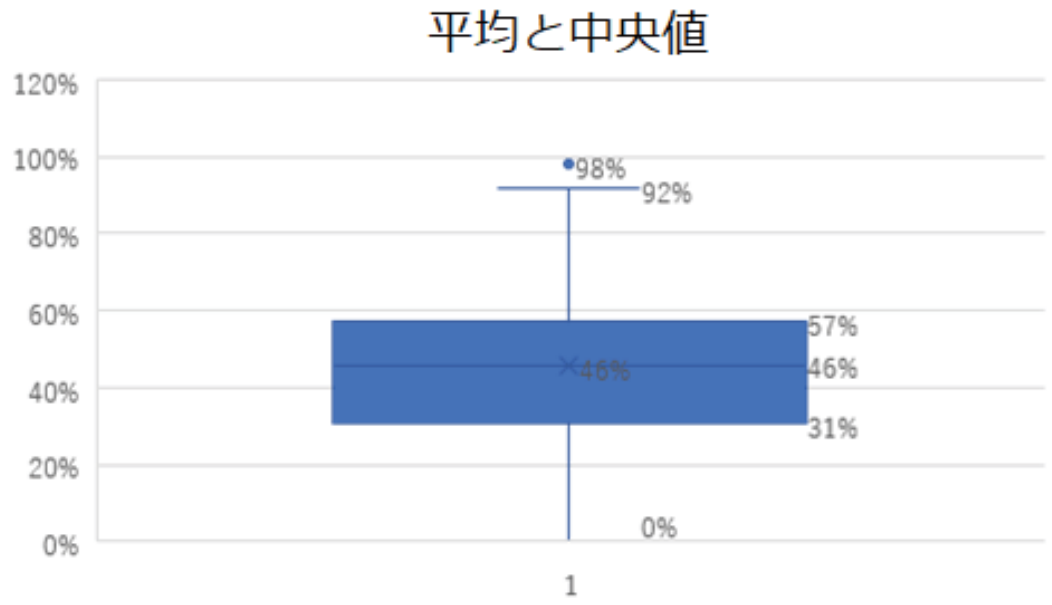
40-50% （Level3-1 まで完了）の進捗が最も多かった 多くの学生が Level3-1 で頭打ちになった

