Описание работы с проектом

Содержание

Используйте эту ссылку, чтобы скачать PDF версию.

Подготовка к работе

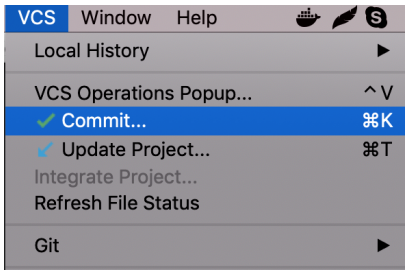
1. Установить JDK (скачать).
2. Установить Git (скачать).
3. Установить IntelliJ Idea Community Edition (скачать).
4. Создать аккаунт на Github (перейти на страницу для создания аккаунта).

Получение новых заданий

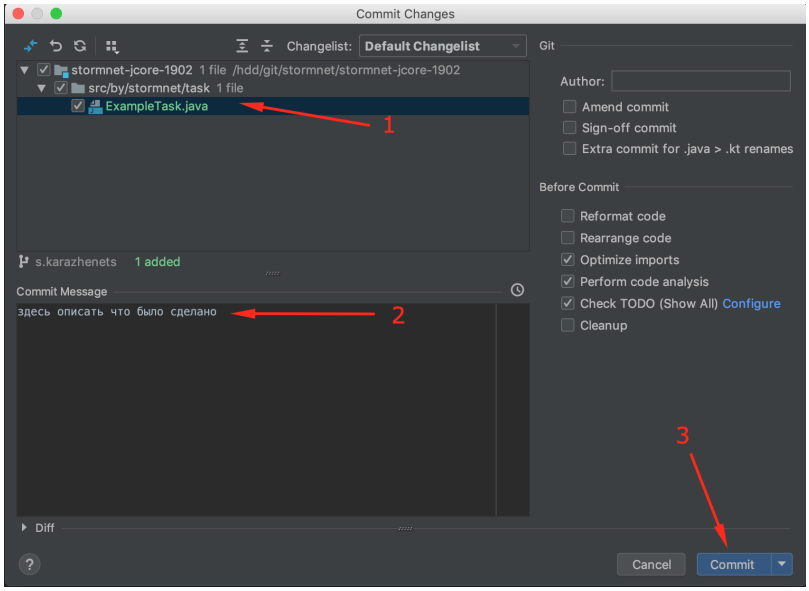
Загрузка заданий с github

Все новые (открытые) задания хранятся на github. Чтобы их получить себе в локальный репозиторий, нужно выполнить 2 шага:

1. **Сохранить в локальном репозитории все изменения файлов:**
   - в Idea из строки меню выбрать *VCS → Commit...*

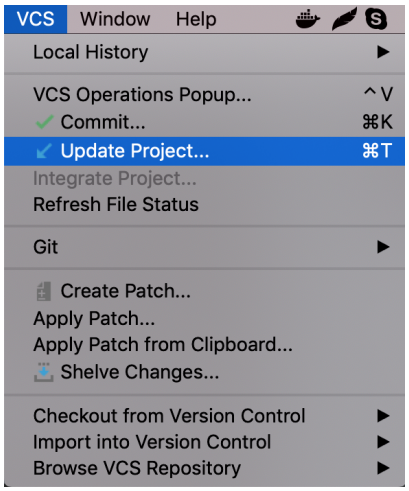


   - откроется окно

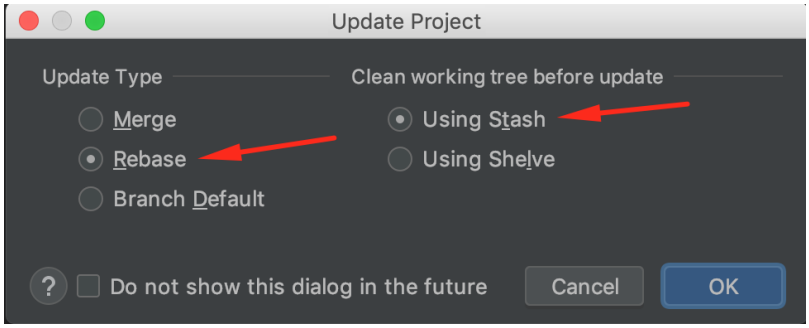


     - в этом окне нужно:
       - (1) отметить все сохраняемые файлы, если они еще не отмечены

        - (2) написать краткое сообщение о том, какие изменения были сделаны
        - (3) нажать кнопку *Commit*
2. **Получить самые свежие изменения файлов с github:**
    - из строки меню выбрать *VCS → Update Project...*



