Инструкция по выполнению марафона

Общая информация о курсе

Каждый день марафона состоит из изучения новых тем и выполнения задач по этим темам. Задачи составлены таким образом, чтобы вы применили как новые знания, так и навыки предыдущих дней. К задачам каждого дня прилагаются справочные материалы, которые содержат всю необходимую теорию.
Дни марафона различаются по нагрузке. Некоторые дни содержат много заданий, некоторые - мало. Это сделано для того, чтобы в “легкие” дни вы могли наверстать пропущенные задачи предыдущих дней. День 10 без заданий - “выходной”.

Что необходимо сделать до старта курса:

1) Установить среду разработки, если еще не установили. Вы можете использовать любую среду разработки, но мы очень рекомендуем IntelliJ Idea. Это одна из самых популярных и наиболее используемых сред разработки на сегодняшний день. Скачать бесплатную версию можно здесь: https://www.jetbrains.com/ru-ru/idea/download (версия Community).
2) Установить и настроить Git.
Этот шаг необходимо выполнить для того, чтобы вы могли отправлять свой код нам на проверку. Git - это стандартный инструмент, который используется программистами для того, чтобы делиться кодом или совместно разрабатывать проекты. В будущем, вы скорей всего будете активно использовать Git в своей работе, так что сейчас самое время познакомиться с этим инструментом.

Видео про то, что такое Git и зачем он нужен:
https://www.youtube.com/watch?v=Ov6SBXWDLxo

Инструкция по установке и настройке Git:
https://git-scm.com/book/ru/v2/Введение-Установка-Git

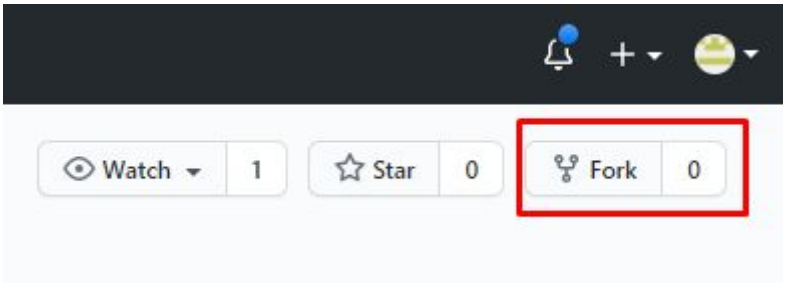
Видео про установку и конфигурацию Git: https://www.youtube.com/watch?v=h8B4Pmz8gKI

Отдельное видео по установке GIT для ОС Windows:
https://www.youtube.com/watch?v=12BIw4GdGYQ

3) Создать аккаунт на GitHub.
GitHub - это сайт, на который можно отправлять свой код, чтобы его увидели другие люди. Перейдите на https://github.com и создайте ваш аккаунт GitHub. На почту, указанную при

регистрации, будут приходить оповещения о проверке вашей работы. Не теряйте доступ к почте.

4) Скопировать себе в GitHub “костяк” проекта, перейдя по ссылке https://github.com/burlakovserge/JavaMarathon2020 и нажав на Fork в правом верхнем углу:



Fork переводится с английского как “вилка”, “развилка” или “ответвление”. Эта команда позволяет скопировать любой репозиторий к себе в GitHub аккаунт. По ссылке выше находится “костяк” Java проекта, где уже созданы необходимые классы и вся папочная структура. Внутри этого проекта вы будете реализовывать задачи каждого дня.
После того, как вы нажали на Fork, в вашем GitHub аккаунте появится проект с названием `JavaMarathon2020`. Этот проект - полная копия оригинального проекта. С этой копией вы и будете работать в рамках этого марафона.

5) Создать папку на вашем компьютере, где вы собираетесь хранить ваши Java проекты.
Эта папка может располагаться где угодно (хоть на рабочем столе) и называться по вашему усмотрению. Моя папка с Java проектами называется javaProjects и находится по такому пути:
`/Users/neil/javaProjects/`

В этой папке я храню все мои Java проекты, которые в свою очередь тоже являются папками.

6) Связать ваш компьютер с удаленным GitHub репозиторием `JavaMarathon2020`.
Перейдите на страницу репозитория, который вы “форкнули” к себе в аккаунт на шаге 4) и скопируйте ссылку из строки браузера. Ссылка имеет следующий вид:
`https://github.com/neil/JavaMarathon2020`
Обратите внимание, что ссылка содержит ваше имя на GitHub (помечено зеленым). Важно скопировать ссылку именно на тот репозиторий, который находится в **вашем** GitHub аккаунте.