（ｂ）A 専門学校

・対象者

26 名



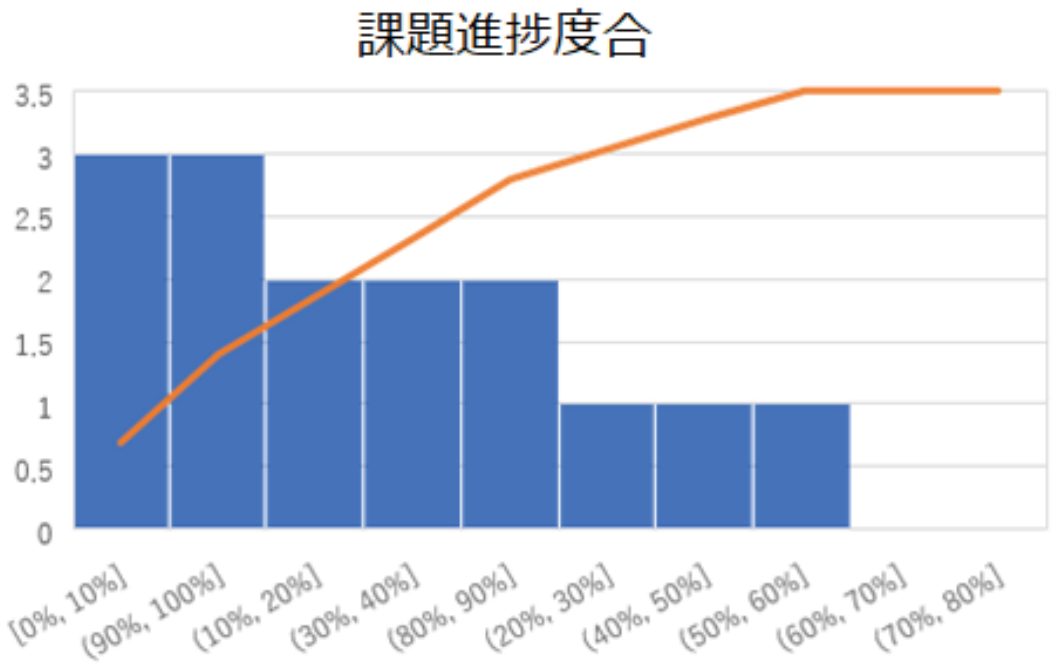


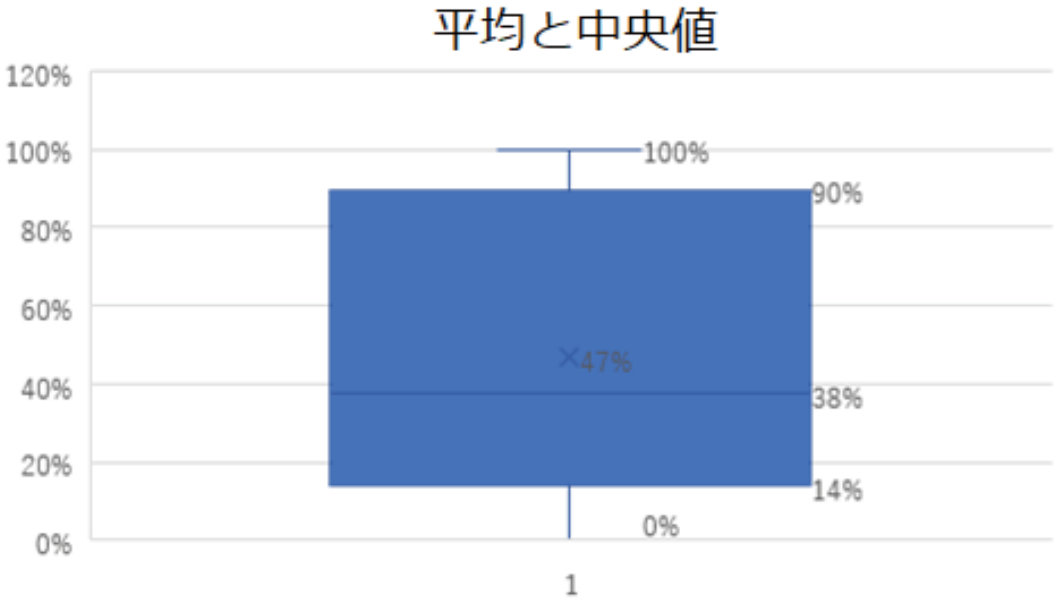
平均：４６％　中央値：４６％

（ｃ）Ｂ専門学校

・対象者

15 名



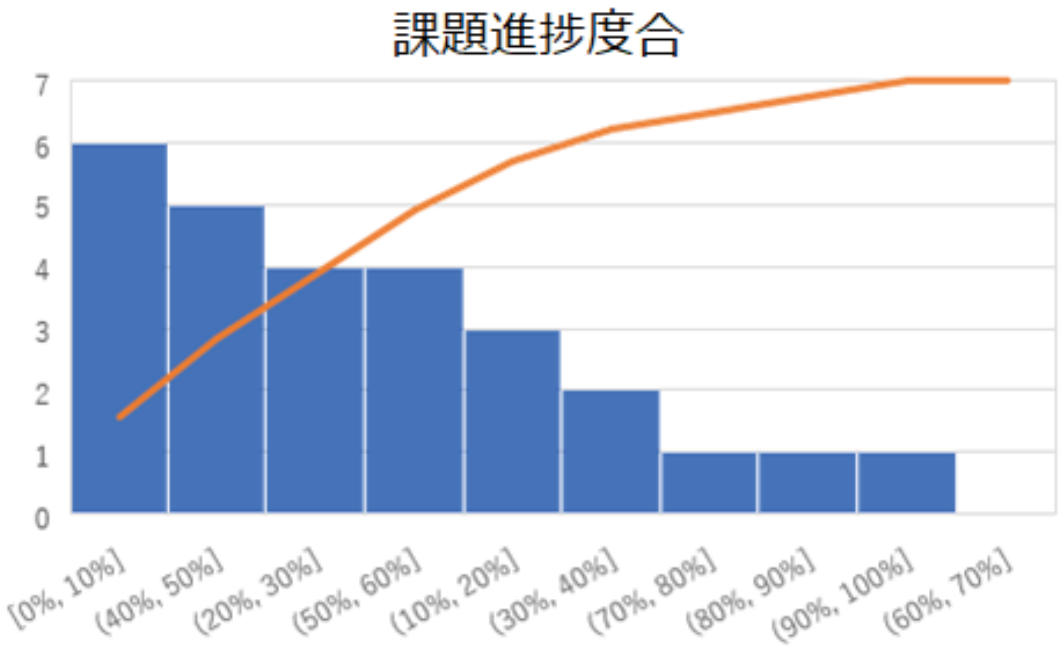


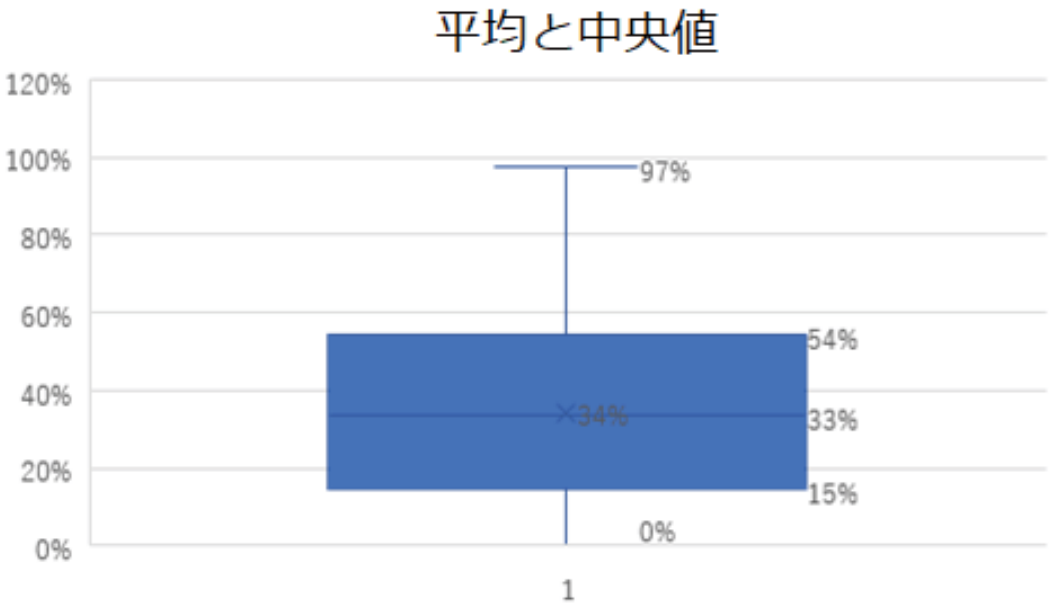
平均：４７％　中央値：３８％

（ｄ）Ｃ専門学校

・対象者

27 名





平均：３４％　中央値：３３％

（ｅ）学生からのＱ&Ａとチャットボットの利用について

・Ａ校

32件（うちチャットボット０件）

・Ｂ校

９件（うちチャットボット０件）

・Ｃ校

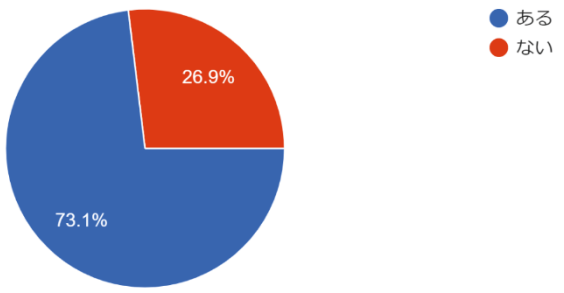
６件（うちチャットボット２件）

（４）実証後アンケート結果

（ａ）A 専門学校

今回の授業の以前にGitHub（またはGitを使ったサービス）を利用したことはありましたか？

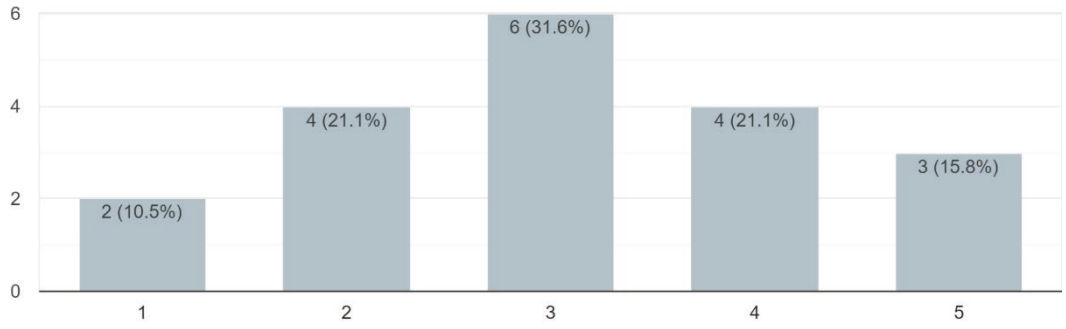
26 件の回答



利用頻度

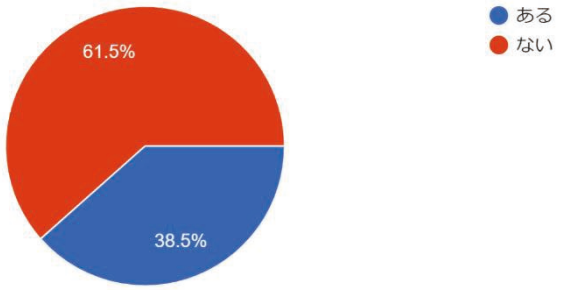
上記であると選択された方はご回答ください。

19 件の回答



今回の授業の以前にSlackを利用したことはありましたか？

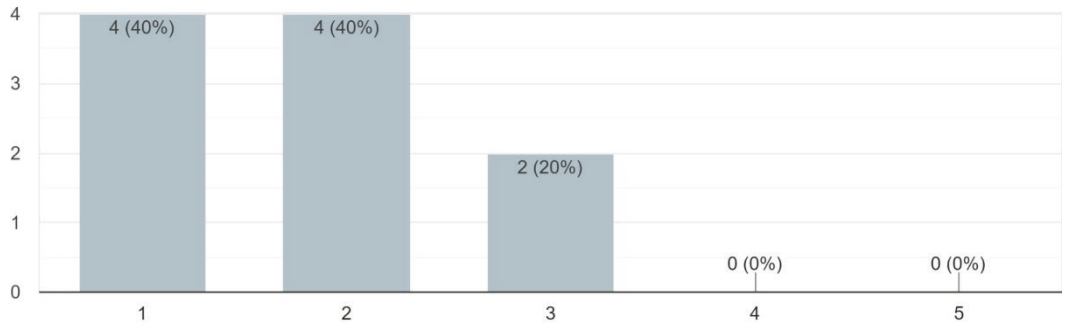
26 件の回答



利用頻度

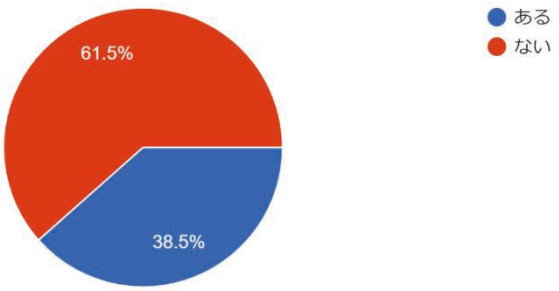
上記であると選択された方はご回答ください。

10 件の回答



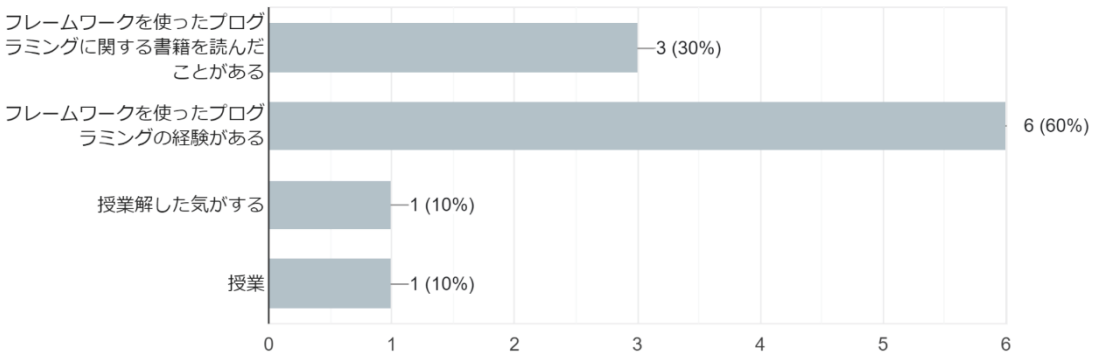
今回の授業の以前にフレームワークに関する知識または経験はありましたか？

26 件の回答

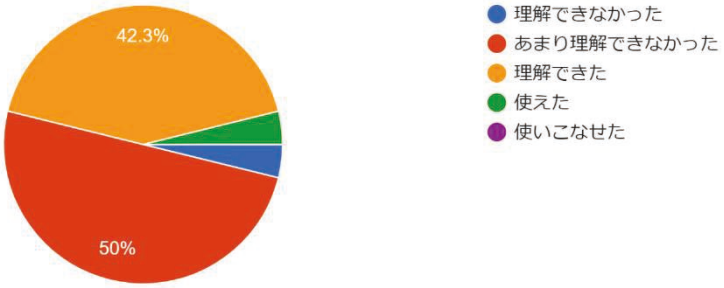


上記であると選択された方はご回答ください。

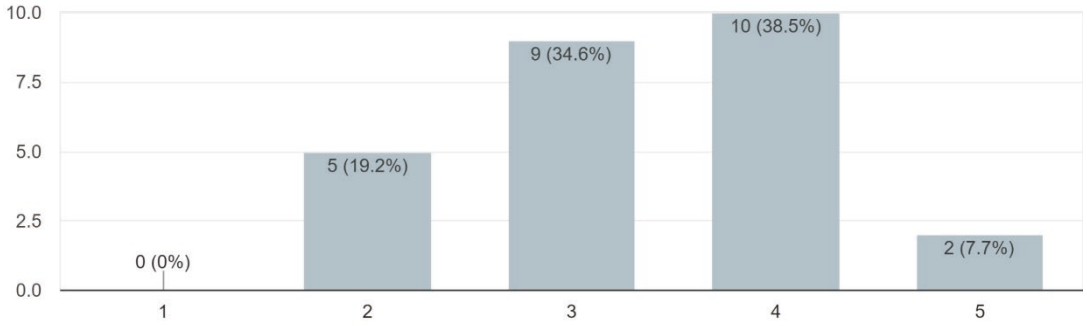
10 件の回答



今回の授業内容である「フレームワークを使ったプログラミング」についてご回答ください。
26 件の回答



フレームワークを使ったプログラミングの教材は適切でしたか？
26 件の回答

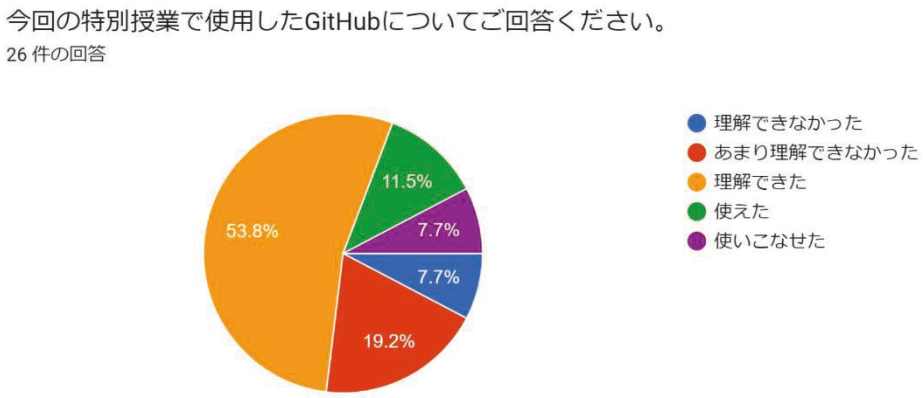


フレームワークをつかったプログラミングの授業について良かった点

- 一から作るよりスムーズにコードを書けて時間を節約できた
- コーディングがしやすかった
- 一部を直したり、そこに追加していったりするのでとてもやりやすかったです
- 簡単にアプリケーションが作れ、成果物がつくれる点はよかったです。
- 多くの場所を変更しなくても何が起こってるかわかりやすかった点

フレームワークをつかったプログラミングの授業について改善点

- lesson のを真似したらあまり理解しなくても作ることができました。なので 1 から作るほうがいいと思います。
- セッティングに時間がかかり、課題に取り組む時間が少なくなる
- 今回の授業ですと Java がメインでしたので Ruby だと Ruby on Rails,PHP だと Laravel のように複数のフレームワークを自分で選んで授業を受けれる環境だとなお良いとおもいました。
- どこからどこまでがフレームワークによる働きかわからなかった
- プログラムが得意ではない人はこれを見て難しそうという固定概念で断念すると思う。



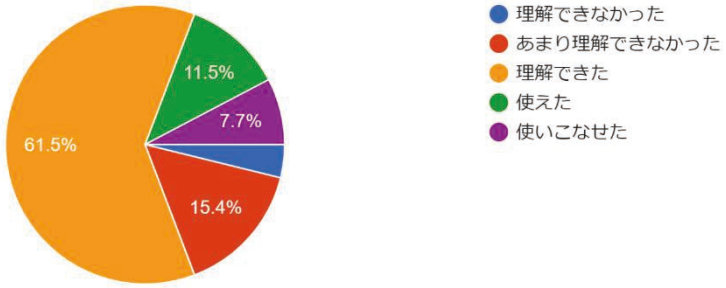
GitHub について良かった点

- URL にとぶとすぐに開かれて使えるので便利だった
- 保存したデータがとりやすかったです
- 変更点がわかりやすく見えてよかったです。
- UI がシンプルで分かりやすかった
- データの共有がしやすいのが便利だと思った。

GitHub について改善点

- 英語だらけでパッと見て分かりずらかった
- 課題のチェックに時間がかかる
- 今回は個人作業だったため仕方ないが複数人で作業するのに便利だという利点が感じづらかった
- ファイルの指定が分かりにくかった。

今回の特別授業で使用したCodespaces(クラウドのVSCode)についてご回答ください。
26 件の回答



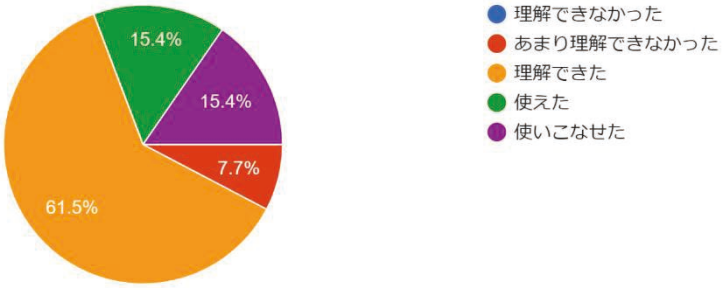
Codespaces について良かった点

- vscode と同じような感じだったので使いやすい
- 縮小などが出来るので見やすかった
- どこに何が入ってるか見つけやすい構造でした
- 保存しなくても自動で保存してくれるので使いやすかったです。
- インストール不要で開発環境が使えるのは便利だと思った。
- ソースアクションがとても便利だった
- Commit するときの差分がわかりやすかった"
- 持ってる PC のスペックにあまり関係ないことがよかった

Codespaces について改善点

- 生成できる数に制限がある
- 時間によりますが場合によって 5〜20 分ほど待たされるのがストレスに感じました。
- 全体的に動作が遅い
- 間違ってブラウザを閉じたりした時にデータが消えたりしたのでローカルでする方がいいなと思った。
- ネット環境が悪いと、データが飛んだりしていた点。

今回の特別授業で使用したSlackについてご回答ください。
26 件の回答



Slack について良かった点

- やり取りが slack で完結する
- チャンネルを切り替えられる
- 分からないことや、注意事項などをすぐ読むことが出来て良かった
- メールと違い情報の共有が簡単そうでよかったです。
- 通知が来るので課題を見落とすことがない点
- UI がわかりやすく使いやすい
- 質問がしやすかった。

Slack について改善点

- 全員がチャットを見ることができるので教師と生徒の 2 人で対話する所がほしかった
- コメントされると問題のリンクが上の方に行って探すのがめんどくさい
- 言葉ではなく文字で伝えないといけないのが難しい。
- 質問を文やスクショで伝えるので伝えずらい
- アカウントの変え方を簡単にしてほしい

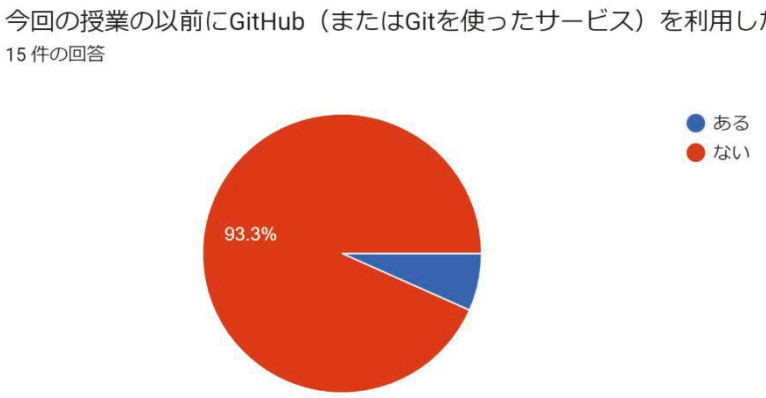
普段の対面での授業と今回の遠隔での授業を比べてみて、良かった点

- 一人で黙々と集中してできた
- 遠隔だと自分で調べてやるしかないので調べる能力がついた
- 画像や文字で文章を送ってコミュニケーションするので、対面とはまた別の感覚でしたが、コメントで質問ができるので、わざわざ手を上げなくて済みました"
- PC 1 つあれば授業が受けれるのでなにか病気などしたときに家でも授業が受けれそうなとこがよかったです。
- 普段の授業と違って今回の授業は自分のペースで進められるから嬉しかったです
- タイミングを気にせずチャットで質問を投げれる。
- 自分で考えてできること。

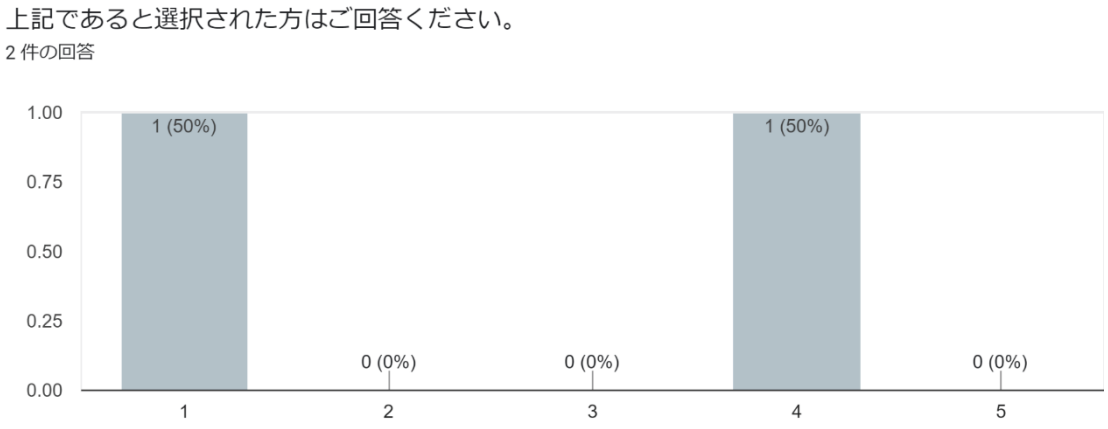
普段の対面での授業と今回の遠隔での授業を比べてみて、悪かった点

- トラブルを解消するまでに時間がかかりやすい
- すぐ質問したいときに slack を開かないといけないし、写真と文字での説明なのでいつもより時間がかかった
- 完成像が想像つかないため細かい部分がわからない"
- 全体質問だからちょっとしにくかった

（ｂ）B 専門学校

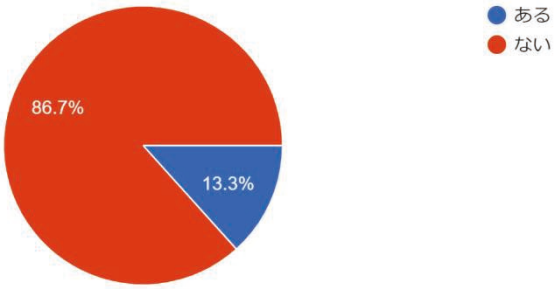

今回の授業の以前にGitHub（またはGitを使ったサービス）を利用したことはありましたか？
15 件の回答
ある
ない
93.3%

利用頻度


上記であると選択された方はご回答ください。
2 件の回答
1.00
0.75
0.50
0.25
0.00
1 (50%)
0 (0%)
0 (0%)
1 (50%)
0 (0%)
1
2
3
4
5