    - откроется окно

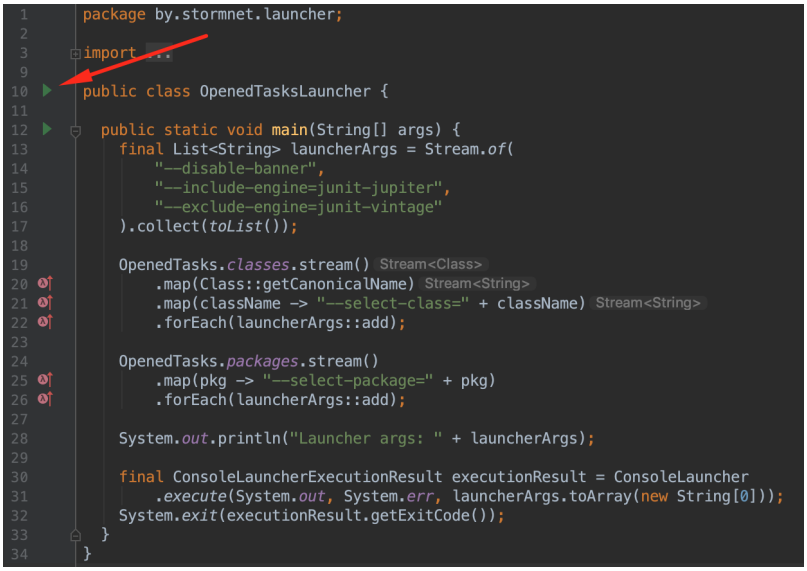


        - в этом окне:
            - выбрать Update Type — *Rebase*
            - выбрать Clean working tree before update выбрать — *Using Stash*
            - нажать кнопку *OK*

Запуск всех открытых заданий

После того, как новые задания будут успешно загружены в локальный репозиторий, нужно в Idea открыть файл *test/by/stormnet/launcher/OpenedTasksLauncher.java*.

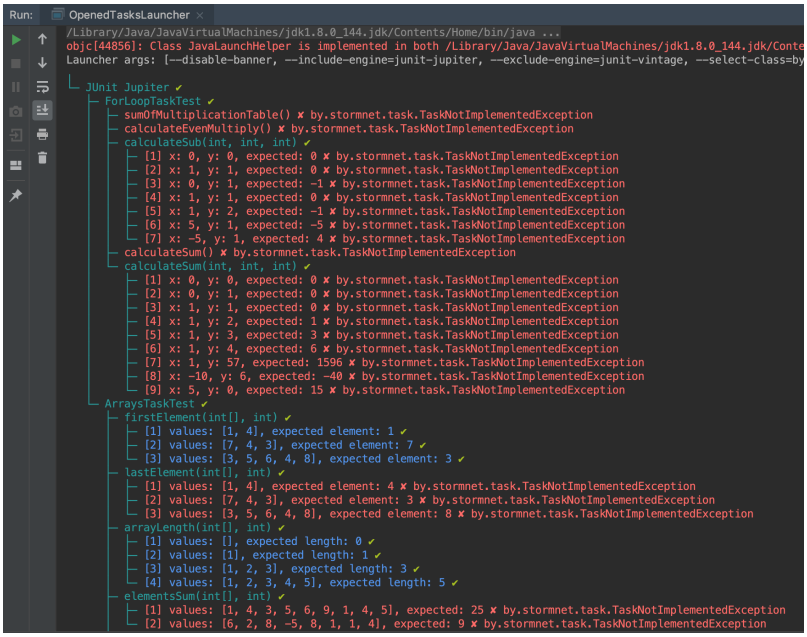
В этом файле написан код, который запустит все открытые задания (старые, которые уже ранее выполнялись, + новые задания):



```
package by.stormnet.launcher;

import ...

public class OpenedTasksLauncher {

  public static void main(String[] args) {
    final List<String> launcherArgs = Stream.of(
        "--disable-banner",
        "--include-engine=junit-jupiter",
        "--exclude-engine=junit-vintage"
    ).collect(toList());

    OpenedTasks.classes.stream() Stream<Class>
        .map(Class::getCanonicalName) Stream<String>
        .map(className -> "--select-class=" + className) Stream<String>
        .forEach(launcherArgs::add);

    OpenedTasks.packages.stream()
        .map(pkg -> "--select-package=" + pkg)
        .forEach(launcherArgs::add);

    System.out.println("Launcher args: " + launcherArgs);

    final ConsoleLauncherExecutionResult executionResult = ConsoleLauncher
        .execute(System.out, System.err, launcherArgs.toArray(new String[0]));
    System.exit(executionResult.getExitCode());
  }
}
```

Для запуска всех заданий нужно нажать на *зеленый* треугольник рядом с именем класса, затем в повившемся окошке нажать *Run*.