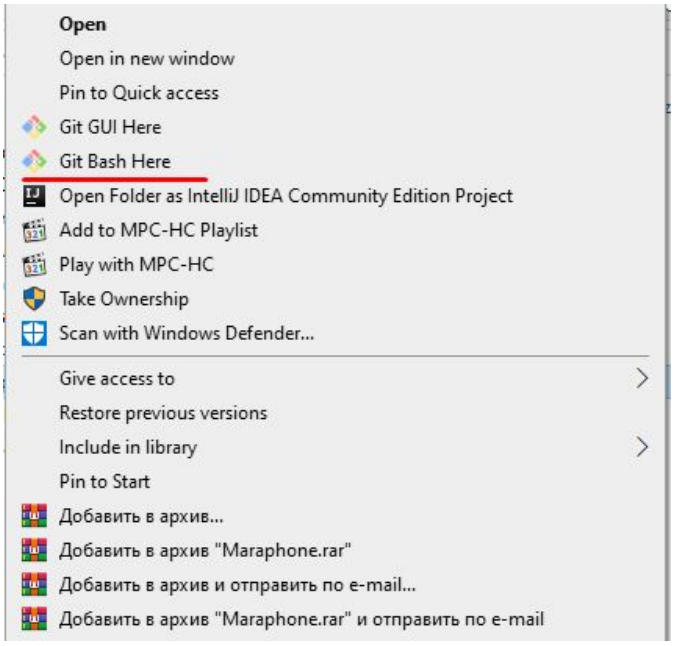
Далее все команды выполняются в терминале (в случае Linux или MacOS) и в Git Bash (в случае Windows)

В первую очередь, необходимо зайти в папку с вашими Java проектами, которую вы создали на шаге 5). Переходить по папкам в командной строке можно с помощью команды `cd`.
Моя папка с проектами находится по такому пути: `/Users/neil/javaProjects/`

Поэтому, чтобы перейти в эту папку, в командной строке я выполню команду:
`cd /Users/neil/javaProjects/`

Пример возможного пути до папки с проектами в ОС Windows:
`C:\IdeaProjects\javaProjects`

В ОС Windows можно сразу открыть Git Bash в нужной папке, нажав на нее правой кнопкой и в выпадающем списке выбрав “Git Bash Here”



Теперь, когда мы зашли внутрь папки с нашим проектом, в ней необходимо выполнить команду:
`git clone [ранее скопированная ссылка на удаленный репозиторий]`

Пример:
`git clone https://github.com/neil/JavaMarathon2020`
Эта команда копирует файлы из удаленного GitHub репозитория в папку на жесткий диск вашего компьютера (появится папка `JavaMarathon2020`). Помимо этого, эта команда устанавливает “связь” между вашим компьютером и удаленным GitHub репозиторием. Поэтому те изменения, которые вы делаете в папке `JavaMarathon2020` на компьютере, можно будет отправить на удаленный GitHub репозиторий.

7) Готово! Конфигурация Git завершена. Теперь вы сможете отправлять свой код на GitHub. Но отправлять еще нечего, поэтому давайте решим первую задачу.

Во-первых, необходимо открыть папку `JavaMarathon2020` в среде разработки. В IntelliJ Idea это делается просто: File -> Open -> укажите путь до папки `JavaMarathon2020`.

После того, как проект будет открыт в среде разработки, вам необходимо перейти в пакет `day0`. В этом пакете содержится класс `Task1`. В этом классе, в методе `main()` вы и должны реализовать первую задачу этого марафона.

`Task1`: Написать программу, которая выводит в консоль сообщение “`Hello, world`!”.
Пример решенной задачи:

```java
package day0;

public class Task1 {
    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

Теперь, когда задача решена, пора отправить решение на ваш GitHub репозиторий.

Можно работать с Git через командную строку или через IntelliJ Idea. Используйте удобный вам способ. В этой инструкции мы покажем работу с Git через командную строку.
На моем компьютере, проект `JavaMarathon2020` находится по такому пути:
`/Users/neil/javaProjects/JavaMarathon2020`

Если я открыл командную строку (или Git Bash) впервые, я должен перейти в эту папку с проектом:
`cd /Users/neil/javaProjects/JavaMarathon2020`

Находясь в папке с моим проектом, вызвав команду `git status` (эта команда сообщает текущее состояние репозитория), я увижу следующее:

```
JavaMarathon2020 $ git status
На ветке master

Еще нет коммитов

Неотслеживаемые файлы:
  (используйте «git add <файл>…», чтобы добавить в то, что будет включено в коммит)
        .DS_Store
        .idea/
        JavaMarathon2020.iml
        src/
```

Git увидел изменения в локальном репозитории (на жестком диске) и эти изменения мы можем зафиксировать и отправить на удаленный репозиторий.

Для того, чтобы зафиксировать изменения, в локальном репозитории необходимо выполнить команду (точка в конце обязательна!):
`git add .`

И после этого сделать так называемый коммит, выполнив команду:
`git commit -m “Сообщение вашего коммита”`

В кавычках необходимо указать сообщение коммита, которое лучше всего описывает изменения, которые вы сделали. В нашем случае, мы решили задачу `Task1`, поэтому наше сообщение коммита может быть таким: “`Add Task1 solution`”. Сообщения коммита обычно пишутся на английском языке.

Мы зафиксировали наши изменения, сделали так называемый “слепок” нашего кода (коммит). Для этого достаточно было выполнить две команды - `git add .` и `git commit`
На данном этапе не обязательно глубоко понимать, как работают эти две команды, но если вам интересно, в этом видео все подробно рассказывается:
https://www.youtube.com/watch?v=yZISr7LtlKQ

Теперь осталось сделать последнюю вещь - отправить ваш код на удаленный репозиторий, чтобы мы могли его посмотреть.

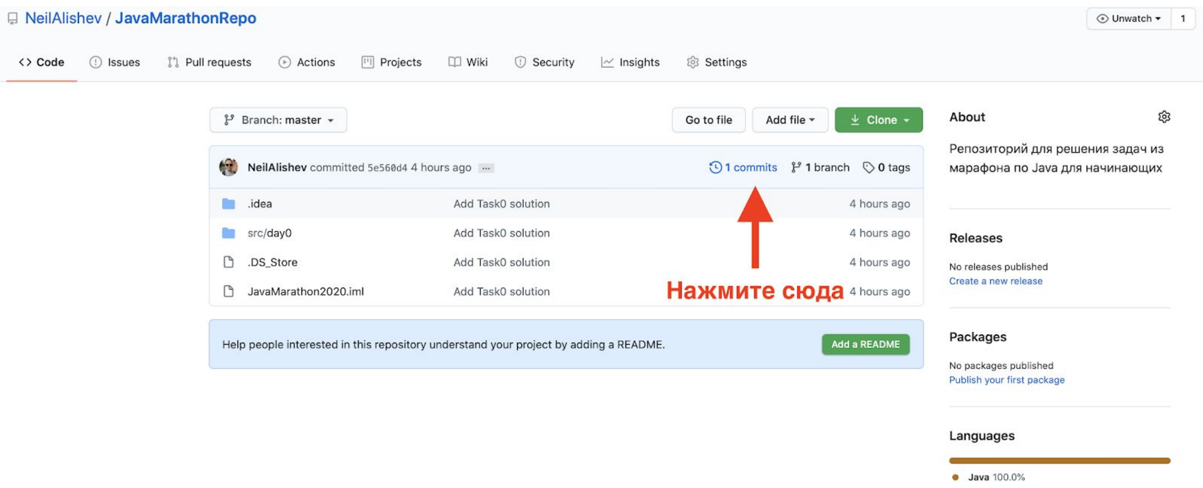
Для этого достаточно выполнить одну простую команду:
`git push origin master`

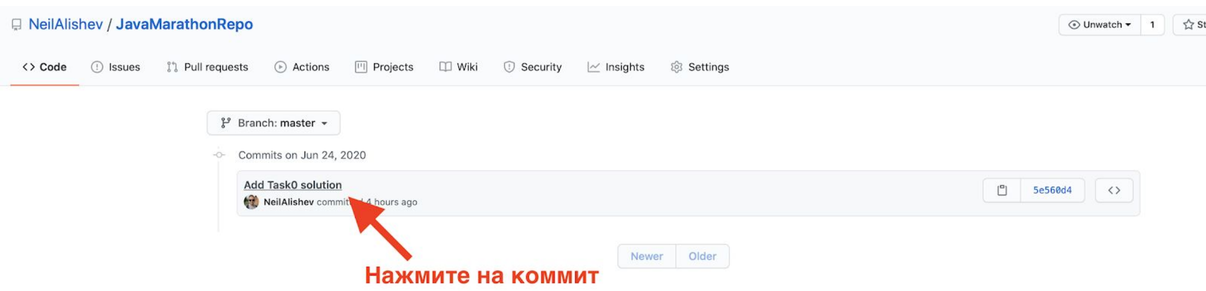
Команда попросит вас ввести данные аккаунта GitHub и отправит код на удаленный репозиторий GitHub.

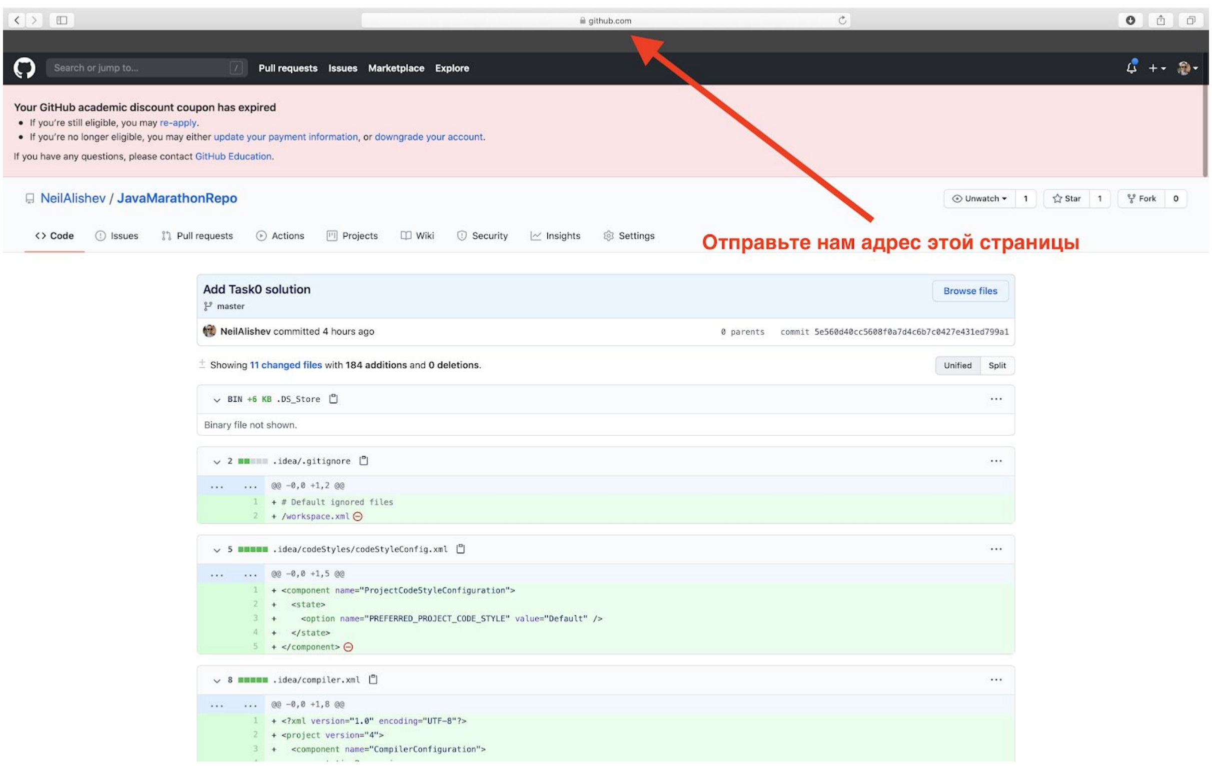
Браво! Вы постигли азы GIT и отправили ваш код на удаленный репозиторий.