今回の授業の以前にSlackを利用したことはありましたか？

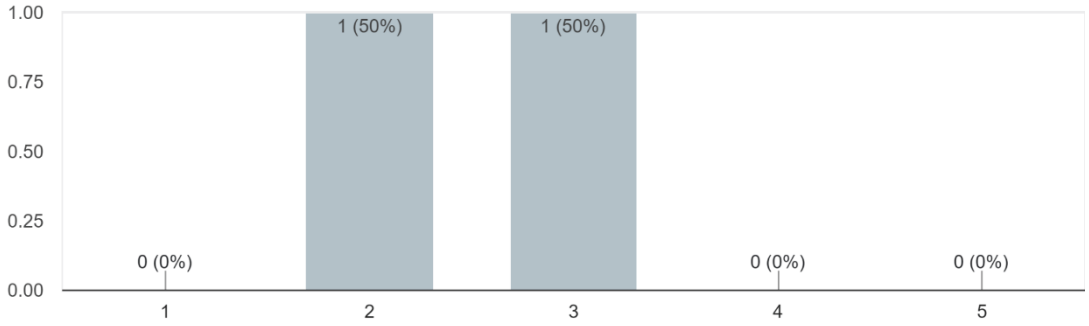
15 件の回答



利用頻度

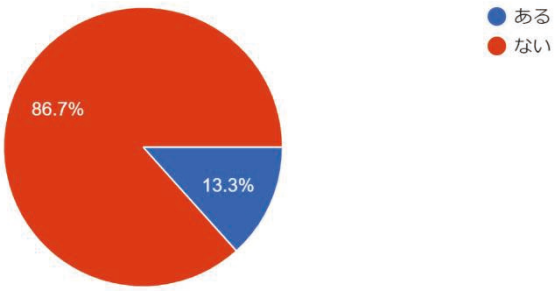
上記であると選択された方はご回答ください。

2 件の回答



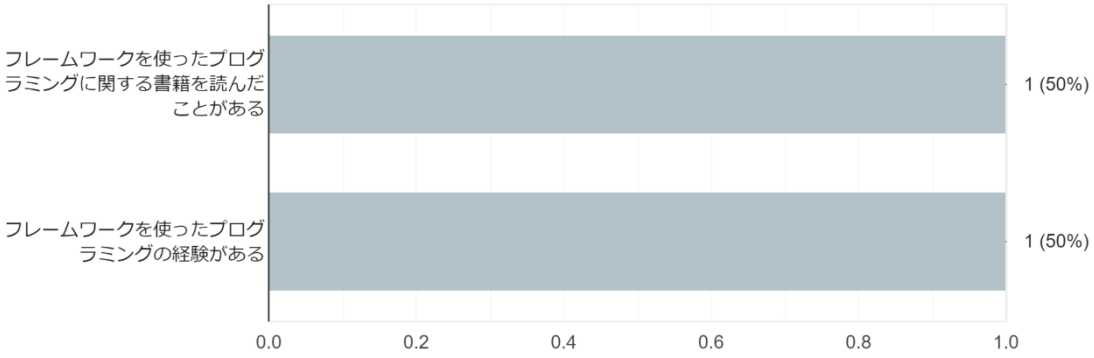
今回の授業の以前にフレームワークに関する知識または経験はありましたか？

15 件の回答



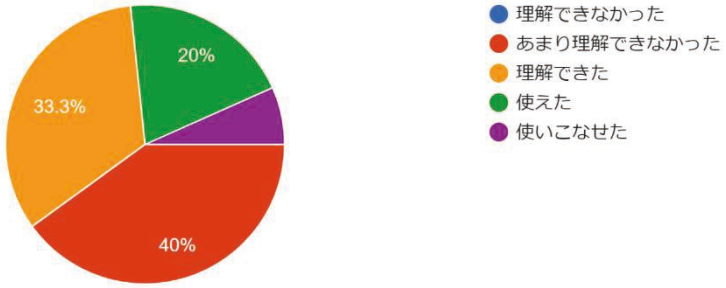
上記であると選択された方はご回答ください。

2 件の回答



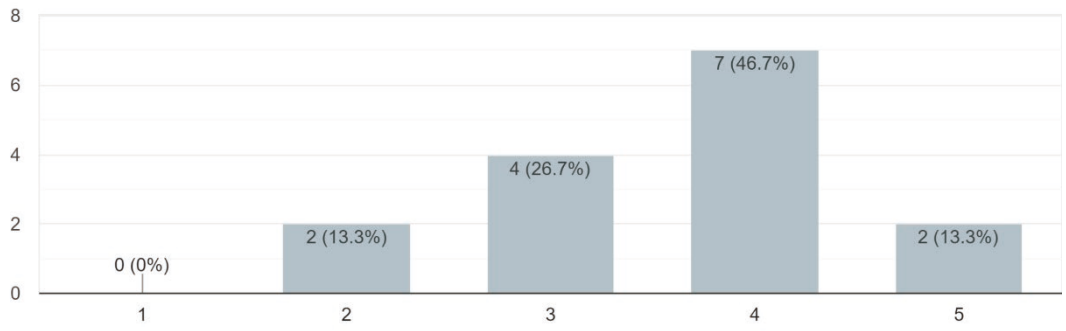
今回の授業内容である「フレームワークを使ったプログラミング」についてご回答ください。

15 件の回答



フレームワークを使ったプログラミングの教材は適切でしたか？

15 件の回答

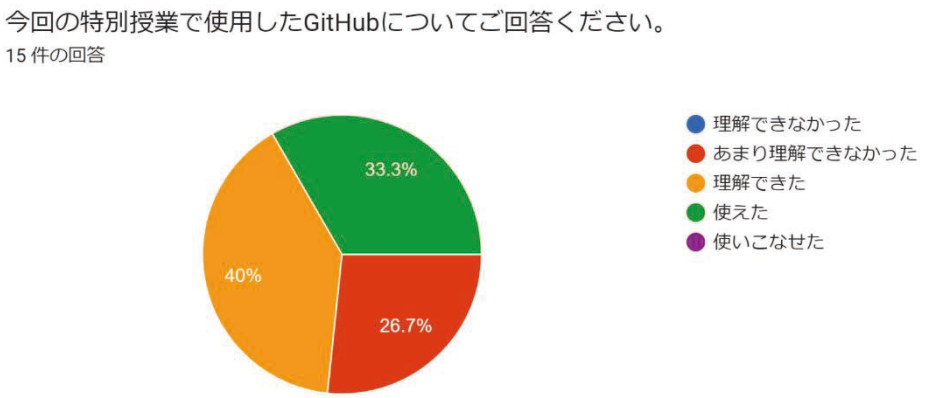


フレームワークをつかったプログラミングの授業について良かった点

- 「java bronze で出た内容を実際にコードに起こすとこんな感じになるんだな」というような感じで理解を促進させることができたので助かりました。
- １から自分でコードを書かなくて良く、必要な部分だけを変えることができる点
- 個人的に理解しにくいと感じていたインターフェースのことを実践手な環境で見れたのはとても良かった
- 実際の現場で使われているものを使うことで今後に生かすことができる。
- 書き方が分からなくなってもフレームワークを参考にサクサク書けるので、分からなくなってモチベーションの低下というのが無かったです。

フレームワークをつかったプログラミングの授業について改善点

- フレームワークを常に見てプログラムを作成していたので、本当に勉強になっているのか少し不安でした。



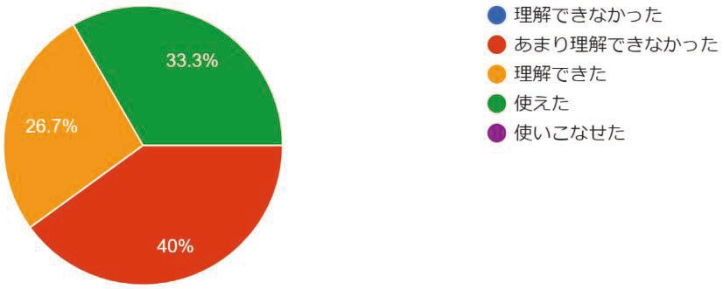
GitHub について良かった点

- ベースとなるプログラムを残したままコード等を変更、追加できること
- 実際の現場で使われているものを使うことで今後に生かすことができる。

GitHub について改善点

- 基本言語が英語だったので日本語で作業できるといいと思った。
- pc の性能によっては、まともに起動出来ず、何回も再起動、エラー処理が起きる点。
- 使い方が分からなくなった時用に github の使い方に関しての質問チャンネルなども欲しいです。

今回の特別授業で使用したCodespaces(クラウドのVSCode)についてご回答ください。
15 件の回答



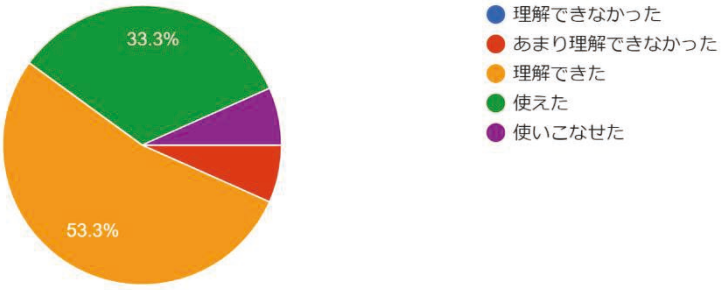
Codespaces について良かった点

- 拡張機能で様々な要素を追加できる点と README なども開ける点がとてもよかったと感じた。
- ローカル上に環境を作成する必要がないこと。特にローカルの git のアカウントを切り替える必要がないこと。
- どこでもブラウザさえあれば環境を構築できるのは強みだと感じました。
- 進捗状況も自動で共有されることも良いと思います。"

Codespaces について改善点

- 一度 commit したら変更を加えてもひとつ前の commit 時点からの変更しか確認できないこと
- エラーが頻繁に発生するので、もっと安定させてほしい。
- リポジトリが異なるためか、テーマが初期化されて少し使いずらかったです。

今回の特別授業で使用したSlackについてご回答ください。
15 件の回答



Slack について良かった点

- 普段使用しているコミュニケーションソフトウェアととても似ていて使いやすかった
- ログを簡単にさかのぼれることや、UI が見やすいところがとても使いやすいところだと感じた。
- 提出した課題の問題点があった時にメンションされて、気が付きやすいように感じました。
- 他の方の質問事項の回答から自分の課題のヒントになりました。

Slack について改善点

- 仕様上、仕方がないような気がしますが、システムメッセージがすこし邪魔に感じます。

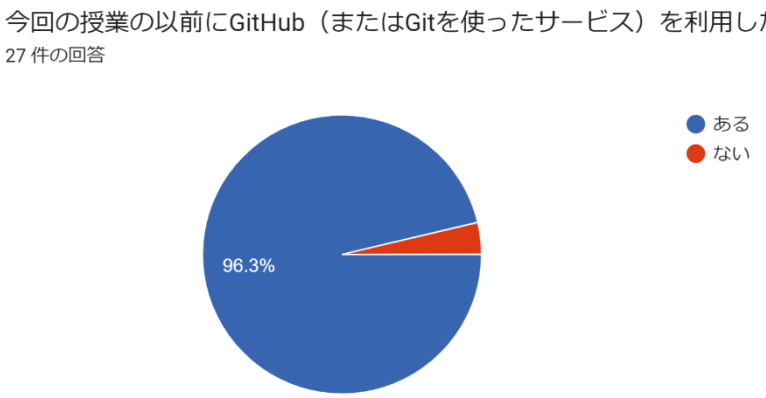
普段の対面での授業と今回の遠隔での授業を比べてみて、良かった点

- 時間帯を問わず質問や課題の提出が可能なので自分で確保した好きな時間に作業できる
- 一人で集中できる
- 空き時間に自分のタイミングで進める事ができるので、ストレスフリー/フレキシブルな学習が行えた

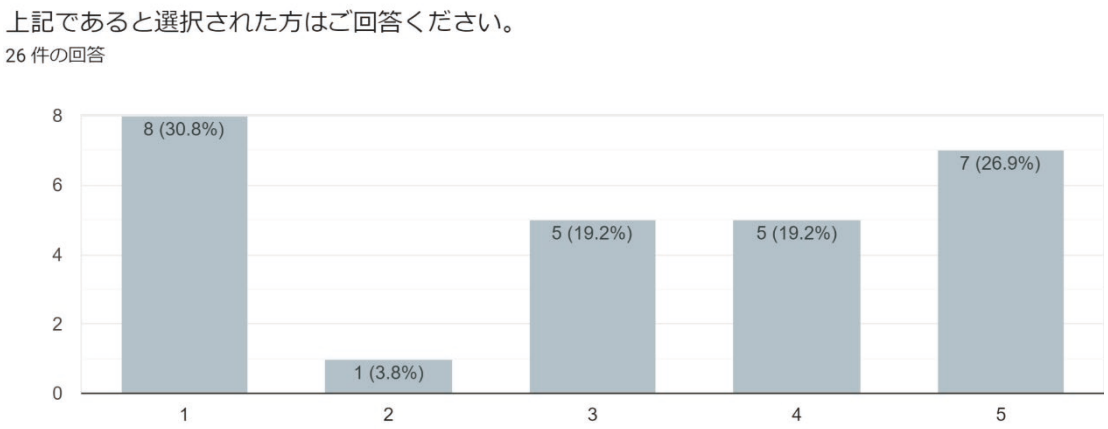
普段の対面での授業と今回の遠隔での授業を比べてみて、悪かった点

- 個人個人で作業を進めているので、失敗しやすいことや操作方法など教師には心理的に聞きずらいことの生徒間での共有や相談ができなかったこと
- 遠隔なので質問をするときに何らかのツールを挟まなければならないところが少し大変に感じた。

（ｃ）C 専門学校

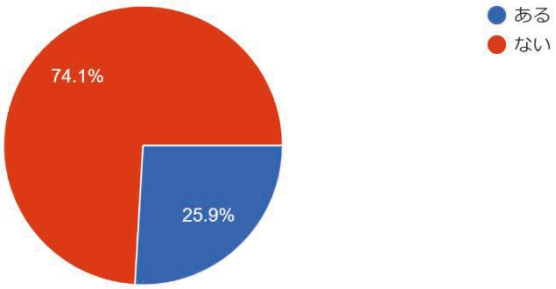

今回の授業の以前にGitHub（またはGitを使ったサービス）を利用したことはありましたか？
27 件の回答
ある
ない
96.3%

利用頻度


上記であると選択された方はご回答ください。
26 件の回答
8
6
4
2
0
8 (30.8%)
1 (3.8%)
5 (19.2%)
5 (19.2%)
7 (26.9%)
1
2
3
4
5

