

Лабораторная работа №1

Код программы продемонстрирован на рисунках 1,2,3:

1) Класс CountController:

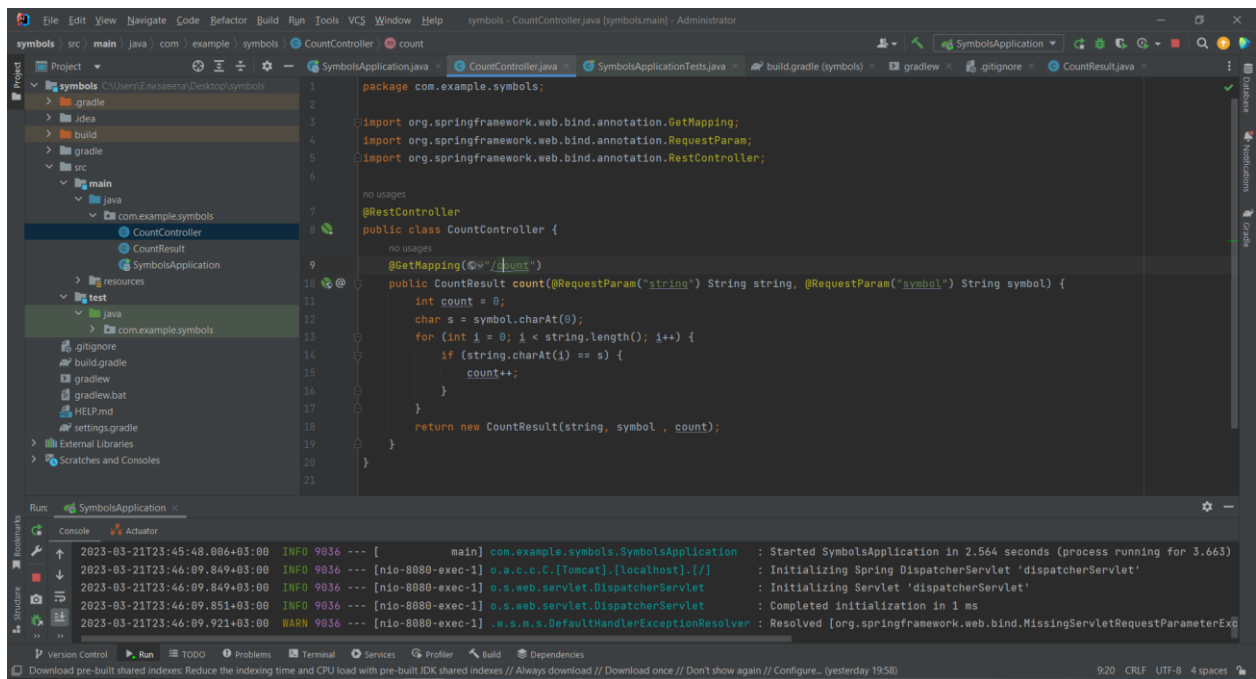


Рисунок 1

2) Класс CountResult:

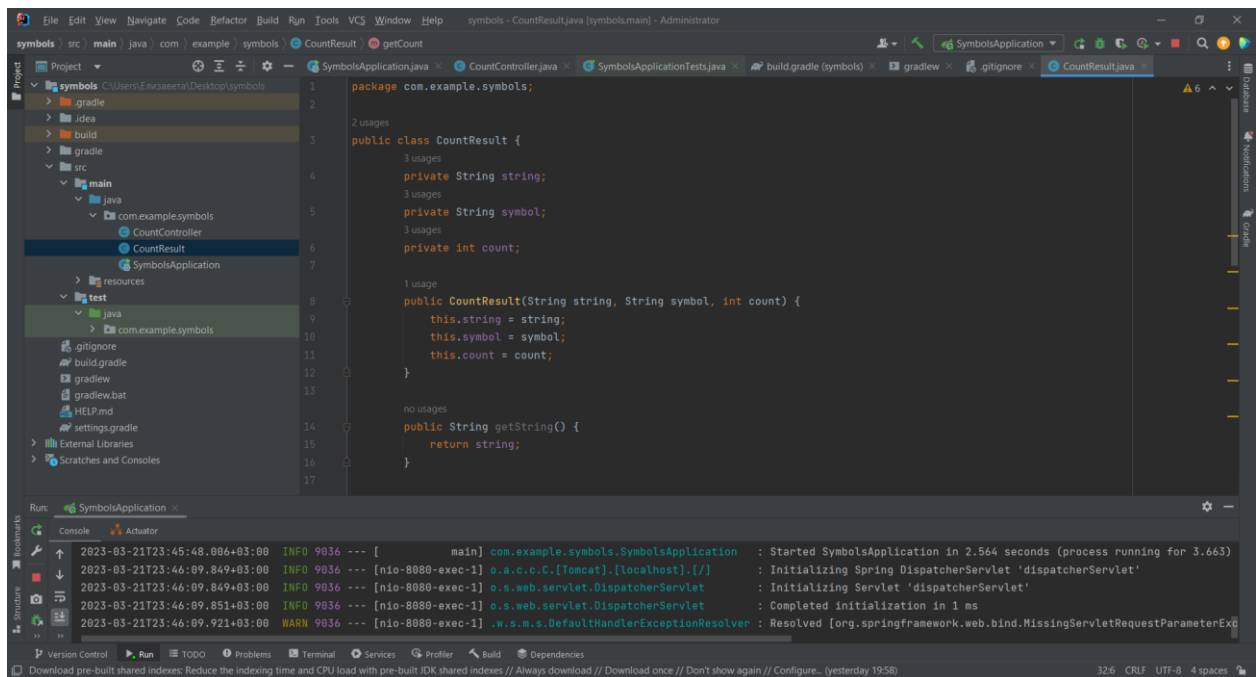


Рисунок 2

Продолжение данного класса:

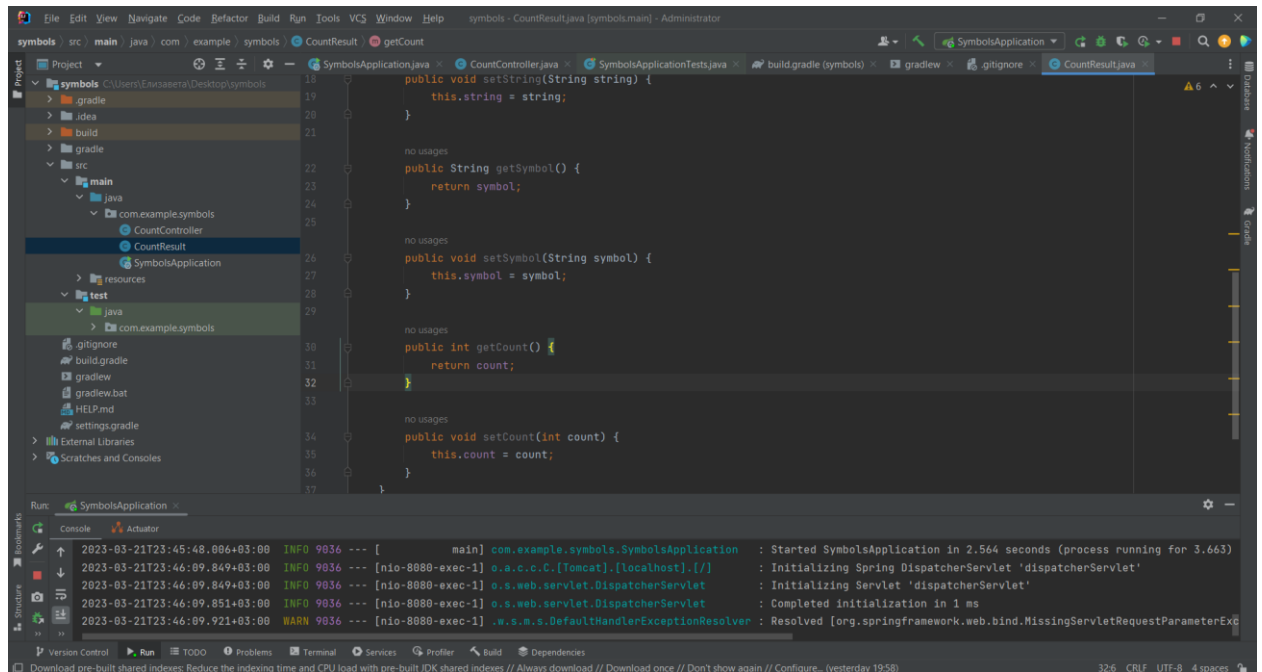


Рисунок 3

После запуска приложения, переходя по адресу <http://localhost:8080/count?string=yourstring&symbol=y> в веб-браузере, где `yourstring` - это строка, в которой по заданию нужно найти символ, а `y` - символ, который будем искать и подсчитывать, сколько раз он встречается в данной строке. Далее видим ответ в формате JSON в браузере.

Скриншоты, на которых продемонстрирована работа программы (см. рис. 4,5,6,7):



Рисунок 4



Рисунок 5



Рисунок 6



Рисунок 7

Работа программы через Postman (см. рис.8):

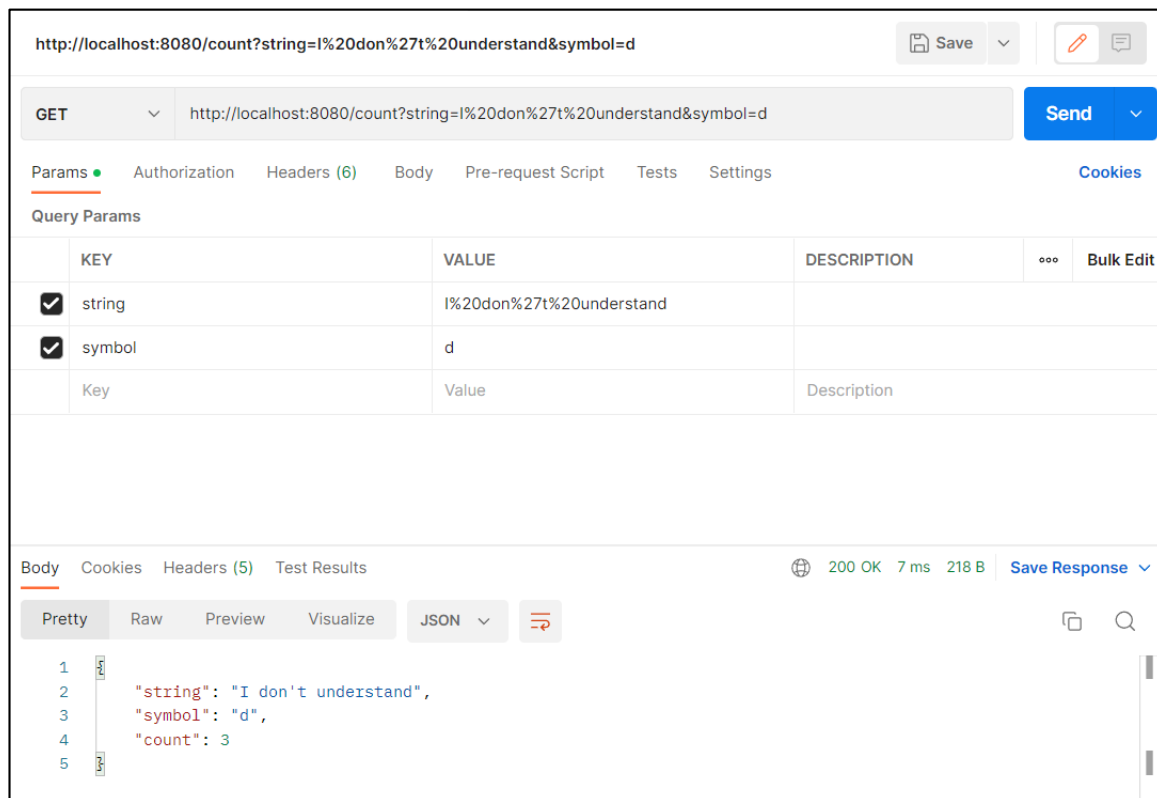


Рисунок 8

Лабораторная работа №2

В данном коде были добавлены:

- Валидация входных параметров с возвращением 400 ошибки;
- Обработка внутренних unchecked ошибок с возвратом 500 ошибки;
- Логирование действий и ошибок