```
JavaMarathon2020 $ git add .
JavaMarathon2020 $ git commit -m "Add Task0 solution"
[master (корневой коммит) 5e560d4] Add Task0 solution
 11 files changed, 184 insertions(+)
 create mode 100644 .DS_Store
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/codeStyles/codeStyleConfig.xml
 create mode 100644 .idea/compiler.xml
 create mode 100644 .idea/encodings.xml
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/uiDesigner.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 JavaMarathon2020.iml
 create mode 100644 src/day0/Task0.java
JavaMarathon2020 $ git push origin master
Username for 'https://github.com': NeilAlishev
Password for 'https://NeilAlishev@github.com':
Перечисление объектов: 17, готово.
Подсчет объектов: 100% (17/17), готово.
При сжатии изменений используется до 12 потоков
Сжатие объектов: 100% (13/13), готово.
Запись объектов: 100% (17/17), 3.31 KiB | 1.66 MiB/s, готово.
Всего 17 (изменения 0), повторно использовано 0 (изменения 0)
To https://github.com/NeilAlishev/JavaMarathonRepo.git
 * [new branch]      master -> master
JavaMarathon2020 $
```

Осталось только скопировать URL страницы с коммитом и отправить ее через форму https://forms.gle/dh4dZaCUtZb2sUQT9. Мы увидим ваш код и сможем дать комментарий. Как найти коммит объясняется в картинках ниже.


NeilAlishev / JavaMarathonRepo
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Branch: master
Go to file
Add file
Clone
NeilAlishev committed 5e560d4 4 hours ago
1 commits
1 branch
0 tags
.idea
Add Task0 solution
4 hours ago
src/day0
.DS_Store
JavaMarathon2020.iml
Нажмите сюда
Help people interested in this repository understand your project by adding a README.
Add a README
About
Репозиторий для решения задач из марафона по Java для начинающих
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
Java 100.0%


NeilAlishev / JavaMarathonRepo
Branch: master
Commits on Jun 24, 2020
Add Task0 solution
5e560d4
Newer
Older
Нажмите на коммит

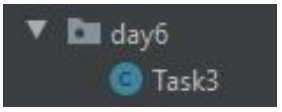

github.com
Your GitHub academic discount coupon has expired
NeilAlishev / JavaMarathonRepo
Отправьте нам адрес этой страницы
Add Task0 solution
Browse files
master
NeilAlishev committed 4 hours ago
0 parents
commit 5e560d40cc5608f0a7d4c6b7c0427e431ed799a1