今回の授業の以前にSlackを利用したことはありましたか？

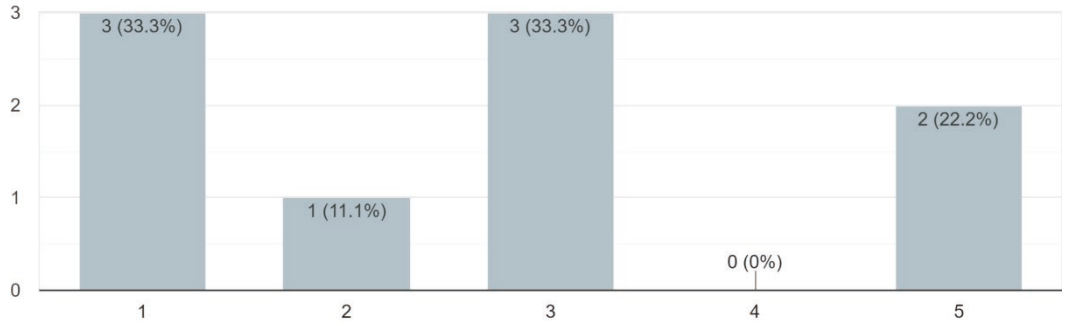
27 件の回答



利用頻度

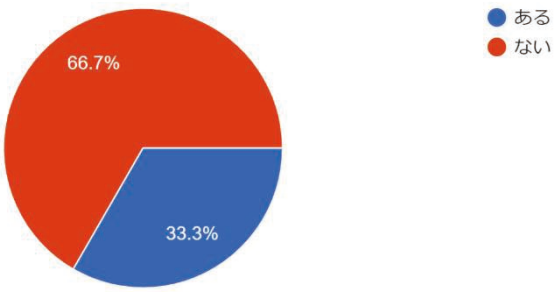
上記であると選択された方はご回答ください。

9 件の回答



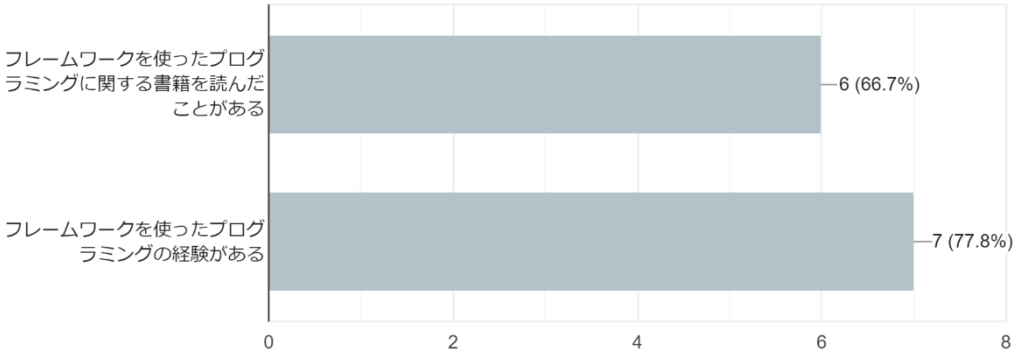
今回の授業の以前にフレームワークに関する知識または経験はありましたか？

27 件の回答



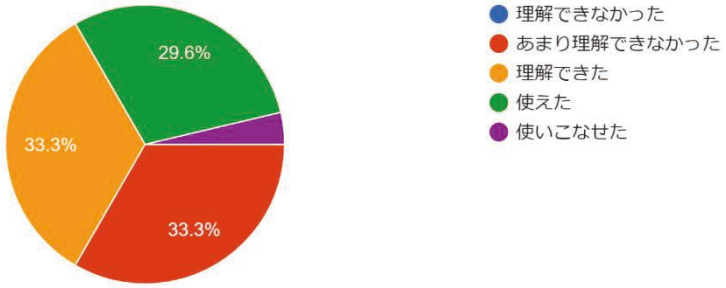
上記であると選択された方はご回答ください。

9 件の回答



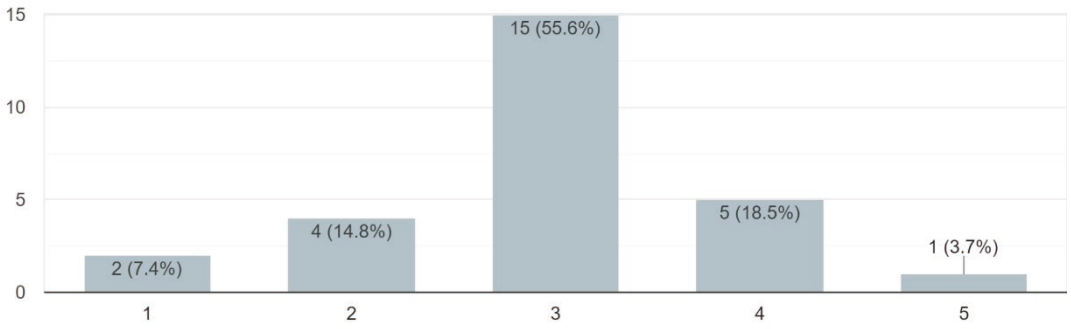
今回の授業内容である「フレームワークを使ったプログラミング」についてご回答ください。

27 件の回答



フレームワークを使ったプログラミングの教材は適切でしたか？

27 件の回答

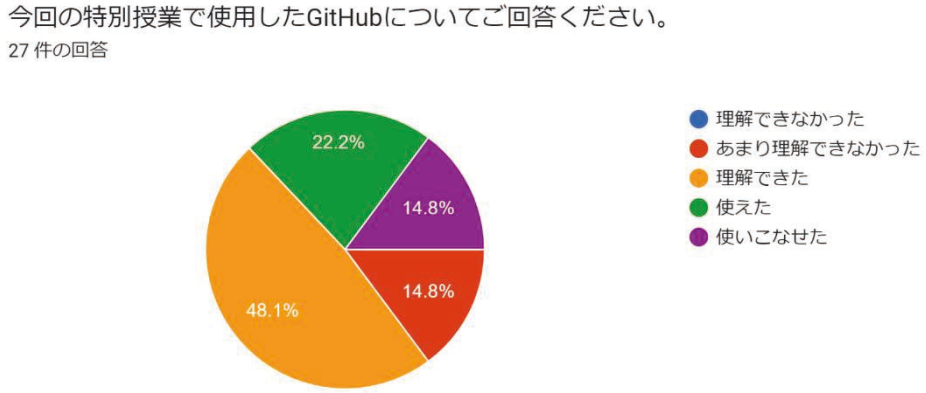


フレームワークをつかったプログラミングの授業について良かった点

- 意味を理解できれば楽にプログラミングすることができた点
- コード書く量が少なく、変更したい部分だけの作業は合理的でよかった

フレームワークをつかったプログラミングの授業について改善点

- 前提知識がないと分からない部分が多い
- クラスの関連やデータの流れを表す図が欲しかった。
- 資料が分かりづらい、このフレームワークを使うメリット等がいまいち説明されていない
- IDE とか使用してコーディング→デバッグが素早くできればよかった。
- ロードが長かった
- 課題内容が見にくかった
- フレームワークの内容を理解するのに時間がかかった点
- どこのファイルを編集するかそのファイルの場所はどこにあるをもっと詳しく書いてくれた方がよかった。

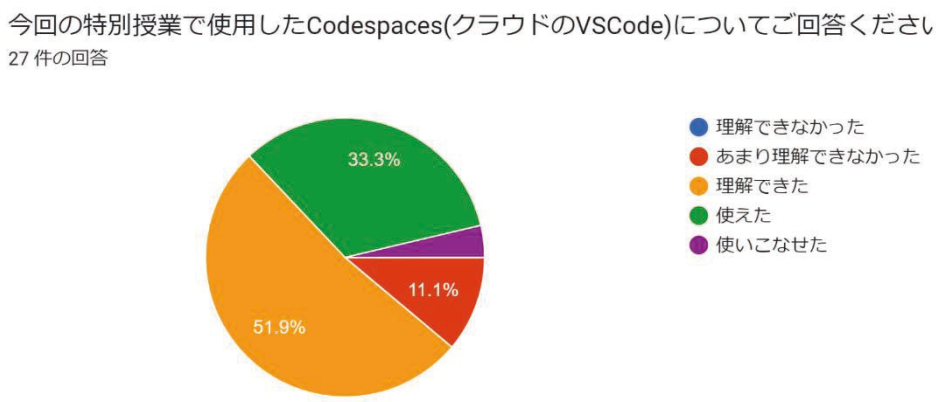


GitHub について良かった点

- 前回からの変更点が分かる
- GitHub に慣れることができる。変更の履歴が残る。
- 簡単にコードが提出できてよかった
- Github バージョン管理がしやすかった。
- コードが素早く共有できた
- チーム開発で非常に便利な点
- 更新をするのが簡単だった
- 使い方がわかれば、すごく便利でよかった

GitHub について改善点

- プルリクエストした方がいいのかどうかが分かりづらかった
- 変更点をコメントで記述しないと何を変更したのかわからない点

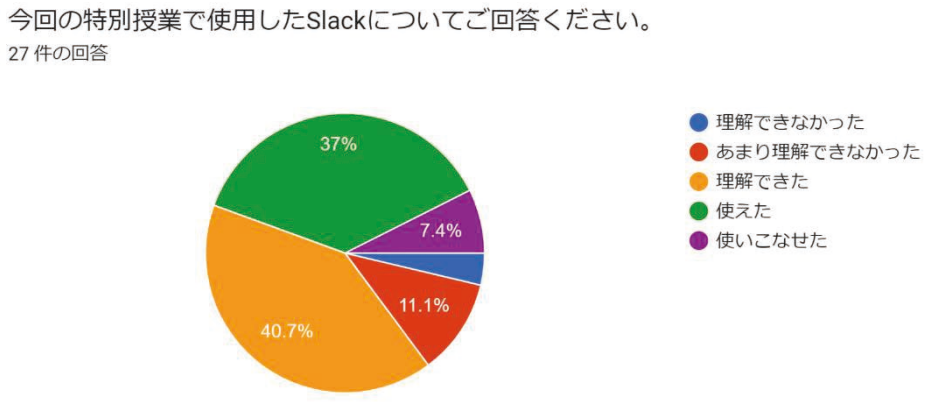


Codespaces について良かった点

- ローカルで環境構築する必要がない
- コミットとプッシュが楽でよかった
- オンラインでコーディングできるのは便利
- プログラミングコードが非常に見やすかった点
- アプリ版とほぼ同じ操作だったのでよかった

Codespaces について改善点

- 動作が重い。起動が遅い。通知が色々きて分かりづらい。
- ファイルの位置などをパス名とかで教えてくれるとよかった



Slack について良かった点

- 気楽に質問出来たりできて便利だった
- ボットに質問ができた
- コミュニケーションが取れやすい
- 誰がチームに参加しているのかわかりやすい点

Slack について改善点

- チャット形式なので過去の発言を振り返るのに時間がかかる
- 通知が多すぎる点

普段の対面での授業と今回の遠隔での授業を比べてみて、良かった点

- 個々のスピードで進められたのがよかった
- オンライン上なので、質問がしやすい点
- 対面での授業では聞き逃したりした場合聞きに行く必要があるが、遠隔だとそれがないためよかった

普段の対面での授業と今回の遠隔での授業を比べてみて、悪かった点

- トラブルがあった際に解決するまで時間がかかる
- 質問しづらい。質問を考えるのが難しい。
- 質問が全体らしく質問を遠慮してしまう
- 対面の説明の方が内容を理解しやすい点
- 分からないことがあったらすぐに質問できないこと。改善点としてはよくある質問はあらかじめ何かにまとめて公開しておくといいきがします

（５）考察

（ａ）課題の進捗について

各学校の進捗平均

A 専門学校（A 校）：46%

B 専門学校（B 校）：47%

C 専門学校（C 校）：34%

C 校の進捗が A 校、B 校と比べて低い結果となった。

（ア）理解度について

理解できなかったと回答

A 校(53%) > B 校(40%) > C 校(33.3%)

進捗

A 校(46%) ≒ B 校(47%) > C 校(34%)

アンケート結果の理解できなっかた人の割合をみたとき、A 校が一番多く、C 校が一番少なかった結果、今回理解度は進捗に影響しなかったと考えられる。

（イ）取り組んだ時間について

授業、自習で取り組んだ時間の平均算出

A 校 ： 18.4 時間

B 校 ： 6.1 時間

C 校：6.2 時間

B 校と C 校の学習時間は変わらないが進捗に差が出ていた

（ウ）学習時の状況

C 校：2 年生 37 名については 2 コマ授業時間で実施

6 時間のうち半分は授業時間**（受動的）**

B 校：授業時間の確保はなし（授業中の自己学習時間の利用は OK）

B 校の学生は能動的に 6 時間を確保し課題を進めていた

この受動的か能動的であったか差が課題への進捗に影響したと推測

能動的 ＝ モチベーションが高い

→ 学習時の環境と状況を確認すると大きく異なっていた

（エ）現地教員の対応

A 校

毎日、開始時と終了時に声掛けを実施

B 校

担当の教員より、学生への動機づけを促す説明

新しいことを学べる＋実社会ではこういった環境で仕事をするから価値があるということを伝達

強制はせずできる範囲でやることを目標とさせた

C 校

2 年生：別課題の締め切りが近かった（そのため、あまり強制はしていない）

1 年生：学生に委ねる形で放置

遠隔で演習課題を取り組ませる上で、学習時の環境が影響
→学生への動機付けや声掛けと、取り組みやすい環境を用意する必要がある

（b）チャットボット利用について

チャットボットの利用が少ない

要因

- 登録回答が少ない
- 学生がチャットボットへの質問利用に慣れていない

対応

- 今回の実証で学生から出た Q&A を登録する
- チャットボット利用のチュートリアル作成を検討する

２．実習ユニットの実証

（１）目的

令和 4 年度に開発した教材およびプラットフォームを検証する。

（２）実施方法

専門学校で Java 言語の基礎を習得している学生を対象として検証した。対面と遠隔にグループを分け、企業から提供される課題（演習ユニットと同じ課題）を 4 名のチームを編成し、開発した。遠隔グループに対しては、実証前に委員 2 名を対象校に派遣して、学習環境確認および学生に対する説明会を開催した。学習時間は 2〜5 日程度を想定した。チーム開発にはモブプログラミング方式で開発を行った。対面グループは 2 日間授業形式で学習した。

遠隔グループは各自の好きな時間に学習した。学習時間は 3〜5 日程度とした。

方式	学生	期間
対面	情報系学生 20 名	12 月 13 日〜12 月 14 日
遠隔	情報系学生 19 名	12 月 13 日〜12 月 17 日

（３）実施結果

（ａ）対面

全体　　：平均 2.1 時間

チーム 1：　2.0 時間（Level3-2 到達）

チーム 2：　2.0 時間（Level3-2 到達）

チーム 3：　2.3 時間（Level3-2 到達）

チーム 4：　2.1 時間（Level3-2 到達）

チーム 5：　2.2 時間（Level3-2 到達）

（ｂ）遠隔

全体　　：平均 4.1 時間

チーム 1：　4.2 時間（Level3-2 到達）

チーム 2：　3.0 時間（Level3-2 到達）

チーム 3：　3.3 時間（Level3-3 到達）

チーム 4：　3.0 時間（Level3-2 到達）

チーム 5：　6.8 時間（Level3-3 到達）

（4）議論時間

- ナビゲーターの指示のもと、作業できていた割合
- プログラムの入力指示、サポータへの調査依頼など、課題に関する会話ができていた時間
- 沈黙等があるとマイナス

（a）対面

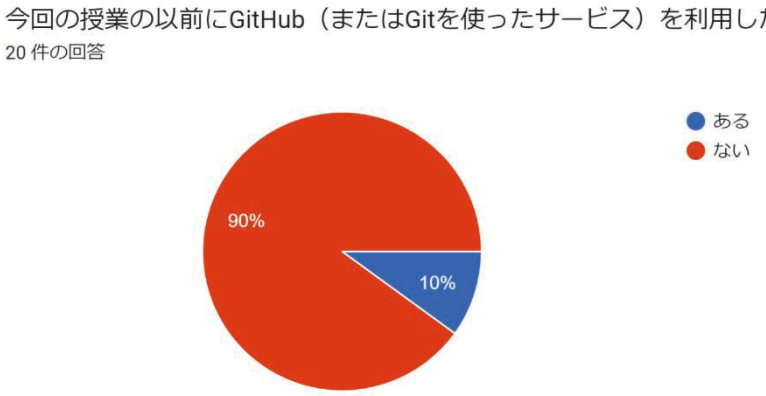
チーム名	平均	1巡目	2巡目	3巡目	4巡目	5巡目	6巡目	7巡目	8巡目	9巡目	10巡目	11巡目	12巡目	13巡目
1	**74%**	100%	50%	70%	70%	100%	60%	90%	60%	70%	70%	70%	80%	
2	**100%**	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
3	**90%**	80%	70%	80%	90%	100%	100%	100%	90%	100%				
4	**75%**	50%	80%	90%	90%	80%	80%	20%	80%	80%	80%	80%	80%	80%
5	**72%**	60%	80%	90%	80%	80%	50%	70%	70%	80%	80%	90%	50%	50%