Код изображён на рисунках 9,10

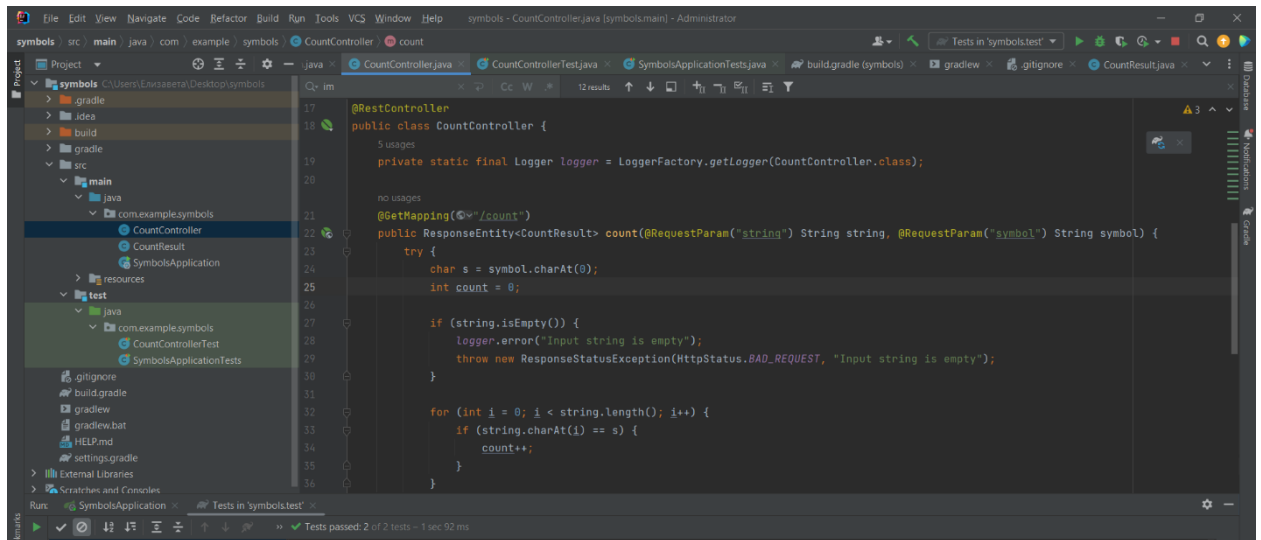


Рисунок 9

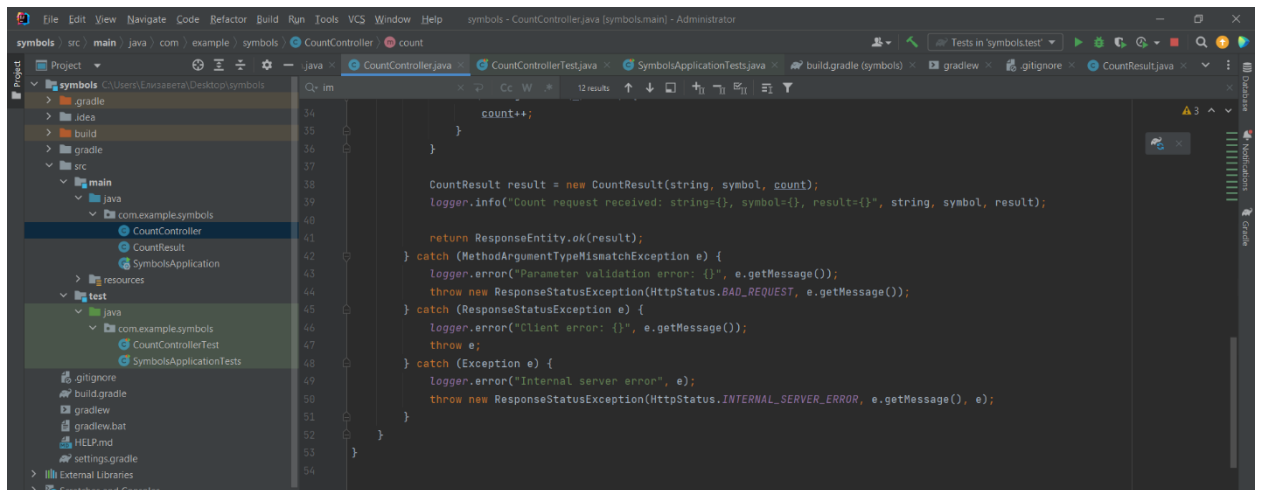


Рисунок 10

Скриншоты, на которых продемонстрирована работа программы (см. рис. 11,12,13,14):

400 ошибка:

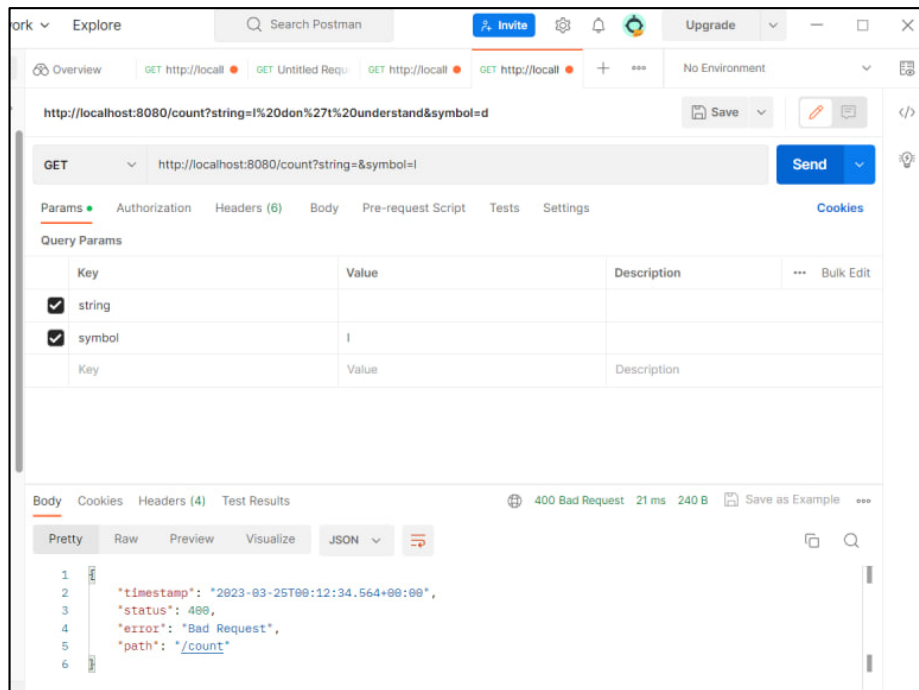


Рисунок 11

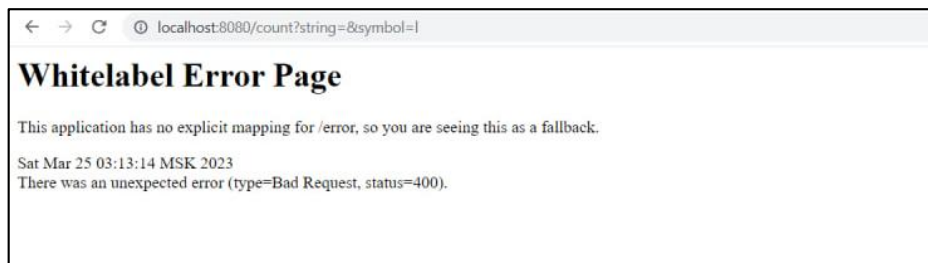


Рисунок 12

500 ошибка:

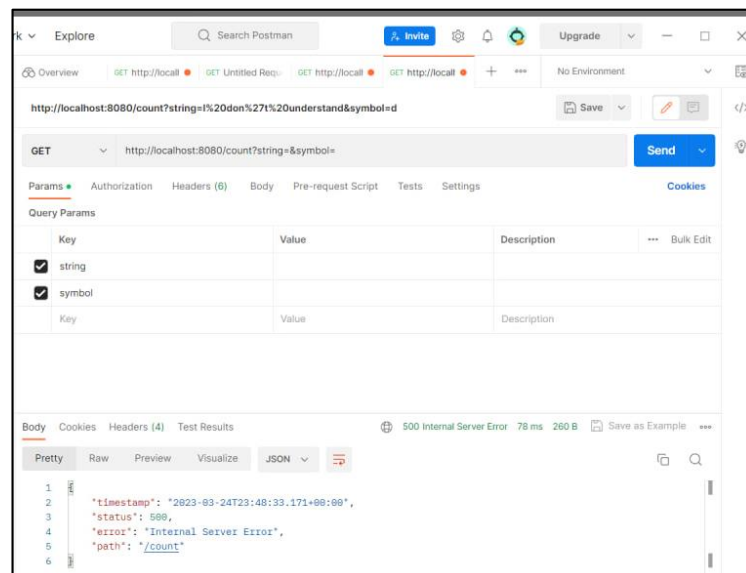


Рисунок 13



Рисунок 14

Далее отображено выполнение задания: написать unit test. См. рис.15,16,17.