Showing 11 changed files with 184 additions and 0 deletions.
Unified
Split
.DS_Store
Binary file not shown.
.idea/.gitignore
.idea/codeStyles/codeStyleConfig.xml
.idea/compiler.xml

8) Теперь осталось лишь посмотреть наш комментарий к вашему commit'у :) Каждый раз, когда вы будете пушить новый коммит с решением задач и отправлять ссылку на этот коммит, мы будем смотреть ваш код и давать комментарии по нему. Этот процесс по-умному называется code review (код ревью - англ. "обзор кода").

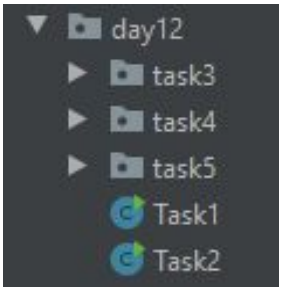
Ссылка на гугл форму будет дублироваться каждый день вместе с заданиями. Если на любом этапе возникнут трудности, пишите в общий чат - мы обязательно поможем!

Инструкция по выполнению и отправке задач

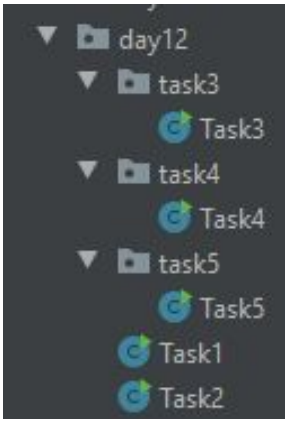
1) Вы будете добавлены в Telegram канал и чат. В канале мы ежедневно в 8:00 МСК будем публиковать задачи текущего дня и решение к задачам предыдущего. В чате вы можете задавать вопросы и обсуждать сложности, с которыми сталкиваетесь в процессе решения.
2) Проект, в котором вам предстоит решать задачи, уже содержит структуру папок и классы для каждого из дней марафона. Пакеты имеют название dayN, где N - номер текущего дня марафона. Классы имеют название TaskN, где N - номер задачи. Например, если мы решаем третью задачу шестого дня, находим класс `Task3` в пакете `day6` и в нем решаем задачу:



Некоторые дни потребуют от вас самостоятельно создавать классы, например день 12 содержит следующую структуру:

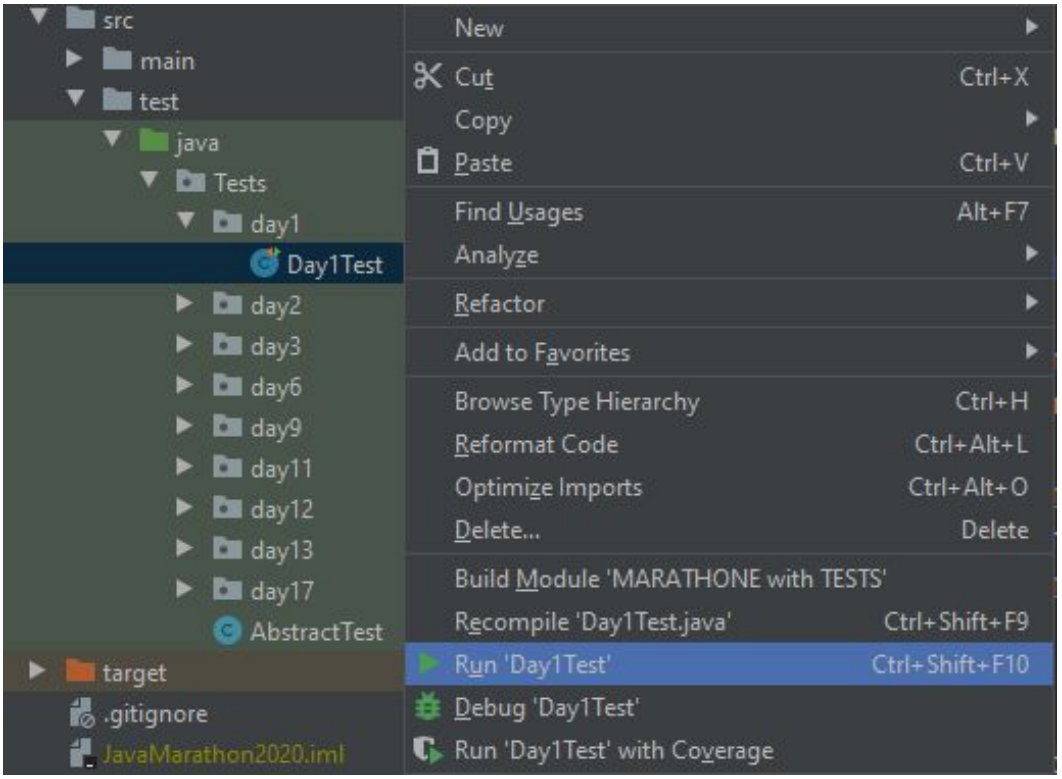


Это означает, что задачи 1 и 2 необходимо выполнять в классах `Task1` и `Task2`, а для решения задач 3,4,5 необходимо создать классы в соответствующих подпапках.

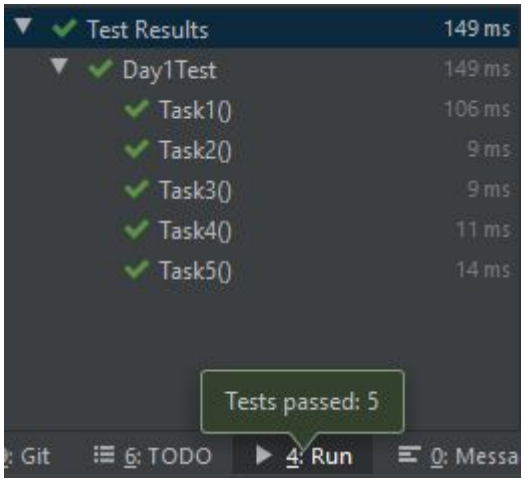


3) В проекте реализованы unit-тесты. Они позволяют проверить ваше решение мгновенно, еще до отправки на code review. Тесты находятся в папке `src/test/java/tests` и разделены по дням и заданиям. По умолчанию все тесты, кроме дня 1, отключены (закомментированы). Вам необходимо самостоятельно снимать комментарии с нужного теста, если вы захотите его запустить. Проверять свой код тестами желательно, но не обязательно. Тесты нужны в первую очередь для вас, чтобы убедиться в правильности вашего решения. Тест - это обычный класс. Чтобы убрать все комментарии в классе за раз, можно выбрать весь код с помощью сочетания клавиш `ctrl + A (command + A на macOS)` и убрать комментарии с помощью сочетания клавиш `ctrl + / (command + / на macOS)`.