（b）遠隔

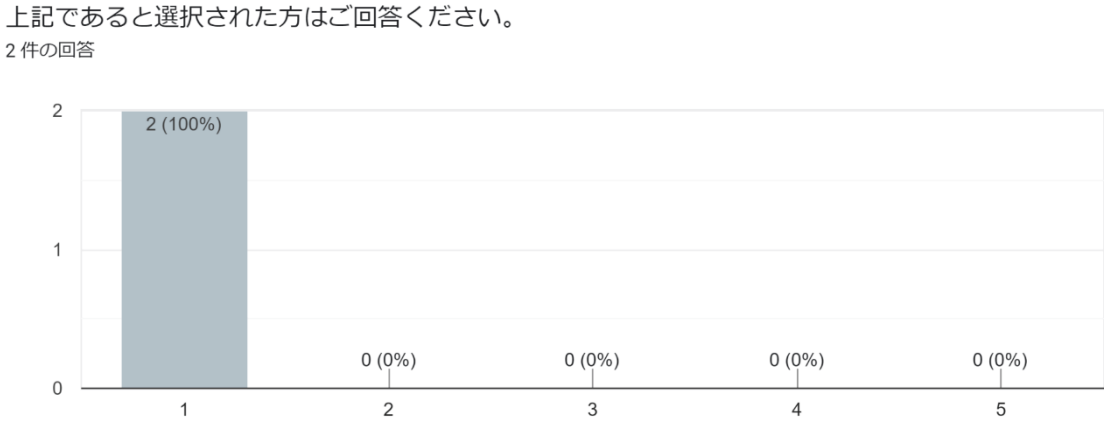
チーム名	平均	1巡目	2巡目	3巡目	4巡目	5巡目	6巡目	7巡目	8巡目	9巡目	10巡目	11巡目	12巡目	13巡目	14巡目	15巡目	16巡目	17巡目	18巡目
1	**83%**	80%	100%	60%	100%	60%	100%	80%	100%	80%	80%	80%	60%	80%	100%				
2	**84%**	30%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	80%	50%	50%	50%	50%
3	**89%**	90%	80%	100%	80%	90%	90%	100%	80%										
4	**83%**	70%	80%	90%	90%														
5	**84%**	100%	90%	100%	80%	90%	80%	100%	80%	80%	80%	70%	80%	80%	70%				

（５）実証後アンケート結果

（ａ）対面グループ

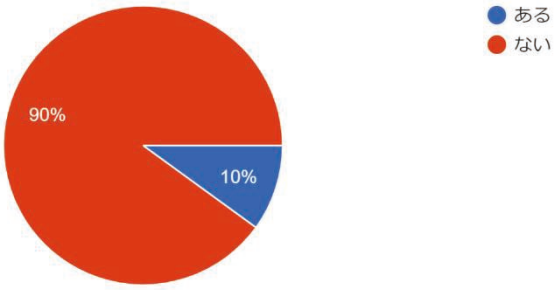


利用頻度



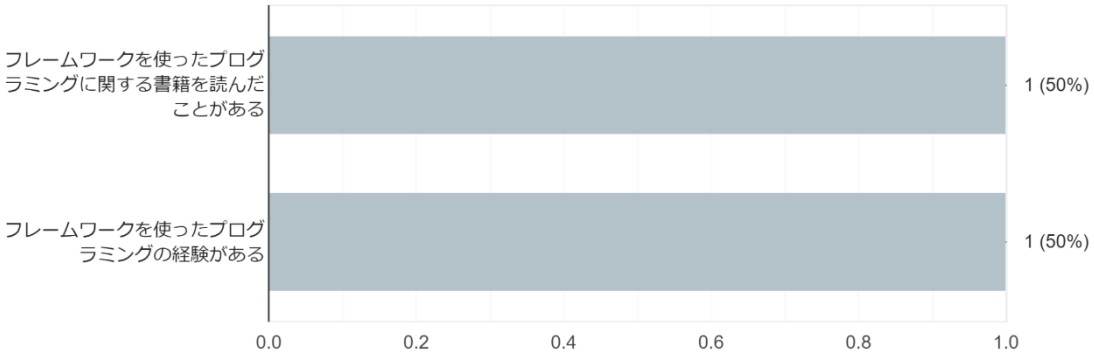
今回の授業の以前にフレームワークに関する知識または経験はありましたか？

20 件の回答



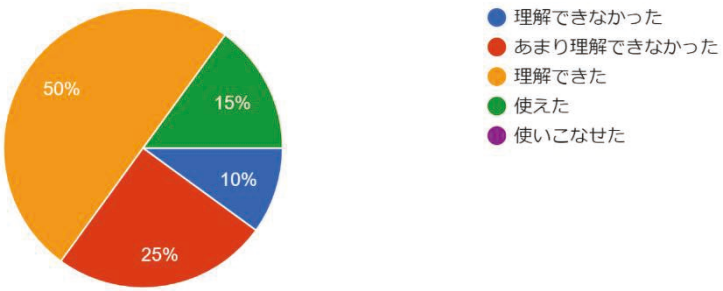
上記であると選択された方はご回答ください。

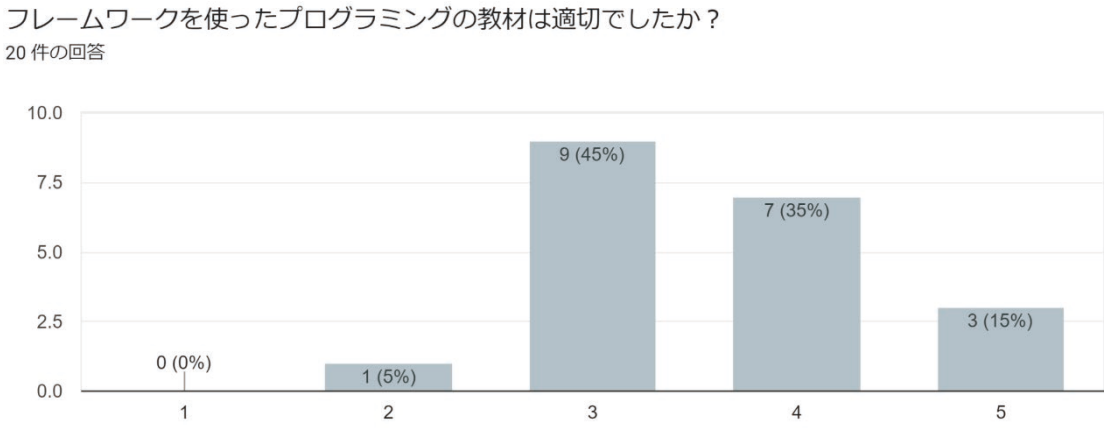
2 件の回答



今回の授業内容である「フレームワークを使ったプログラミング」についてご回答ください。

20 件の回答



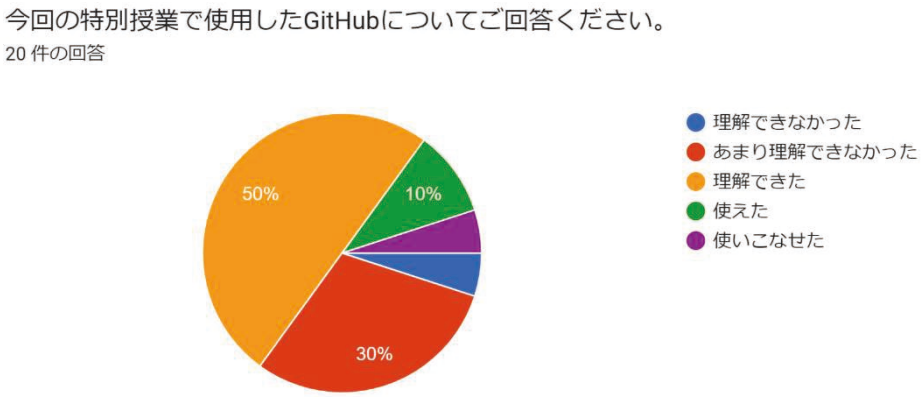


フレームワークをつかったプログラミングの授業について良かった点

- 細かいミスをおこしにくくなって理解がしやすかったと思います。
- 実際の業務でも使用されているフレームワークを体験できたことが良かった点です。
- 普段の授業ではプログラミングを一から作ることがほとんどなので、あらかじめ作成してあるプログラムに手を加えるという作業は新鮮なものでした。
- 開発の時間とコストの削減に役立つことが分かった。

フレームワークをつかったプログラミングの授業について改善点

- フォルダがたくさんあり、どのフォルダを開けばいいのか少し分かりづらかった
- 内容が難しかった
- 自分が作っていない部分に何が使われているかがわからないこと。
- プログラミング言語とは別にフレームワークの知識が必要なためもっと時間が必要。

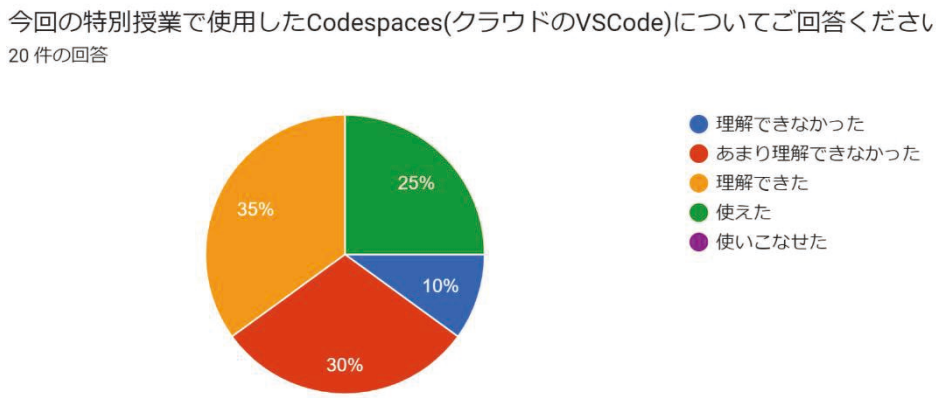


GitHub について良かった点

- プログラムの改善したところが見やすくなっていて、改善したところを同じグループの他の人に説明しやすかった。
- そのプログラミングが間違っていないかどうかがすぐに分かったこと。
- チーム内で作成したコードを共有出来たり、修正箇所を把握しやすい点が良いと感じました。
- みんなで内容を共有出来るから使いやすい

GitHub について改善点

- 誤ってコードファイルを複数作ってしまった場合ファイルにランダムでおかしな名前がつくので見分けがつかない点。
- エラーの内容を理解することが難しかった。
- 内容やメールが英語で理解しにくい部分が多い。

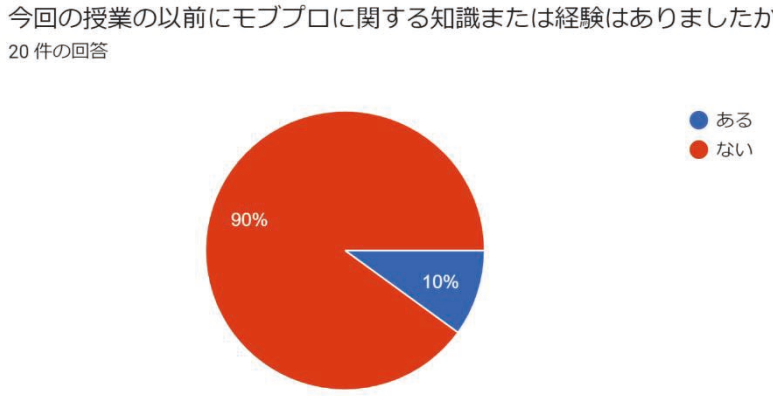


Codespaces について良かった点

- チームで共有できるところが良かったです。
- コミットする前の確認が楽で事故を防ぐことが出来た点。
- 実行結果がマイページに表示される点がわかりやすくてよいと感じた。
- vscode とほぼ同じで使いやすかった

Codespaces について改善点

- 操作で詰まったとき、操作方法が分かるガイドがあると良いと思いました。
- 動作が重くて読み込みに時間がかかったり上手くいかなかったりして作業が止まってしまうことがあったこと。
- 指示文が難しく、該当するコードがどこにあるのかわかりずらかった。
- エラーがなんのエラーか表示されたらより間違いが少なくなるのではと思いました
- 終了手順が普通のと違ってそのまま閉じていいものとそうでないものが混在していて少し複雑だったこと。
- 勝手に改行されるのがやりづらかった



モブプロについて良かった点

- チーム一体となってプログラムを作成できた
- 全ての役割を担ったので、自分の向いている役割、苦手な役割を知ることができました。
- 個々人の役割を交代していたのでそれぞれのやるべきことや負担を理解しやすい"
- 友人と協力しながらやっていると実感できてプログラムが上手くいった時にとても嬉しく感じる点
- しっかりと話し合うことが出来ました。
- 自分１人だと時間がかかるような問題でもモブプロだと周りの人と協力しながらやったので早く解決できたところ
- チームで意見を出し合って一つの課題に取り組むという点が新鮮であったため、よいと感じた。
- 考え方の共有出来たことで、新たな視点を発見でき、それぞれの立場が体験できて良かった
- お互いの知識を共有する経験ができることや、協調性の大切さを再認識できること。
- 一人一人にかかる負担を分け合えるできる点
- コミュニケーション力も鍛えられそうなところ。

モブプロの改善点

- 時間が決められていたので、深く考える前に時間が来てなかなか集中できなかったです。
- モブプログラミングに参加出来ない、しない人が出てきてしまうのでなんとか改善したい。
- 問題の難しさによっては人任せになってしまうところ
- 役柄通り絶対的に動くべきかそれとも状況次第に応じて全体で話しても良いのかなど役を完全に全うするべきか明言した方が良いのではないかと感じた
- グループ内でアイデアや修正案が出尽くしたような場面では、全体的に視野が狭くなってしまい、根本的な問題や小さなミスなどを見逃してしまう可能性があること。
- 役割毎にかかる負担の重さが違う
- もう少しいろいろな知識を身に付けておいた方がいいなと思った。
- 内容によっては効率が悪くなる。人が多くなると発言の機会が減るので集中力が散漫になるかもしれない。
- 役割によっては全く話し合いに参加することがなくなってしまう

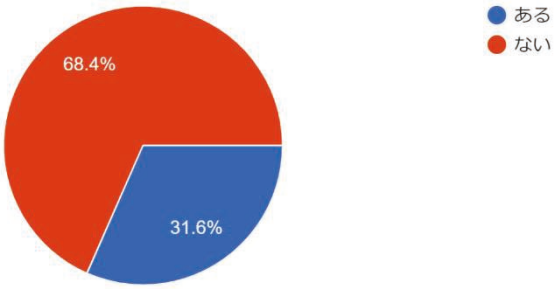
モブプロをやってみての反省点

- うまく自分の考えを発言できなかったところです。
- ナビゲーターが指示したり問題を解いたりなど大変だと思うので積極的により議論に参加すべきだと思いました
- 課題のブログラムが難しくなるほど自分ともう一人の人の二人で話し合っていて他の人を置いてけぼりにしてしまっていた
- モブプログラミングにあまり積極的ではない人が見られた。少し強引にでも巻き込めたらよかった。
- 難しい問題の時にあまり意見を出せなかった
- 途中で役割があやふやになり、全員がナビゲーターのような立ち位置となってしまったため、反省したいです。
- 話し合いの評価は無くても良いのでないかと感じました。傷つくひとや後の行動に行動に支障が出ると思いました
- モブプロの改善点にも書いたのですが、エラーに対してグループとしての意見が出尽くした後、講師の先生に指摘されるまで小さなミスに気づけなかった場面があり、時間を浪費してしまったこと。また、課題に対する読み込みが浅く、理解が足りなかった為、結果的にほとんど課題をこなせなかったことです。
- 回数を重ねるごとに話が途切れやすかったこと。
- ４人いたのにスペルミスにかなりの時間気づかなかった。誰かが見ていると思うせいか案外ミスがおこりやすいのかもしれないと思った。