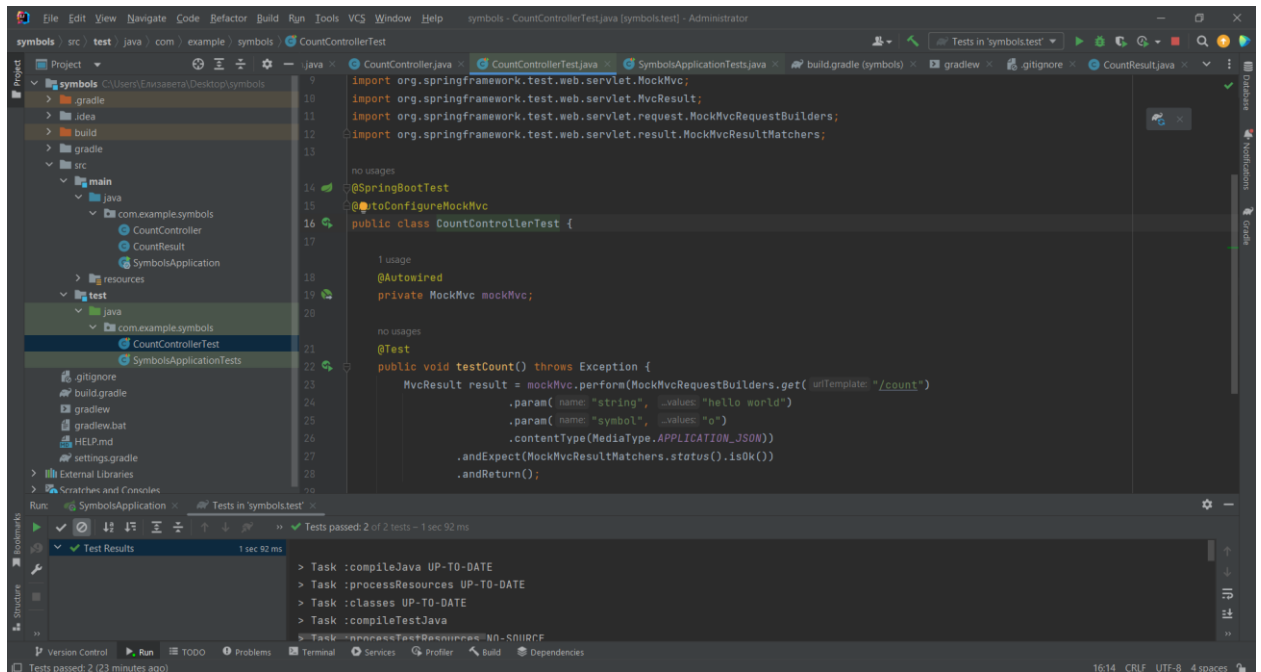


Рисунок 15

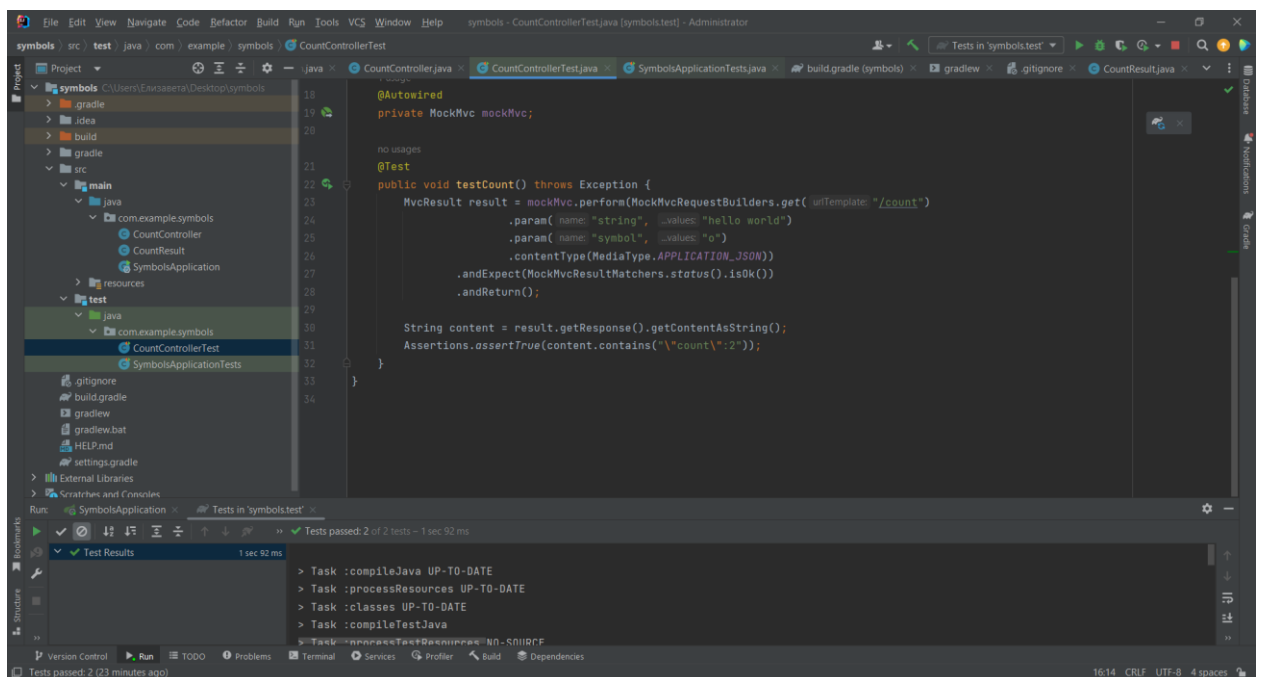


Рисунок 16

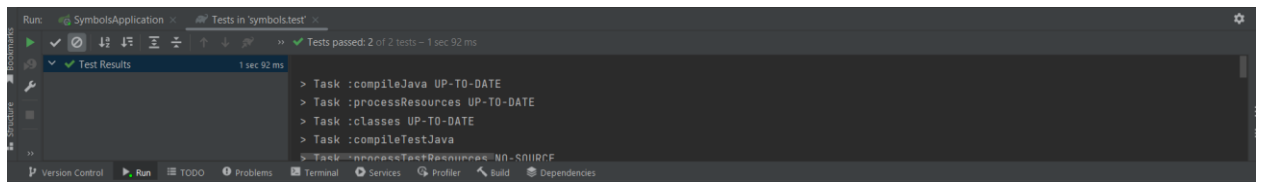


Рисунок 17

Лабораторная работа №3

1) Класс CountController (см. рис.18, 19,20,21):