   Сочетания клавиш работают на английской раскладке клавиатуры.

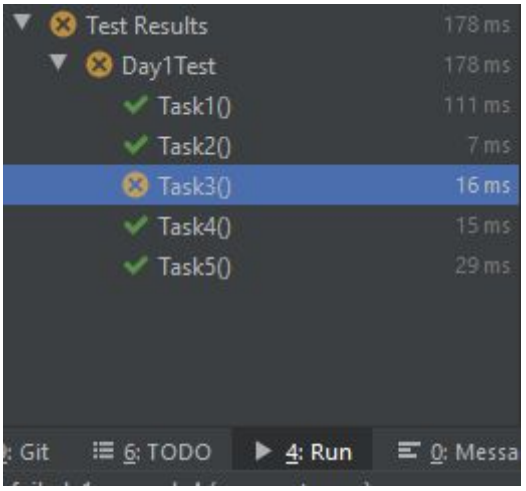
Пример использования тестов:
Для запуска теста Дня 1, нажать ПКМ на `Day1Test` и в выпадающем списке выбрать `Run 'Day1Test'`



После выполнения тестов вы увидите одну из следующих ситуаций:



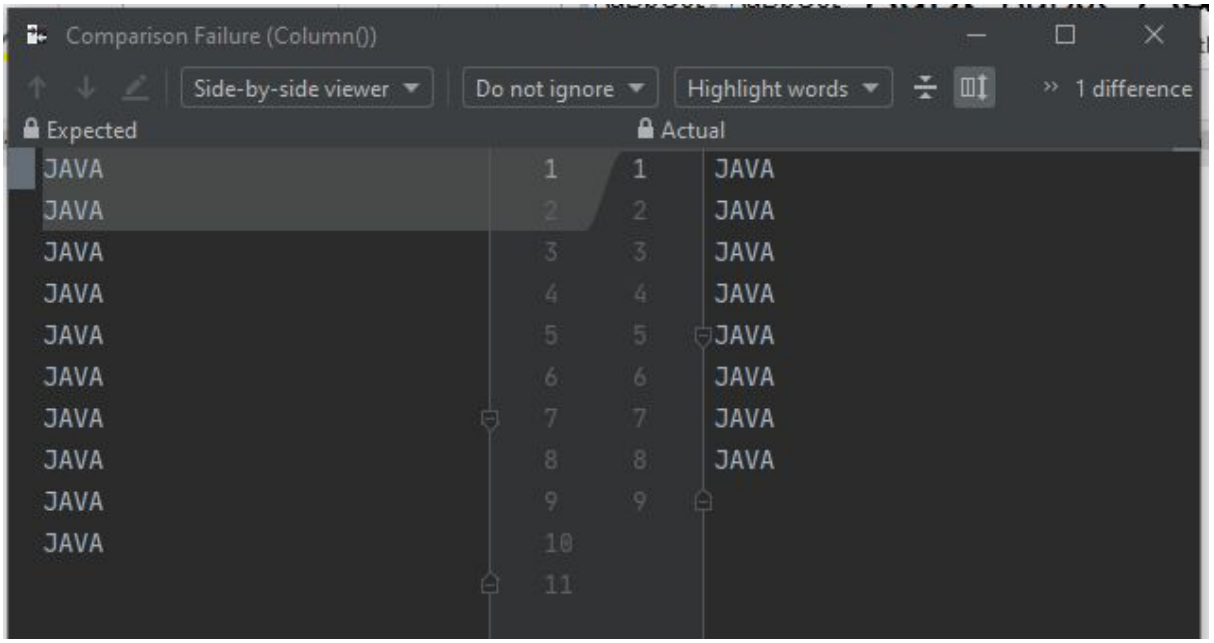
Это означает, что все тесты прошли успешно



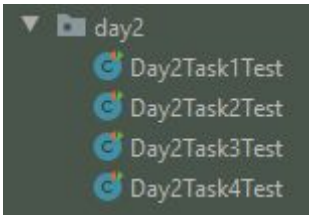
Один из тестов (для 3 задачи первого дня) «не прошел», значит есть ошибка

Из названия тестового метода видно, что ошибка в задании №3 (класс `Task3`). В консоли также выводится дополнительная информация об ошибке:

Сообщение об ошибке содержит описание “Количество повторений слова JAVA не равно десяти строкам”, а при нажатии на `<Click to see difference>` можно увидеть сравнение результатов expected / actual (ожидание / реальность)



Делаем вывод, что ожидается 10 строк “`JAVA`”, а при текущей реализации цикла получили 8. Значит необходимо доработать цикл. Возвращаемся в класс `Task3`, вносим изменения, снова запускаем тесты. Дорабатываем наш класс до тех пор, пока все тесты не пройдут.
В примере выше, тесты для всех задач Дня 1 были размещены в одном классе. Для некоторых дней тесты устроены по-другому: для тестирования каждой задачи отводится отдельный класс.
Пример (тесты Дня 2):

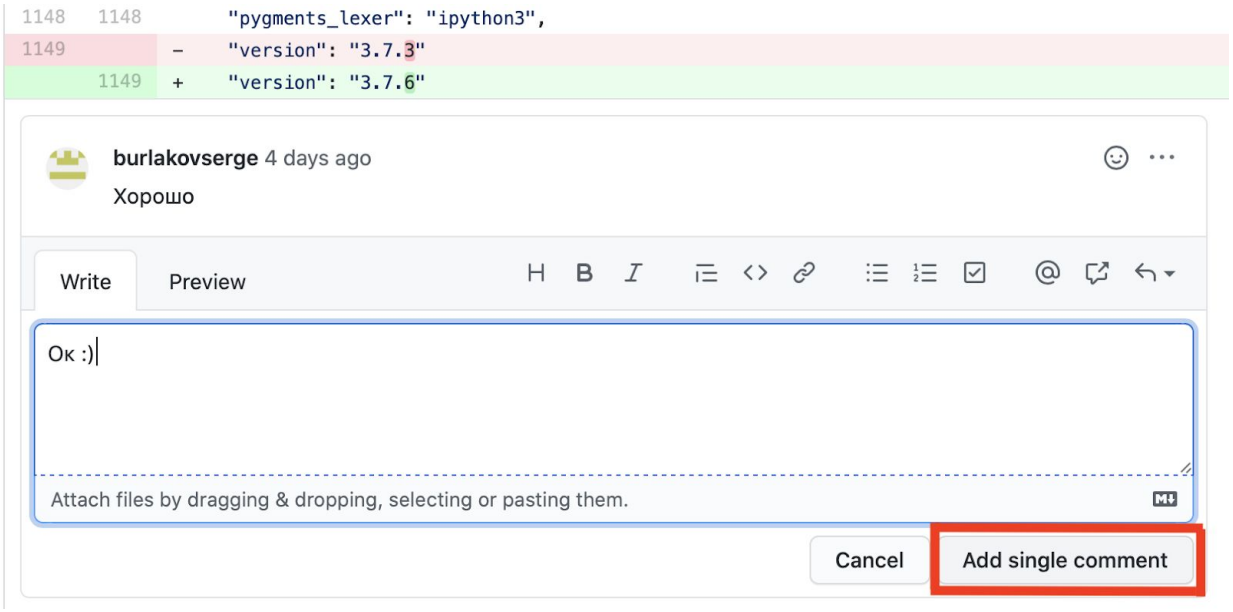


В этом случае можно запускать как отдельно каждый тест, так и все вместе для Дня 2, нажав ПКМ на пакете `day2` и выбрав пункт `Run 'All Tests'`.

Для заданий дней 4, 5, 15 тестов нет.

4) Как только задачи текущего дня решены, вы должны закоммитить изменения, запушить их на удаленный репозиторий и отправить нам ссылку на ваш коммит через гугл форму (https://forms.gle/dh4dZaCUtZb2sUQT9). Название коммита должно быть таким: `“complete dayN tasks”`, где N - номер текущего дня.
5) Отправка ссылки на коммит обязательна для получения код ревью. Если вы запушили коммит, но не отправили на него ссылку через гугл форму, код ревью сделан не будет.
После отправки ссылки на коммит, через некоторое время мы сделаем код ревью и