Результат запуска будет отбражен в следующем виде:



```
Run:  OpenedTasksLauncher
/Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java ...
objc[44856]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Conte
Launcher args: [--disable-banner, --include-engine=junit-jupiter, --exclude-engine=junit-vintage, --select-class=by

└ JUnit Jupiter ✔
  ├ ForLoopTaskTest ✔
  │ ├ sumOfMultiplicationTable() ✘ by.stormnet.task.TaskNotImplementedException
  │ ├ calculateEvenMultiply() ✘ by.stormnet.task.TaskNotImplementedException
  │ ├ calculateSub(int, int, int) ✔
  │ │ ├ [1] x: 0, y: 0, expected: 0 ✘ by.stormnet.task.TaskNotImplementedException
  │ │ ├ [2] x: 1, y: 1, expected: 0 ✘ by.stormnet.task.TaskNotImplementedException
  │ │ ├ [3] x: 0, y: 1, expected: -1 ✘ by.stormnet.task.TaskNotImplementedException
  │ │ ├ [4] x: 1, y: 1, expected: 0 ✘ by.stormnet.task.TaskNotImplementedException
  │ │ ├ [5] x: 1, y: 2, expected: -1 ✘ by.stormnet.task.TaskNotImplementedException
  │ │ ├ [6] x: 5, y: 1, expected: -5 ✘ by.stormnet.task.TaskNotImplementedException
  │ │ └ [7] x: -5, y: 1, expected: 4 ✘ by.stormnet.task.TaskNotImplementedException
  │ ├ calculateSum() ✘ by.stormnet.task.TaskNotImplementedException
  │ └ calculateSum(int, int, int) ✔
  │   ├ [1] x: 0, y: 0, expected: 0 ✘ by.stormnet.task.TaskNotImplementedException
  │   ├ [2] x: 0, y: 1, expected: 0 ✘ by.stormnet.task.TaskNotImplementedException
  │   ├ [3] x: 1, y: 1, expected: 0 ✘ by.stormnet.task.TaskNotImplementedException
  │   ├ [4] x: 1, y: 2, expected: 1 ✘ by.stormnet.task.TaskNotImplementedException
  │   ├ [5] x: 1, y: 3, expected: 3 ✘ by.stormnet.task.TaskNotImplementedException
  │   ├ [6] x: 1, y: 4, expected: 6 ✘ by.stormnet.task.TaskNotImplementedException
  │   ├ [7] x: 1, y: 57, expected: 1596 ✘ by.stormnet.task.TaskNotImplementedException
  │   ├ [8] x: -10, y: 6, expected: -40 ✘ by.stormnet.task.TaskNotImplementedException
  │   └ [9] x: 5, y: 0, expected: 15 ✘ by.stormnet.task.TaskNotImplementedException
  └ ArraysTaskTest ✔
    ├ firstElement(int[], int) ✔
    │ ├ [1] values: [1, 4], expected element: 1 ✔
    │ ├ [2] values: [7, 4, 3], expected element: 7 ✔
    │ └ [3] values: [3, 5, 6, 4, 8], expected element: 3 ✔
    ├ lastElement(int[], int) ✔
    │ ├ [1] values: [1, 4], expected element: 4 ✘ by.stormnet.task.TaskNotImplementedException
    │ ├ [2] values: [7, 4, 3], expected element: 3 ✘ by.stormnet.task.TaskNotImplementedException
    │ └ [3] values: [3, 5, 6, 4, 8], expected element: 8 ✘ by.stormnet.task.TaskNotImplementedException
    ├ arrayLength(int[], int) ✔
    │ ├ [1] values: [], expected length: 0 ✔
    │ ├ [2] values: [1], expected length: 1 ✔
    │ ├ [3] values: [1, 2, 3], expected length: 3 ✔
    │ └ [4] values: [1, 2, 3, 4, 5], expected length: 5 ✔
    ├ elementsSum(int[], int) ✔
    │ ├ [1] values: [1, 4, 3, 5, 6, 9, 1, 4, 5], expected: 25 ✘ by.stormnet.task.TaskNotImplementedException
    │ └ [2] values: [6, 2, 8, -5, 8, 1, 1, 4], expected: 9 ✘ by.stormnet.task.TaskNotImplementedException
```

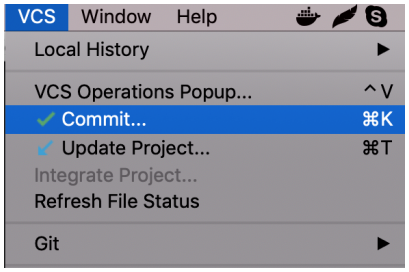
Здесь, если результат выполнения задания *красный*, то оно сделано не верно или вообще еще не сделано (новое задание). Если результат отбражен *синим* цветом, то задание выполнено верно.