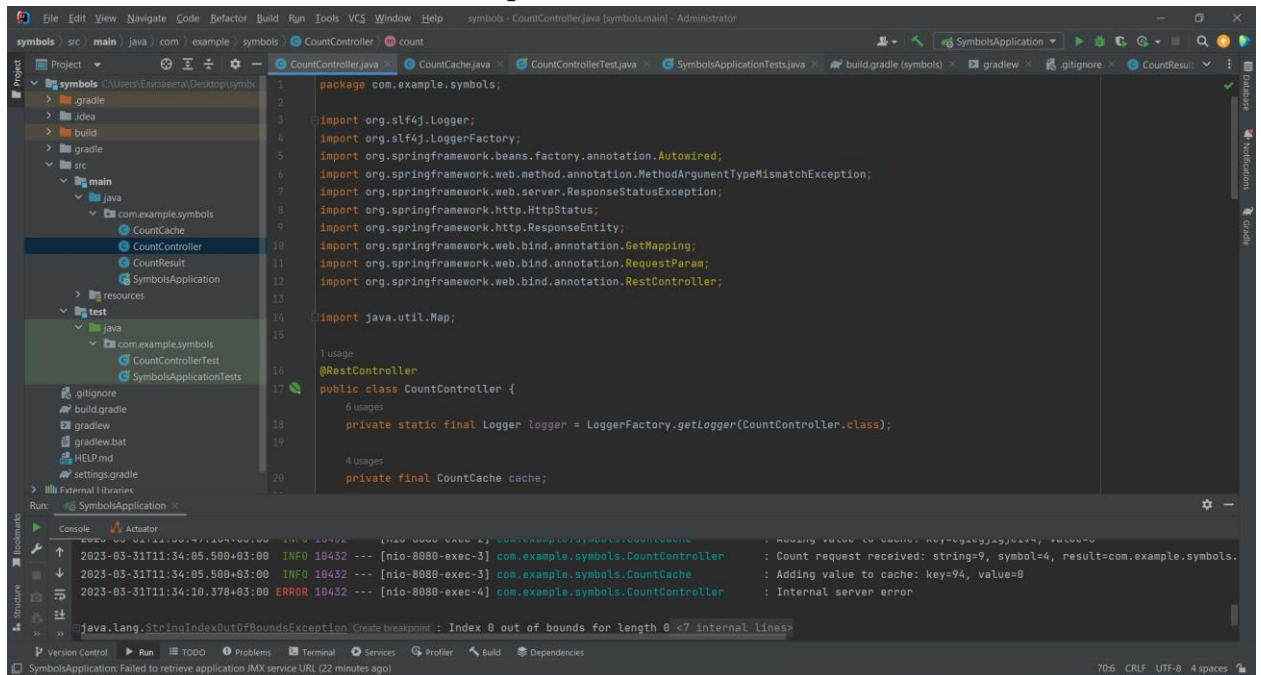


Рисунок 18

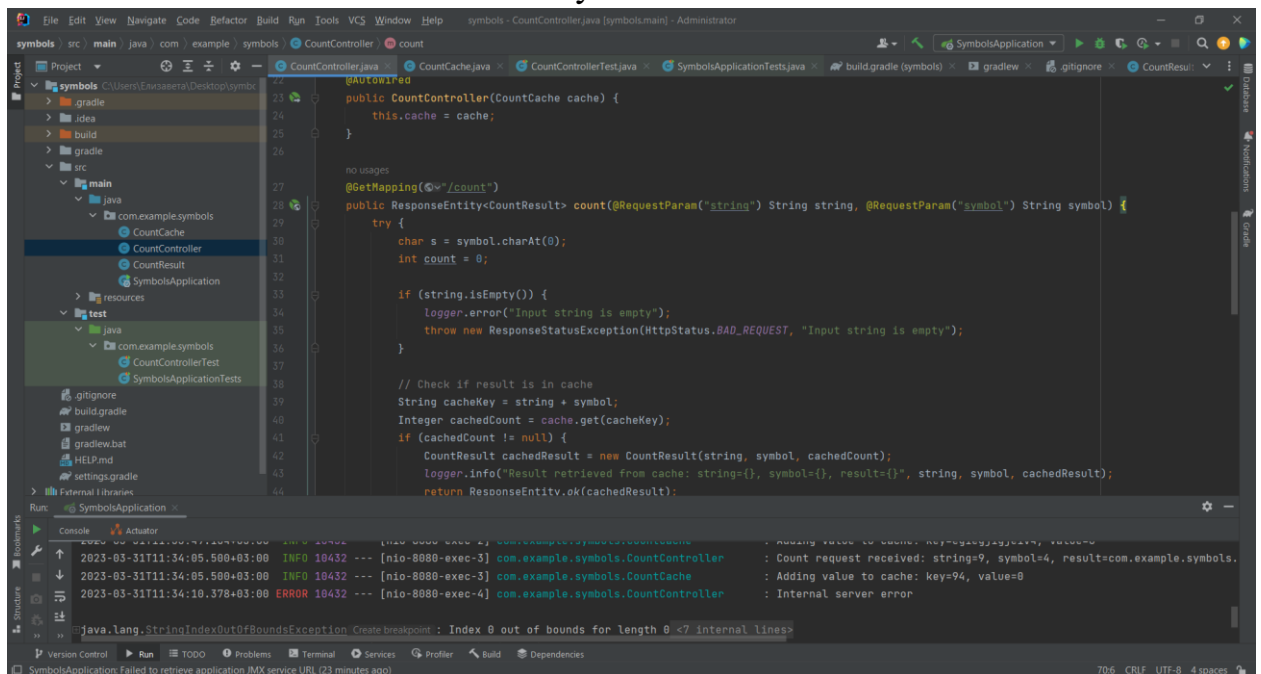


Рисунок 19