оставим комментарии на странице коммита. На почту, указанную при регистрации аккаунта GitHub, придет оповещение об этом.
Если возникает вопрос по поводу комментария, оставляйте его там же, на странице коммита, в ответ на наш комментарий:



Code review

Зачем он нужен если есть тесты?
Решение может быть верное, тесты все успешные, но сам подход не рациональный. Частые ситуации - это использование дополнительных переменных, без которых все так же хорошо будет работать, не очевидное именование переменных, использование нескольких циклов вместо одного с грамотным условием и др. Для выявления всех этих ошибок и проводится code review.
Если все верно и грамотно реализовано, последует короткий ответ «ок» или «верно». Это означает, что работа «принимается» и дорабатывать ничего не надо.

Что делать после проверки?
Если замечания не понятны – задать вопрос, ответив на наш комментарий на странице коммита (на нашу почту тоже приходит письмо-оповещение). Если понятно, то внести изменения и повторно не присылать задание с исправлениями.

А если не все тесты «успешны», присылать коммит?
Если решить не получается, пишите в чат. Там всегда находится кто-то на помощь. Лучше формулировать вопрос, описав что было сделано и что не получается, чем написать «как решить ... задание?». При долгом отсутствии ответа со стороны других марафонцев, обращайте внимание пользователя @java_marathon_admin, обязательно через символ «@». Старайтесь решать задания так, чтобы тесты проходили.

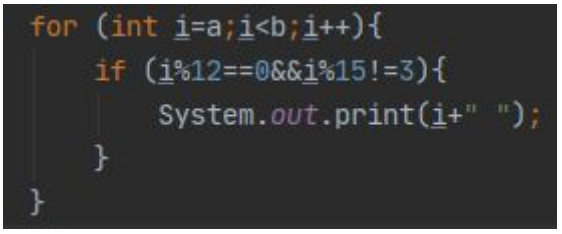
В гугл форме есть вопрос “тесты прошли?”. Выбирая “нет”, оставьте комментарий на странице коммита под тем заданием / методом, где тесты не прошли, с описанием ошибки. Это поможет в ходе обсуждения на GitHub вместе решить задание.

Я не успел прислать «день в день», можно с опозданием?
Да. Можно присылать на ревью решения любых дней до окончания марафона.

Каждый день, начиная со второго, вместе с новым заданием, в чате публикуется видео с решением задач прошедшего дня, с подробными комментариями. Присылайте свои решения в любое время, в течение всего марафона, хоть несколько дней сразу. Проверка проводится регулярно, вечером по мск.

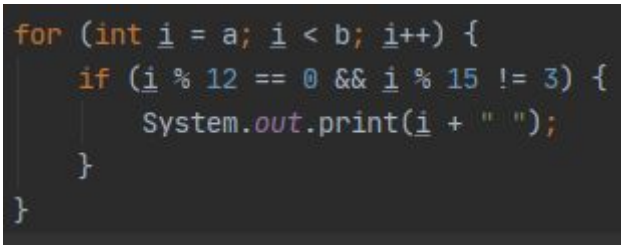
Общие ошибки

1. Нет отступов между переменными и операторами