Таким образом, список всех *красных* заданий это и есть все открытые задания, которые нужно делать. В этом списке могут быть как новые задания, так и старые задания, которые вы еще не успели доделать.

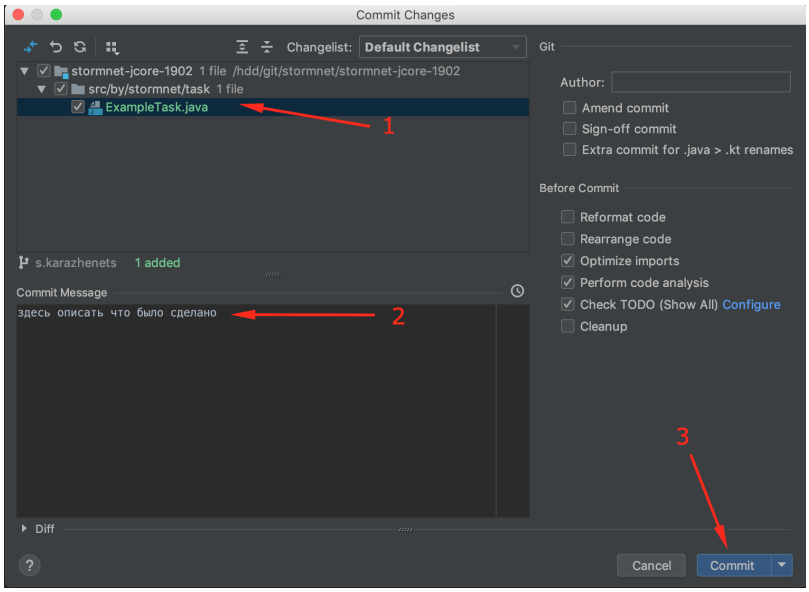
Отправка выполненных заданий на github

Нужно выполнить 3 шага:

1. **Сохранить в локальном репозитории все изменения файлов:**
   - в Idea из строки меню выбрать *VCS → Commit...*

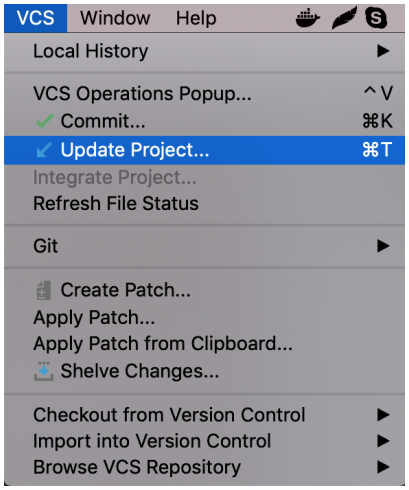


   - откроется окно

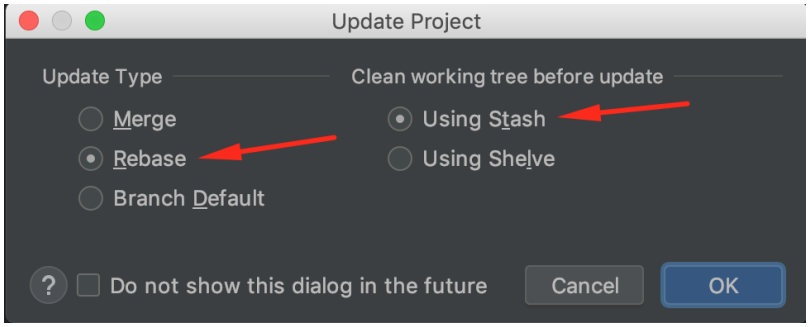


     - в этом окне нужно:
       - (1) отметить все сохраняемые файлы, если они еще не отмечены
       - (2) написать краткое сообщение о том, какие изменения были сделаны
       - (3) нажать кнопку *Commit*

2. **Получить самые свежие изменения файлов с github:**
   - из строки меню выбрать *VCS → Update Project...*