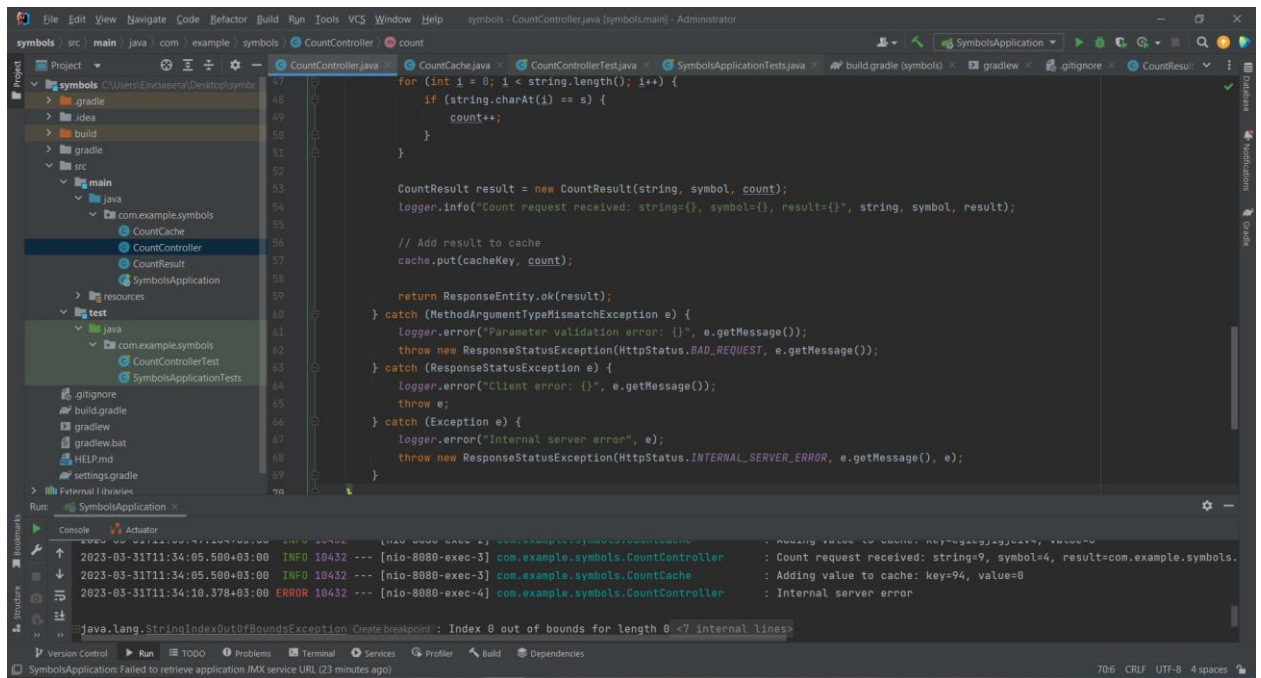


Рисунок 20

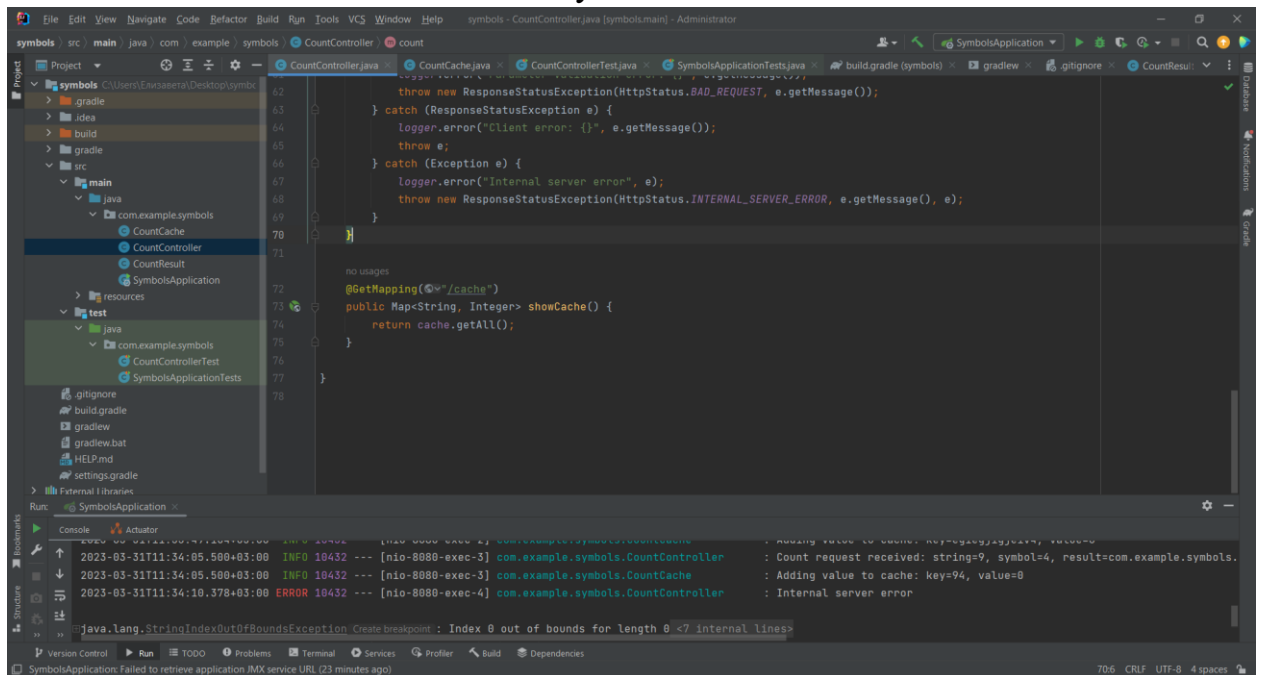


Рисунок 21

2) Класс CountCache (см.рис. 22,23):