（ｂ）遠隔グループ

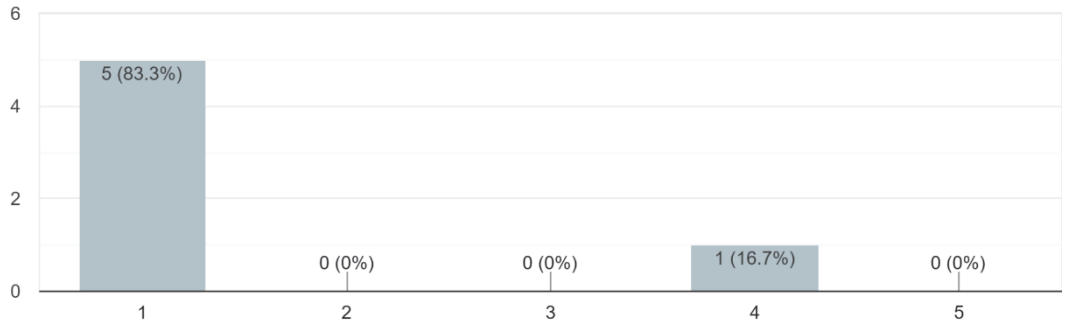
今回の授業の以前にGitHub（またはGitを使ったサービス）を利用したことはありましたか？

19 件の回答



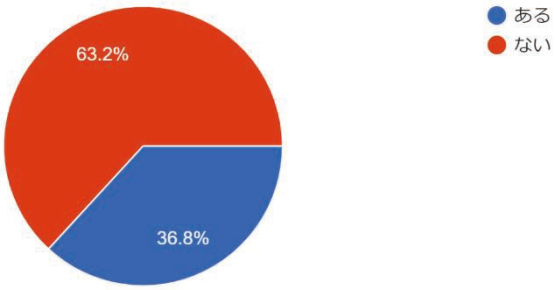
上記であると選択された方はご回答ください。

6 件の回答



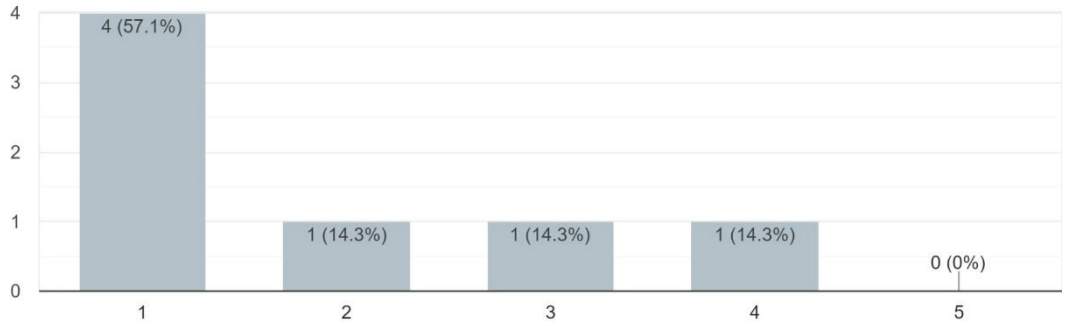
今回の授業の以前にSlackを利用したことはありましたか？

19 件の回答



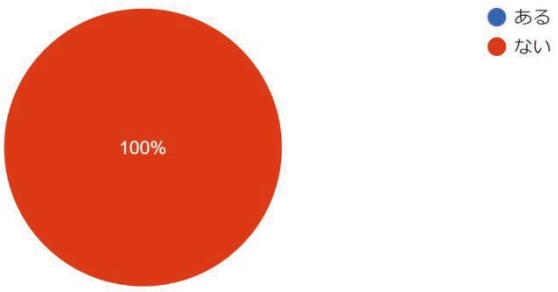
上記であると選択された方はご回答ください。

7 件の回答



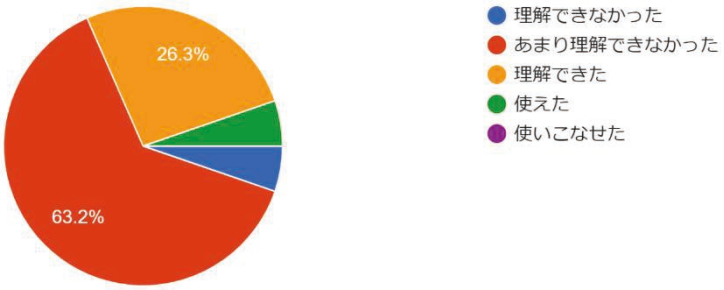
今回の授業の以前にフレームワークに関する知識または経験はありましたか？

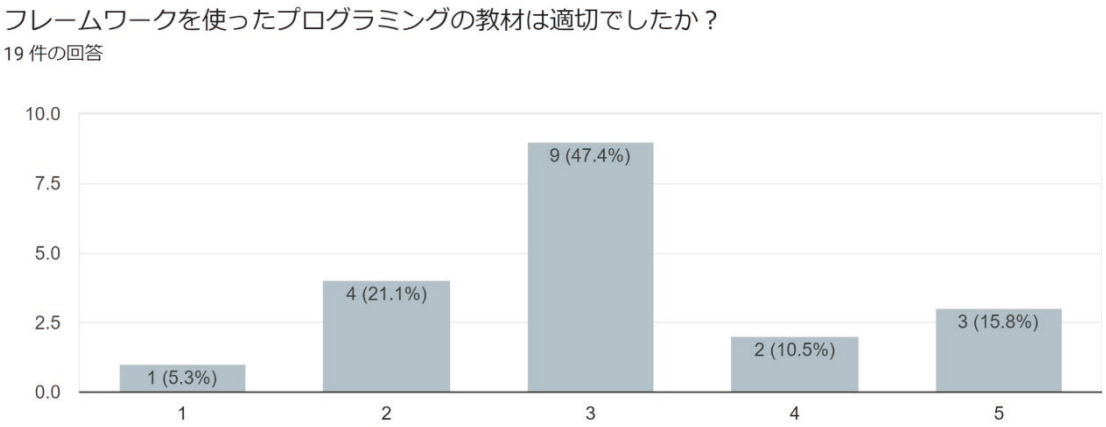
19 件の回答



今回の授業内容である「フレームワークを使ったプログラミング」についてご回答ください。

19 件の回答



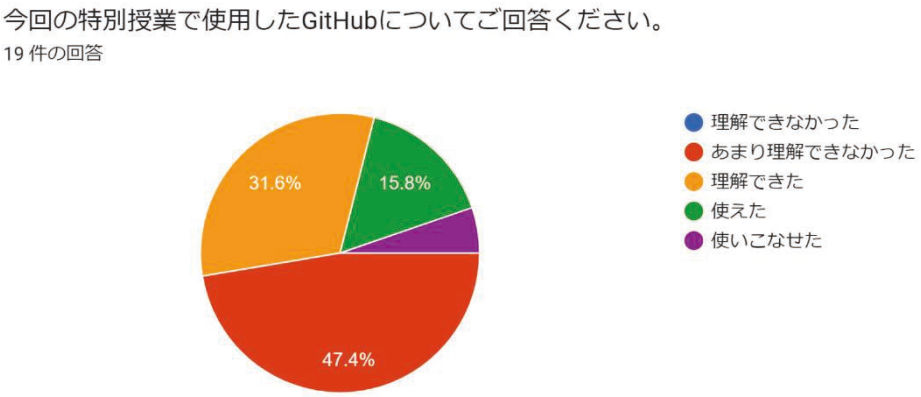


フレームワークをつかったプログラミングの授業について良かった点

- ほとんど事前に記述されているので入力ミスが起き辛い。
- 他のファイルを参考にできるので、何も参考資料がないよりも理解しやすかった
- 既にある程度作られている状態なので手を加える場所が推測しやすかった
- 社会で使われているフレームワークを使った授業は新鮮だった

フレームワークをつかったプログラミングの授業について改善点

- 難易度が高すぎてあまり理解できなかったため、もう少し短くて簡単なプログラムを使うか、説明を増やすかしていただけるとありがたかったです。
- ガイド（ReadMe)が少し分かり辛く、具体的に何処に何をすれば良いのか分かり辛い
- プログラミングの教材として、能力に見合ったものでないと理解ができず、分からない内容を調べなくてはいけないため、非常に時間がかかった。また、教えられる人が一人は必要だと感じた。

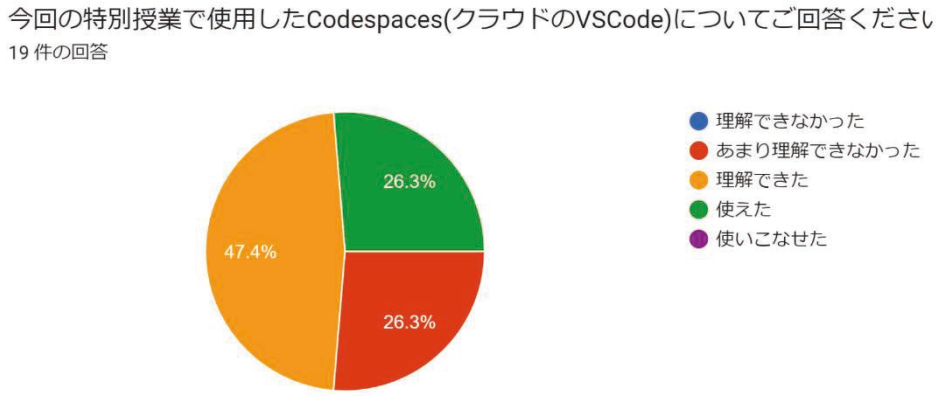


GitHub について良かった点

- 何を変更したかを書けるメモ欄があったので、変更した人以外の人にも分かりやすくて良いなと思った。
- リアルタイムでみんなと同じプログラムをくむことができること
- リモートでプログラミングするための準備が簡単にできて良かったです。
- プルリクエストで合っているかすぐ確認できる
- 社会で使われている GitHub を使った授業は新鮮だった

GitHub について改善点

- 英語表記なので、慣れるまでは大変だなと思った。
- ガイド（ReadMe)が少し分かり辛く、具体的に何処に何をすれば良いのか分かり辛い
- ホストの立ち上げが必要な点



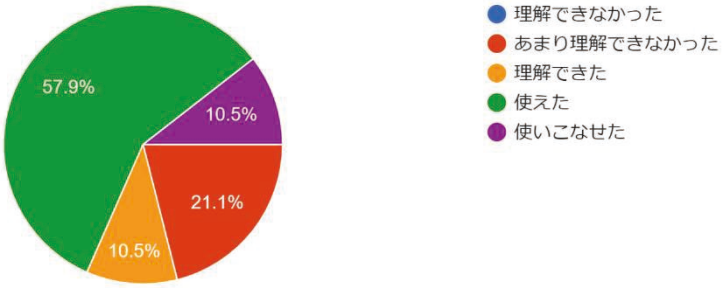
Codespaces について良かった点

- フォロー機能がとても便利だった
- ライブシェアでグループの人と共有できることや、作業している人を自動で追跡してくれる機能があったこと。
- 誰がどこで作業しているのか分かるので進めやすかったです。
- 全員の入力したコードが即座に反映される
- 一度理解出来たら共有などがしやすくてよかった

Codespaces について改善点

- 使用中に何度か切断された
- プログラムを書いている時に、時々、勝手に改行されてしまう点。
- エラーの内容がホスト側とゲスト側で違う（エラー内容が同期されていない）
- VScode 上でエラーを見るときに英語でしか表示されなかったので、日本語表示ができればよかった

今回の特別授業で使用したSlackについてご回答ください。
19 件の回答



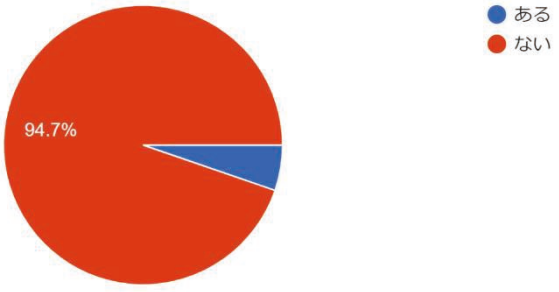
Slack について良かった点

- 一人でも通話が開けたので、先に開いてあとからメンバーが入ってこれるのがよかった
- 通話しながら画面共有もできたので、状況把握がしやすかったこと。
- このアプリ一つでチャットや通話ができ、画面共有もできるので便利でした。
- 会議の参加が容易、コミュニケーションがとりやすい
- チームごとに分けられていて良かった。

Slack について改善点

- パソコンを開かないと見えないので面倒だった
- 関係ない人が他チームの会話内容（チャット）を覗けてしまうこと
- ui が使いづらく、音質や音量バランスなども満足できなかった

今回の授業の以前にモブプロに関する知識または経験はありましたか？
19 件の回答



モブプロについて良かった点

- 自分では気づけなかった点に他の人が気づくことで、よりスムーズに作業できたので良かったです。
- 自分では分からない点があってもチームメンバーが補ってくれること
- 役割が分担できるので効率がいい
- 分からなくなったときに相談ができるのがモチベーションに繋がった
- 自分の意見だけでなくたくさんの意見や視点を知ることができる

モブプロの改善点

- 人数がもっと増えたら発言しにくそうだと思った。
- 三人寄れば文殊の知恵のようなことは起こらず知識も技術もない三人が集まっても課題解決はできませんでした。
- 討論の時間が長引くと交代することを忘れたり、交代するとキリが悪い場合がある

モブプロをやってみての反省点

- 途中から役割を無視していたことがあった。
- 答えを考えるのに集中しすぎて、他の人の意見を聞けていない時があったので、自分の役割をしっかり意識し、考える時間、聞く時間のメリハリをしっかりつけるべきだったと思いました。
- グループ内で課題に関する解釈がそれぞれ違っていたりしたので読解力が足りなかった。

（6）考察

- 議論時間の平均をとると、遠隔が 84%と対面が 82%であまり差がない結果となった。この結果、実施する環境の違いがグループワークに影響しないと考えられる。
- 各課題の完了数、品質、開発速度の計測ができなかった
  対象者（1 年生）に対する課題のレベルが高く十分なデータを計測するに至らなかった
- チーム全体のレベルアップ効果は認められなかった
  課題が進む際に難易度が上がるためレベルアップ効果が見えづらかった
- 実習ユニットで使用する課題内容を検討する