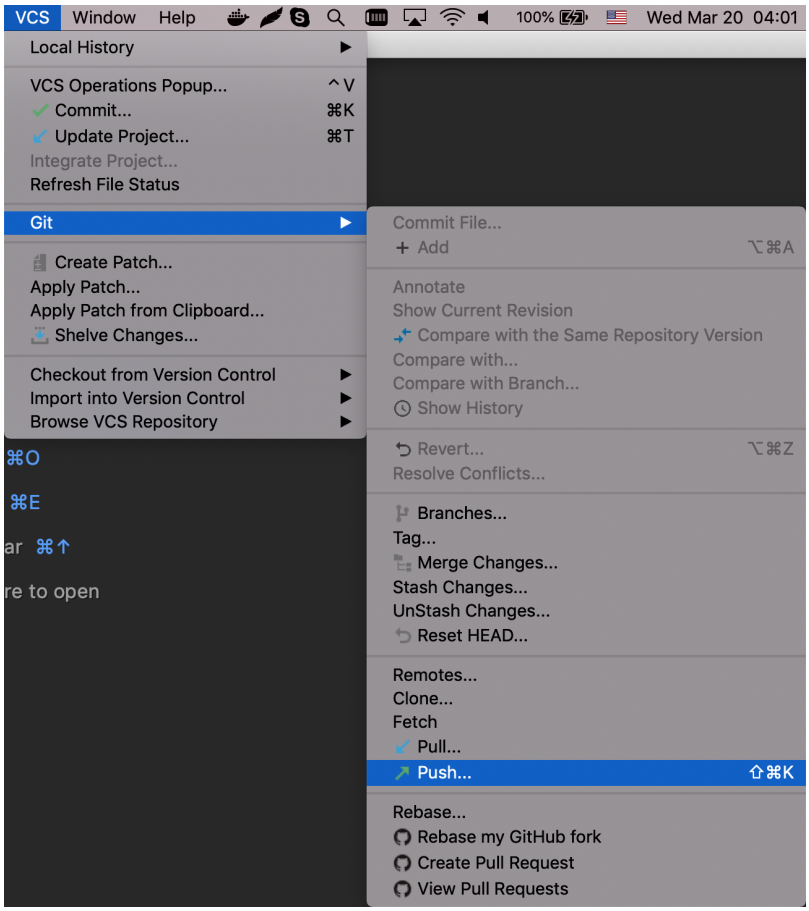


- откроется окно

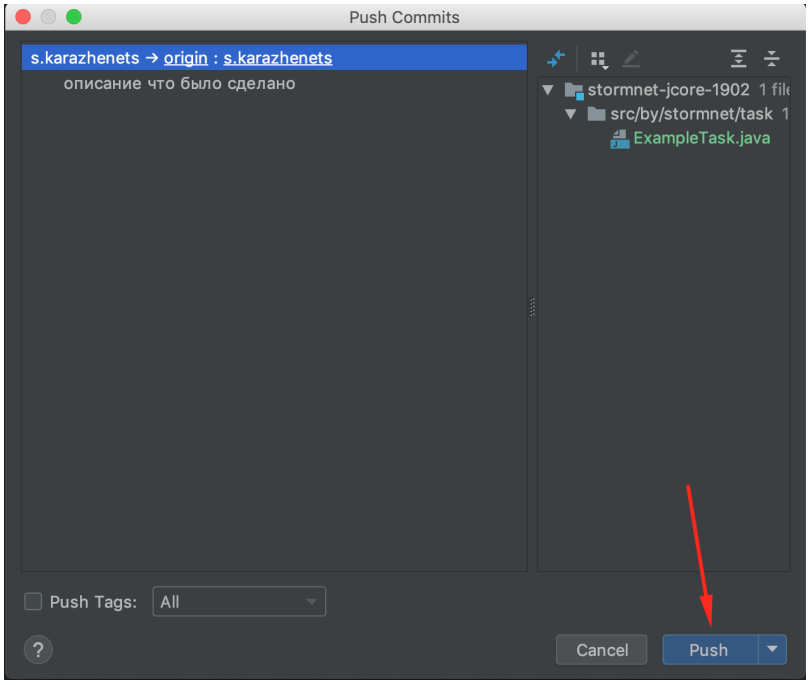


  - в этом окне:
    - выбрать Update Type — *Rebase*
    - выбрать Clean working tree before update выбрать — *Using Stash*
    - нажать кнопку *OK*

3. **Отправить на github все изменения файлов, сохраненные на 1-ом шаге:**
   - из строки меню выбрать *VCS → Git → Push*



   - откроется окно



- в этом окне нажать кнопку *Push*