```
for (int i=a;i<b;i++){
    if (i%12==0&&i%15!=3){
        System.out.print(i+" ");
    }
}
```

Это трудно читаемо, используйте пробелы, а еще лучше автоформатирование кода в intelliJ idea, горячие клавиши `ctrl + alt + L` (`command + option + L` на MacOS) на en раскладке.



```
for (int i = a; i < b; i++) {
    if (i % 12 == 0 && i % 15 != 3) {
        System.out.print(i + " ");
    }
}
```

2. Имена переменных неинформативны.

Для переменных, хранящих сумму, количество, максимальное число, минимальное число, год, цвет и т.д., правильные имена: `sum, count, max, min, year, color`.

Неправильные (неинформативные) имена: `a, b, c, x, x1, s2`.
Неинформативных имен переменных надо избегать.

3. Пустые строки везде и много.

Пустыми строками отделяются друг от друга методы, логические блоки в классе или в методе, при этом количество пустых строк = 1 и не более.

4. Вывод в консоль.

Существует три статических метода класса System для вывода в консоль:
`System.out.println()` - добавляет перенос на новую строку.
`System.out.print()` - не добавляет перенос на новую строку.
`System.out.printf()` - позволяет выводить и сразу форматировать сообщение, но не добавляет перенос строки. Чтобы добавить перенос используйте %n вместо \n.

Удачи!