事業評価

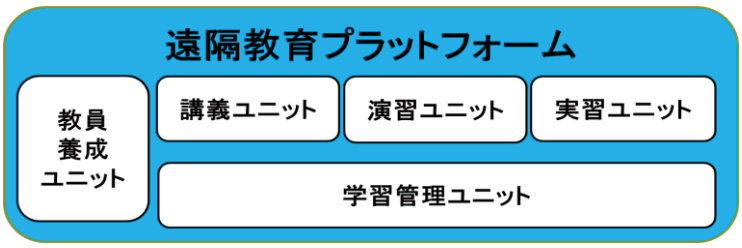
令和４年度「遠隔教育によるチームプログラミング導入モデル構築」事業の評価

1.　「遠隔教育によるチームプログラミング導入モデル構築」事業の全体像

1.1.　事業の概要

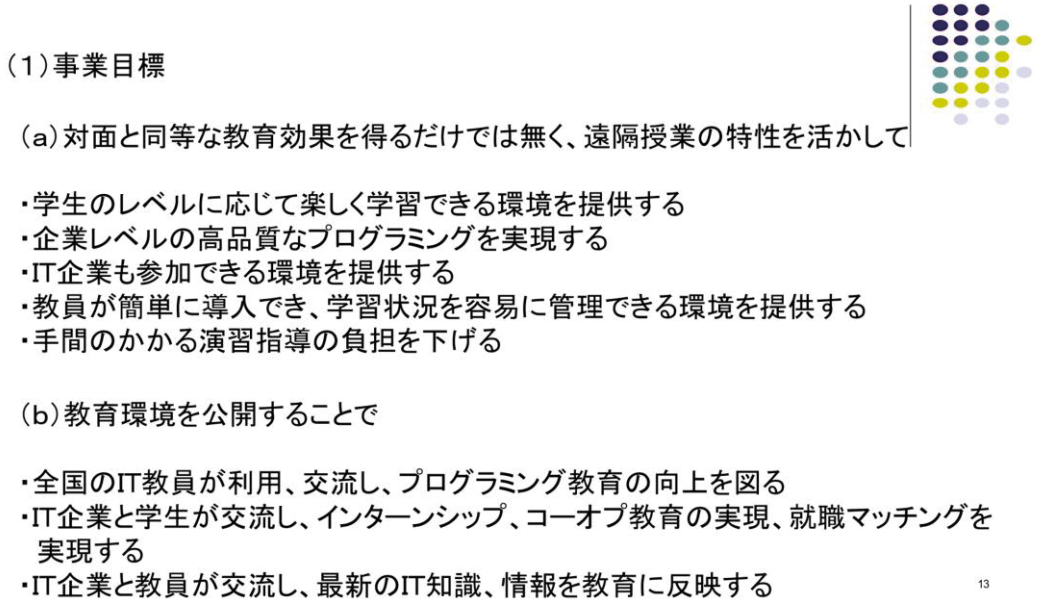
事業概要を記す。

- クラウド上に無料または安価な既存サービス、アプリケーションを組み合わせて、遠隔教育プラットフォームを構築する。プラットフォームの概念図を下記に示す。



- 学生にプラットフォーム上の演習課題を取り組ませ、対面授業と同等以上の教育効果を実現しているか、企業レベルのプログラミングに達しているか検証、評価する。
- 遠隔教育プラットフォームを全国に復旧させる。

1.2.　事業の目的



1.3. 事業のスケジュール

事業スケジュールを記す。

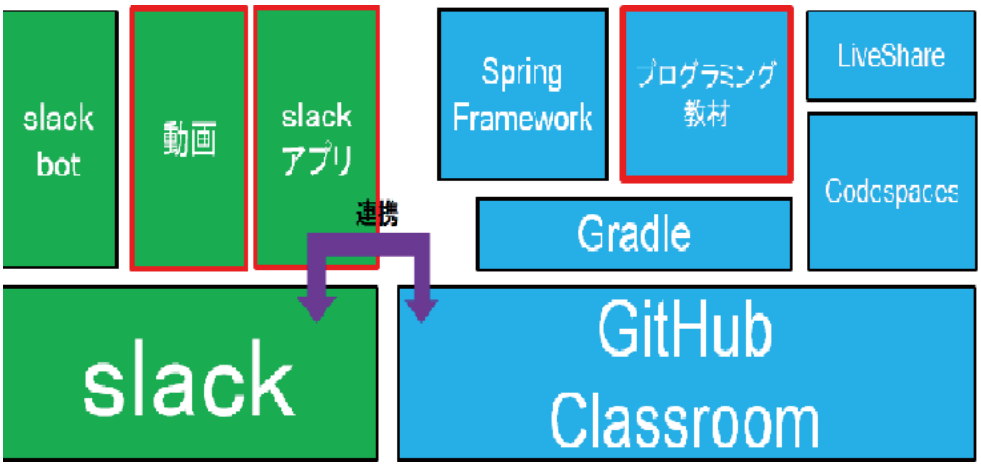
項目	令和3年度	令和4年度	令和5年度
調査	• 専門学校プログラミング教育調査 • プログラマ育成実態調査		
講義ユニット			• 開発 • 実証
演習ユニット	• 一次開発 • 実証	• 二次開発 • 実証	
実習ユニット		• 一次開発 • 実証	• 二次開発 • 実証
学習管理ユニット	• 一次開発 • 実証	• 二次開発 • 実証	• 三次開発 • 実証
教員養成ユニット			• 開発 • 実証

2. 令和 3 年度事業の結果

2.1. 事業の概要

令和 3 年度の事業を記す。

- 専門学校のプログラミング教育に関する状況調査
- プログラマー育成に関する実態調査
- 遠隔教育プラットフォームの開発。プラットフォーム構成図を下記に示す。
  プラットフォームで動く演習ユニットと学習管理ユニットを開発。



- 遠隔教育プラットフォームの実証
  富山情報専門学校の学生 40 名を対面授業と遠隔授業に分けて、同じ演習を実施し、教育効果（完了数、速度、品質など）を測定した。

2.2. 事業の成果

報告書にある成果を記す。

- 利用費、運用費が無料のプラットフォームを開発することができた
- 導入の手間の少ないプラットフォームを開発することができた
- 対面と同等の教育効果を出すことができた
- フレームワークを利用したプログラミングを学生に理解させることができた
- プラットフォームに対する学生の参加性の高さを確認できた

2.3. 課題と対応

報告書にある課題と対応を記す。

- 遠隔教育プラットフォームを利用できない学生が少数ながらいた
  →利用説明などの動画を強化する
- 教員―学生間に二通りのコミュニケーション手段を使用している。
  →企業での標準形式を優先するか、教育を優先するか検討する
- 企業は学生に高いプログラミングスキルを求めていない
  →プログラミング習得を容易にし、その他のスキル習得時間を確保す。

3.　令和 4 年度事業の達成状況

令和 4 年度の事業計画は令和 3 年度事業の結果・課題を踏まえて作成した（令和 4 年度「専修学校における先端技術活用実証研究」事業計画書参照)。事業計画上の作業概要は以下 6 点。

① 完成版演習ユニットの完成
② 学習管理ユニットの第 2 次開発
③ 実習ユニットプロトタイプの作成
④ 演習ユニットおよび学習管理ユニットの実証
⑤ 実習ユニットの実証
⑥ 情報公開

以下、上記 6 点の作業について、その達成状況を確認する。。

3.1.　「①完成版演習ユニットの完成」の達成状況

番号	内容	達成状況
1	可読性、信頼性、拡張性等を担保するためのプログラミング規約を追加する。	○
2	ベースとなる業務プログラムを用意し、設計書を元にプログラム改変を行う形式教材を新規に開発追加する	○
3	上記教材の改変内容はレベル 1 ～ 4 に分けて、各レベルに 2，3 本の課題を用意する	○
4	特に低いレベルの課題は多めに用意する。	○
5	レベル 0 としてベースプログラムの解析課題を用意する	○
6	Spring フレームワークを使用してマイクロサービスをイメージしたスパイラルな開発を行う教材とする	○

3.2.　「②学習管理ユニットの第 2 次開発」に関する達成状況

番号	内容	達成状況
1	チャットボット対応項目を追加する	○
2	チーム開発機能を追加する	○

3.3. 「③実習ユニットプロトタイプの作成」の達成状況

番号	内容	達成状況
1	レベル付けを可能なプログラミング課題管理システムを開発する	○
2	企業別のプロジェクトを管理するためのシステムを開発する	○
3	チーム内でのコミュニケーションを管理するためのシステムを開発する	△（注）
4	テスト（品質）を管理するためのシステムを開発する	△（注）
5	プログラムを評価するためのシステムを開発する	○
6	就職マッチングシステムを開発する	△（注）

注）部分的には開発した。

3.4. 「④演習ユニットおよび学習管理ユニットの実証」の達成状況

番号	内容	達成状況
1	令和 4 年度開発した教材、プラットフォームを検証する。福岡、高松、千葉の専門学校で Java 言語の基礎を習得している学生 20 名を対象とする。	○
2	実証前に、委員 2 名程度を対象校に派遣し、学習環境確認及び学生に対する説明会を実施する	○
3	令和 4 年の１１～１２月に各自の好きな時間に学習する。学習時間は 3 ～ 5 日程度を想定する	○

3.5. 「⑤実習ユニットの実証」の達成状況

番号	内容	達成状況
1	令和 4 年度開発した教材、プラットフォームを検証する。富山情報ビジネス専門学校の学生 20 名を対象とし、プラットフォームを検証する。	○
2	企業から提供される課題を 3 ～ 5 名程度のチームを編成し、開発する。	○（注）
3	令和 4 年の１１～１２月にチーム主体で日程を調整し学習する。学習時間は 3 ～ 5 日程度を想定する	○

注）企業からの課題ではなく演習ユニットの課題を使用。演習ユニットは企業が行なっている開発に近い課題となっているので問題はない。

3.6. 「⑥情報公開」に関する達成状況

番号	内容	達成状況
1	HP や YouTube による成果物の公開や動画コンテンツによる事業紹介を行う。	—（注）
2	成果報告会を 1 度実施する。	—（注）

注）評価時点で実施されていない。

4.　令和 4 年度事業の評価

3．項では令和 4 年度事業に計画された作業項目を実施したかどうかを確認した。3．項のサマリーを下図に示す。4．項ではそれぞれの作業結果を評価することとする。

番号	作業概要	達成状況
1	完成版演習ユニットの完成	○
2	学習管理ユニットの第 2 次開発	○
3	実習ユニットプロトタイプ開発	△
4	演習ユニットおよび学習管理ユニットの実証	○
5	実習ユニットの実証	○
6	情報公開	—（注）

注）評価時点で実施されていないので未評価とする。

4.1.　「完成版演習ユニットの完成」の評価

まず完成版演習ユニットの概要を説明する。次に完成版演習ユニットを使った実証講座の結果を元に完成版演習ユニットを評価する。

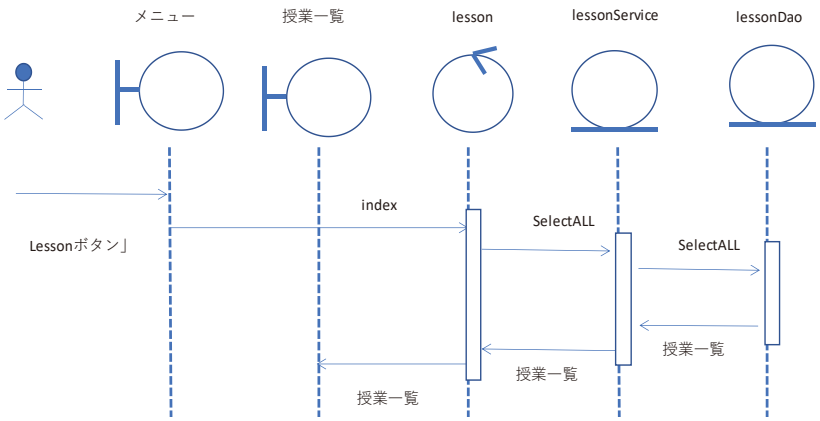
4.1.1.　完成版演習ユニット概要

演習ユニット教材は単なる Java プログラミング演習ではなく、実際、企業で使われているコーディング規約やフレームワーク、MVC モデルに則ったプログラミング演習となっている。当然、フレームワークや MVC モデルの知識を要求される演習である。演習は 0 から 4 までの 5 つのレベルからなり、レベル 0 を除き各レベル 1 個ないし数個の課題がある。各レベルの内容は以下。

レベル		内容	ユースケース
0		フレームワーク、MVC モデルの解説用サンプル	授業一覧表示
1	プログラム修正		ポータル画面表示
	1	Controller の修正	
	2	Service（天気）の修正	
2	プログラム修正		ポータル画面表示
	1	Service（おみくじ）の修正	
	2	Service（挨拶）の修正	
3	新規プログラム開発		
	1	DB 事前準備（ＤＤＬ，ＤＭＬの作成）	
	2	実績一覧表示ユースケースの新規開発	実績一覧表示
	3	実績登録ユースケースの追加に対する開発	実績一覧表示に追加
	4	実績更新ユースケースの追加に対する開発	実績一覧表示に追加

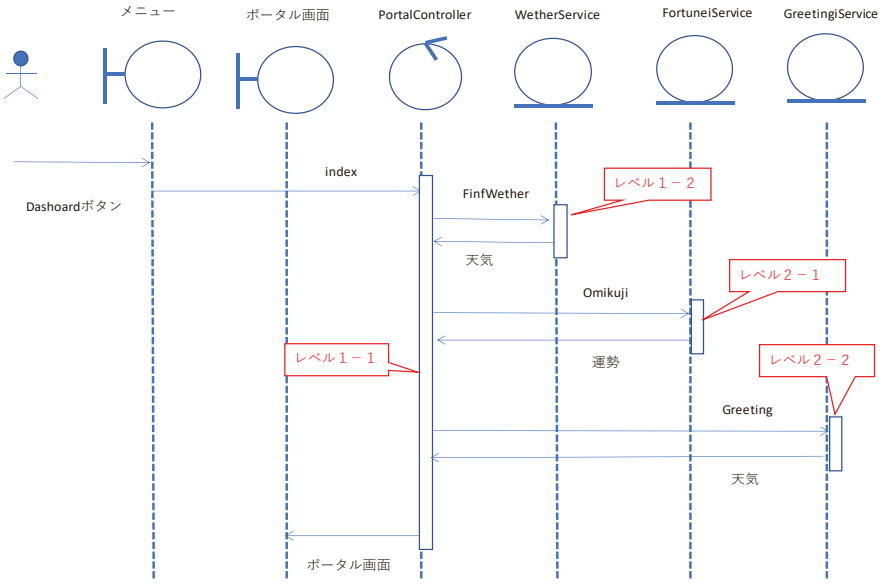
レベル		内容	ユースケース
	5	実績削除ユースケースの追加に対する開発	実績一覧表示に追加
	6	ポータル画面表示に実績一覧を追加	ポータル画面表示に追加
4		ポータルユースケースに ToDo 管理ユースケースを追加	ポータル画面表示に追加

演習のレベル 0 は授業一覧表示のユースケースをサンプルに、フレームワークと MVC モデルを解説している。このユースケースを MVC 分解した簡易シーケンス図を下記に示す。実際の開発現場では要件を記述したユースケースから、ユースケースを実現する処理の流れを記述したシーケンス図が描かれる。シーケンス図にはクラスに属するメソッド（プログラム）の動きが記述される。○に左横棒が View、○に→が Controller、○に下棒が Model、縦棒部分がメソッド（プログラム）である。実際の開発では個々のメソッド（プログラム）毎に処理の詳細を記した仕様書が作成され、それによりプログラムが作成される。



レベル 0 の解説にはボタン押下から一覧画面表示までの Spring を通して動くメソッド（プログラム）作成に必要な要素がほぼ含まれている。レベル 0 の理解はその後の課題、特にレベル 3 以降の課題を解く前提となる。

レベル 1、2 の課題はポータル画面表示のユースケースを使う。このユースケースをＭＶＣ分解した簡易シーケンス図を下記に示す。レベル 1，2 の課題は既存のプログラムが準備され、課題はその既存プログラムの修正である。



レベル１－１は Controller の既存メソッドを修正して自分の名前を表示させる課題である。レベル１－２では天気予報、レベル２－１ではおみくじ、レベル２－２ではあいさつ文というそれぞれの既存 Service のメソッド（プログラム）を指示に従って修正していく。レベル１，２は既存プログラムの修正だけの比較的容易な課題である。

レベル３の課題は実績データの一覧表示ユースケースの開発である。これは修正ではなく、必要なクラス、メソッドを新しく作成しなければならない。レベル３－２課題はレベル０の授業一覧表示と全く同じＭＶＣ構造である。ただ違うのは授業 DB の代わりに実績 DB を使うだけなので、レベル０のサンプルを理解していれば、レベル３－２課題は容易である。レベル３－３は実績の登録、レベル３－４は実績の変更、レベル３－５は実績の削除という新たなユースケースを実績一覧ユースケースに追加する課題である。クラス構成は変わらず、各クラスのメソッドを追加していく課題になっている。レベル３－６は実績一覧をポータル画面に表示させる課題である。Portal コントローラーの既存メソッドにレベル３－２課題で作成した実績サービスを組込み、実績を取得、表示するよう修正する。MVC モデルの利点であるサービスの再利用を実感できる課題である。レベル３の課題は MVC モデルの理解が肝になる。

レベル４はポータル画面ユースケースに ToDo を管理する機能（ToDo の一覧表示、登録、削除）を追加する課題。新たな知識を要求する機能はないがレベル３課題全てを一挙に開発するのとほぼ同じになるので、これまでの集大成と言える。以上のように、今回開発した演習課題はスパイラルに教育効果が出るよう、既存プログラムの修正から新たなユースケース、更にはユースケースの改変へと、フレームワークやＭＶＣモデルの理解が深まるよう、よく考えられた構成になっている。

4.1.2.　完成版演習ユニットの評価

令和 4 年度は演習ユニットの実証も実習ユニットの実証も令和 4 年度開発の演習ユニット教材が使用された。演習ユニットを使った実証講座が計 4 回（福岡、高松、千葉、富山）実施されたことになる。演習ユニットの実証を行った 3 校（福岡、高松、千葉）ではレベル３－１の課題まで完了した学生が最も多かった（21%）。また、そこまで達していない学生は 72%だった。しかし、レベル 4 まで全て完了した学生も 10%いる。レベル 1，2 の課題はベースプログラムの修正である。（レベル３―１は DDL、DML を記述するだけでレベル３－２の準備でレベル３－２に含めて考えることにする。）レベル３－２からは改変ではなくユースケース全体を構成するプログラム全部を新規作成する課題であり難易度が増す。実習ユニットの実証をした富山情報専門学校ではチームで開発したにも拘らず、対面授業では全チームがレベル３―１まで、遠隔授業では 3 チームがレベル３－１まで、残り 2 チームがレベル３－２までを完了したに過ぎなかった。この進捗の妥当性は判断できないが、前述のようによく考えられた教材としては残念な結果と言える。その原因を探ってみよう。

レベル 1，2 は既存プログラムの修正である。既存プログラムを指示に従って修正するだけで MVC モデルの実装という概念は必要ない。レベル 3 からは新たなユースケースが使われる。実績一覧を表示するというユースケースである。ユースケースが新しくなるということはユースケースを MVC モデルに従って分解し、シーケンス図に落とし込まなければならない。シーケンス図で明らかになった全てのメソッドをプログラミングすることになる。単にメソッドを新たにプログラミングするのではなく、シーケンス図を作るという設計作業を理解し、作成しなければならない（勿論、シーケンス図に相当する文章は教材にあるが）。その作業は設計であり、プログラマーの作業範疇ではない。従って、プログラミング能力育成と言う意味では範囲を超えている。プログラミング教育事業としてはシーケンス図を所与として、そこに現われるメソッドをその仕様書から作成するという演習で十分である。フレームワークや MVC モデルの深い知識は必要ない。しかし、昨今、アジャイル開発の普及とともに設計者とプログラマーの垣根がなくなりつつあること、また、IT を教える専門学校は単にプログラマーを育成する機関ではないこと、企業は単なるプログラマーを求めていない事などを考慮すれば、企業の現場で実際使用されているフレームワークや MVC モデルの開発技法を学ぶことは大きな意義がある。難易度の高い演習の中、レベル 4 まで完了した学生が 7 人いたことはフレームワークや MVC モデルをちゃんと理解できる演習ユニットであることを証明している。今回開発の完成版演習ユニットは十分評価できる内容である。

しかし、この実証講座の実績を踏まえ、レベル 3 で躓かない工夫も必要である。実際の開発現場と同等というならシーケンス図も提示し、その読み方も教えるべきである。また、DB 操作に不慣れという意見もあったが、それならレベル 2 にＤＢ操作の課題を追加することも考えるべきであろう。全体の達成レベルを上げる工夫が重要になる。

4.2. 「学習管理ユニットの第 2 次開発」の評価

学習管理ユニットの第 2 次開発の成果物は以下の二つ。

- 対応項目を追加したチャットボット
- チーム開発機能

チャットボットについては教員の負担軽減に重要な役割を担う。平成 3 年度事業の実証講座ではチャットボットの使用は 0 回だったので、令和 4 年度事業で新たな QA 対応を追加したが、結果は 49 件の QA のうちチャットボットの利用は 2 回に留まった。なぜ、使用率が低いのか分析し、対応策を策定しなければならない。

チーム開発はモブプログラミングの考え方を採用して実施。遠隔でモブプログラミングを実践するツールとして Visual Studio Code の Live Share を利用し、作業 PC を共有、同時作業を可能とした。実習ユニットの実証講座で LiveShare を使用し、円滑に活用ができることを検証した。

チャットボットについては今後改善の余地はあるが、学生管理ユニットの第 2 次開発成果物としては十分な機能を備えている。

4.3. 「実習ユニットプロトタイプの開発」の評価

実習ユニットプロトタイプのうち、作業番号 1 、2 、5 については演習ユニットから流用することにより新たな開発はなかった。残りの作業については部分的には開発されたが、まだ完成されていない。従って実習プロトタイプの機能としては不十分である。

4.4. 「演習ユニットおよび学生管理ユニットの実証」の評価

実証講座の検証は事業計画にもあるように対面と遠隔の比較を前提としている。従って、令和 4 年度に実施された 3 校での実証講座は遠隔のみの実証だったので評価できないが、事業全体では演習ユニットの対面と遠隔での比較検証は令和 3 年事業で実施済みなので特に問題はない。しかし、令和 4 年度開発した演習ユニットを使い、更には 3 校に対象を拡げて、対面と遠隔の比較ができればより精緻な検証になったと思われる。ただ、令和 4 年度開発の演習ユニットが遠隔プラットフォーム上でトラブルなく稼働することを複数校で確認できたことは評価できる。

4.5. 「実習ユニットの実証」の評価

今回の評価では演習ユニット教材を使った実習ユニットの実証結果を検証する。事業計画では検証項目を課題の完了数、品質、課題完了までに要した時間、フレームワークの理解度としているのでこれに沿って検証する。今回行われた実習ユニットの実証（富山）は前に 3 校（福岡、高松、千葉）で実施した演習ユニットの実証と同じである。異なるのは実習ユニットの実証がチーム開発で行われ、演習ユニットの実証が個人での開発と言うことに限られる。

4.5.1. 完了数と課題完了までに要した時間

富山での実証では対面の 5 チームは全て課題レベル３－１までの完了でレベル３－２の途中だった。要した時間は各チームほぼ同等で平均 2.1 時間だった。遠隔は 5 チームのうち 2 チームが課題レベル３－２まで完了し、残り 3 チームはレベル３－１までの完了だった。要した時間は最高 6.8 時間、最低 3 時間、平均４．１時間だった。課題レベル３－２を完了したチームの要した時間は 3.3 時間、6.8 時間だった。時間をかければ完了レベルが上がるという訳ではない。課題完了レベルは遠隔の方が高いが、要した時間も遠隔は個人のほぼ 2 倍だった。また、課題ごとの所要時間が不明なので、遠隔、対面での所要時間を比較することはできなかった。表題について遠隔と対面での優劣は判定できない。

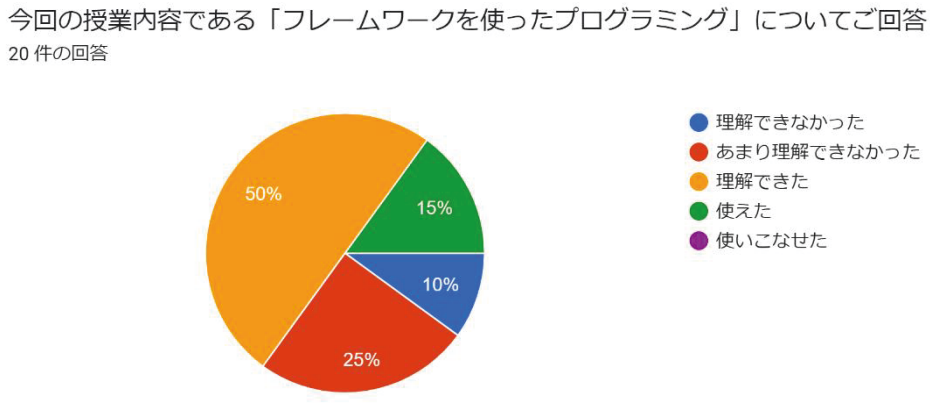
4.5.2. 品質

事業計画ではプログラムテスト設計書を作成することを前提とし、事前に教員と IT 企業が評価指標を準備し評価するとあるが、今回の実証講座ではテスト設計書を作成していない。また、評価指標も準備されていない。ただ、この評価方法が妥当かどうか検討した方がいい、品質の評価がテスト設計書の検証でいいとは思えない。この表題についても遠隔と対面の優劣を判定できない。

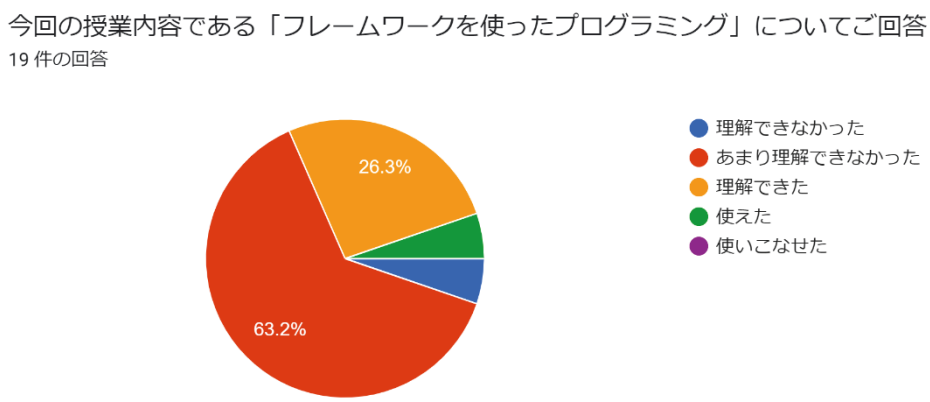
4.5.3. フレームワークの理解度

フレームワークの理解度は対面が圧倒的に高い。対面の「理解できた」以上は 65%、遠隔は 30%程度だった。しかし、その結果が完成数に反映している訳ではない。このアンケートだけでは両者の理解度を判定することは難しい。

対面の理解度



遠隔の理解度



4.5.4.　チームプログラミング

4.5.1.　　チームプログラミング

最後にチームプログラミングについて言及する。今回の実証講座（富山）ではチームプログラミング手法としてモブプログラミングを採用した。モブプログラミングと単独開発との比較検証をしっかりと行ったわけではないので正確な評価はできないが、同じ演習教材を使って 3 校（福岡、高松、千葉）で演習ユニットの実証を個人開発で実施しているので、両者を比較してみる。全チームがレベル 3 の中途で躓いでいる、個人でもレベル 3 の中途で躓いているのでモブプラミングの効果を検証するのは難しい。ただ、全課題を完了した学生が個人開発では 10%いたが、チーム開発では全くいない。優れた学生個人には物足りない手法と言えるだろう。また、富山の実証ではモブプログラミングはレベル 3 からしか実施していない、仮にレベル 1，2 でモブプログラミングを実施していれば、個人開発で課題となった 19%の落ちこぼれ学生をチーム開発で救える可能性も検証できたかもしれない。レベル 2 までもモブプログラミングを実施すべきだったと思う。

モブプラミングは 4 人の学生の役割をドライバー、ナビゲーター、サポーター、記録係と規定し、10 分毎に役割をローテーションする方法で実施した。作業が滞る時間を計測したが、遠隔でも、対面でも作業が滞った時間は全体の 20%弱で、モブプログラミングとしては円滑に実践されていたと言える。学生アンケートではその効果について多くの学生からの好意的意見が寄せられている。

モブプログラミングは本来 1 台の PC の前にチーム全員が集まって開発することを想定している。今回、遠隔では Live Share を活用してモブプログラミングを実践した。遠隔と対面で比較したかったが、前述のように完了数とか生産性、品質を測定できなかったので比較評価はできてない。

以上から、令和 4 年度の実習ユニットの実証については検証できるだけのデータが不十分だったため評価は未達とするが、初めての試みであるチームプログラミングの有効性は確認できたものと評価できる。

4.6. 評価のまとめ

令和 4 年度事業の作業達成状況と作業結果の評価を下記にまとめた。6．の情報公開の作業は来年度に延期したので評価していない。全体的に概ね、目標通りの結果だったと評価できる。

番号	作業概要	達成状況	評価
1	完成版演習ユニットの完成	○	○
2	学習管理ユニットの第 2 次開発	○	○
3	実習ユニットプロトタイプ開発	△	△
4	演習ユニットおよび学習管理ユニットの実証	○	○
5	実習ユニットの実証	○	△
6	情報公開	—	—

5. 結び

令和 4 年度の事業について評価した。まず、令和 4 年度に計画した作業の達成度を確認し、その作業結果を評価した。概ね、事業の目的を達成しているものと評価した。事業全体の目的は、遠隔と対面での教育効果を比較し、遠隔教育が対面教育と同等或いはそれ以上であることを検証することにある。教育内容は企業レベルの高品質なプログラミングであり、IT 企業の参画や、教員の負担の軽減できる仕組みの構築を目指す。更にできたプラットフォームを公開し、教員や IT 企業が交流し、プログラミング教育の向上をはかり、就職マッチングにも利用できるようにすることにある。平成 4 年度では実際の企業で使っているフレームワークを使った MVC モデルに則ったプログラミング教材を開発し、実証講座を実施できた。しかし、以下のような積み残しもある。

- 実習ユニットに搭載すべき機能の改変
  - 品質管理システムプログラム評価システム
  - 就職マッチングシステム
  - チーム内コミュニケーション管理システム
- 実習ユニットの検証
  - 令和 4 年度の実習ユニットでの実証では検証すべきデータ取得が不十分だった。検証のポイントはチームプログラミングの遠隔と対面の比較であり、上記積み残し機能の検証である。
- 情報公開
- その他
  - チャットボットの使用率向上
  - 品質管理の考え方を策定した上での実証

今後、令和 5 年度の事業計画に上記積み残しをどう組み込むかを検討する必要がある。当初の事業目的と整合するよう、実行可能な計画策定が求められる。

以上

令和 4 年度「専修学校における先端技術利活用実証研究」
専修学校遠隔教育導入モデル構築プロジェクト

遠隔教育によるチームプログラミング導入モデルの構築

事業成果報告書

令和5年2月発行

学校法人浦山学園　富山情報ビジネス専門学校
〒939-0341　富山県射水市三ヶ576
TEL:0766-55-1420　FAX:0766-55-0757

●本書の内容を無断で転記、掲載することは禁じます。