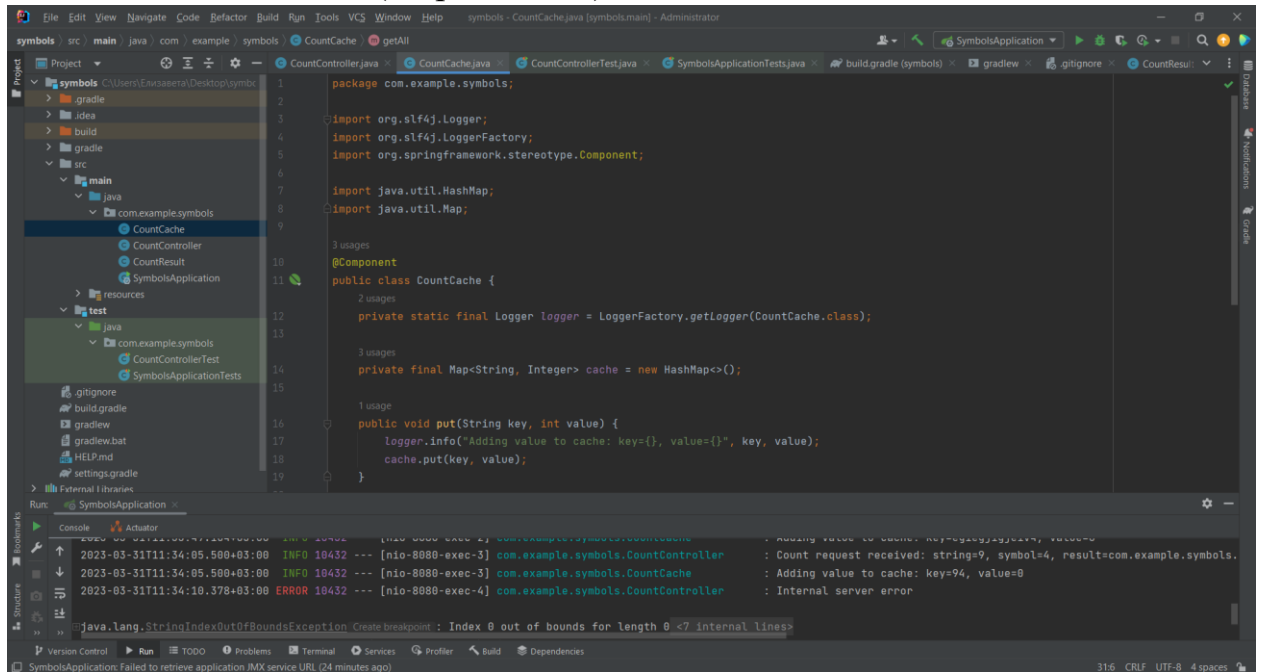


Рисунок 22

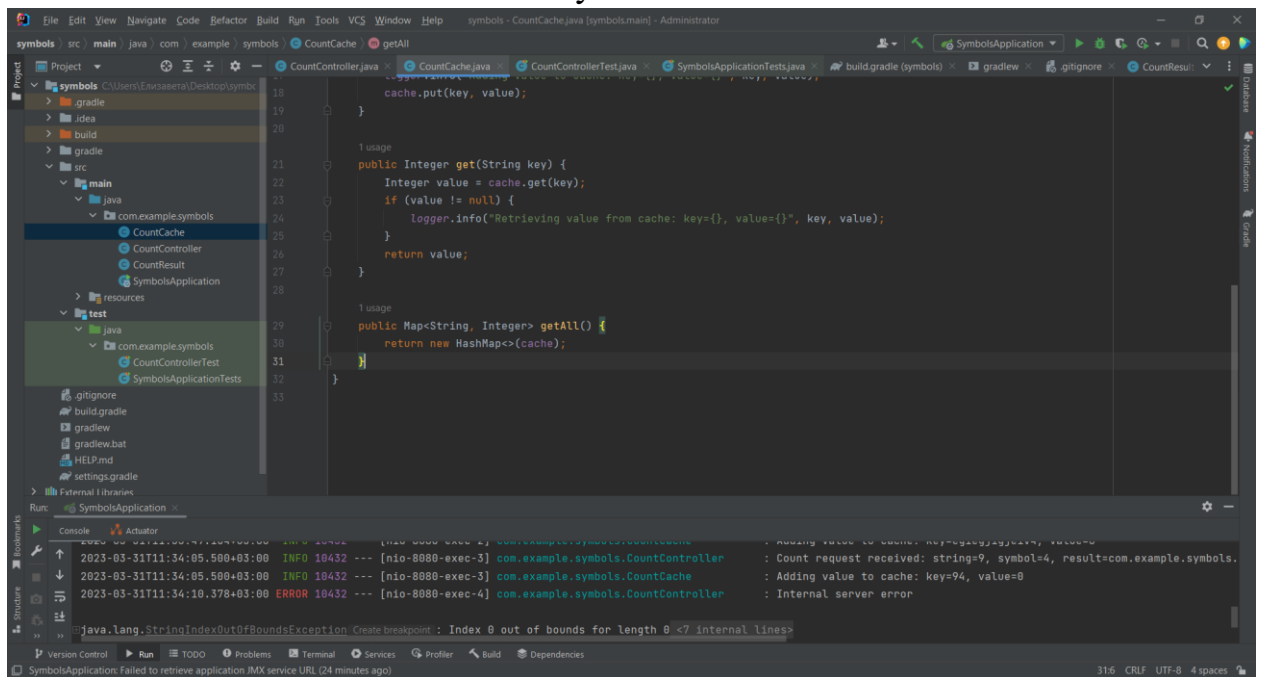


Рисунок 23

Класс CountResult остаётся без изменений.

После того, как произойдёт переход по URL /cache в браузере, Postman получим (см.рис.24,25):

- Если 3:

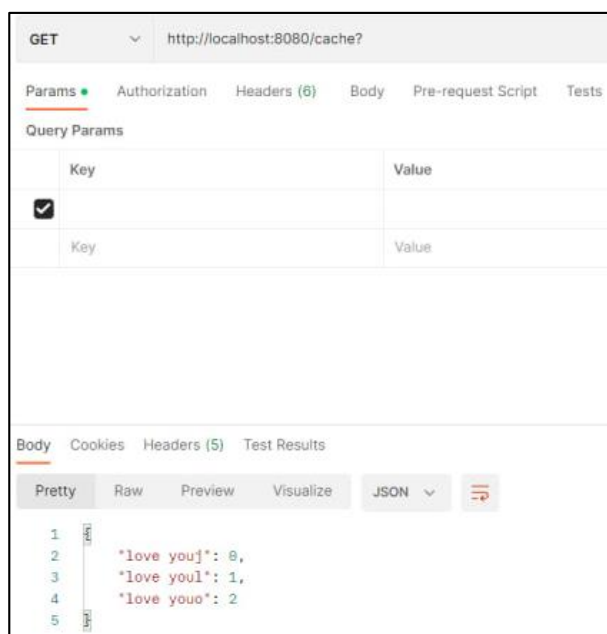


Рисунок 24

- Если 2:



Рисунок 25

Лабораторная работа №4

1) Класс **Counter** (см.рис.26) :

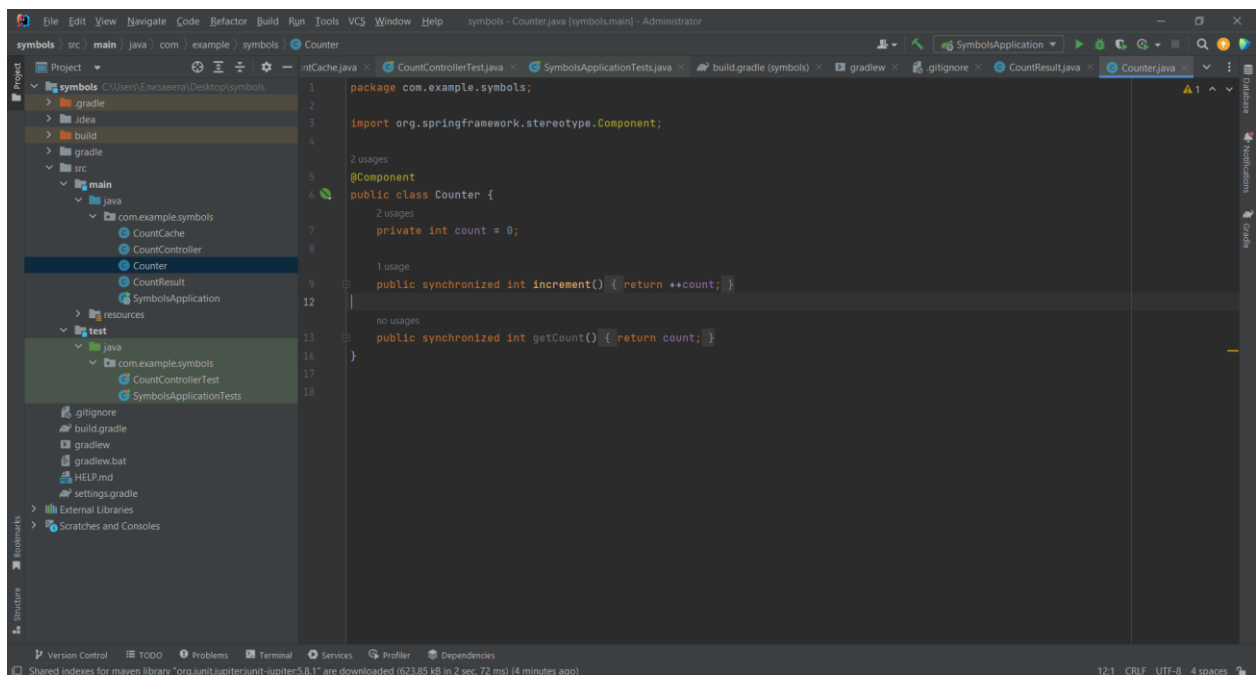


Рисунок 26

2) Класс **CountController** (см. рис. 27,28,29) :

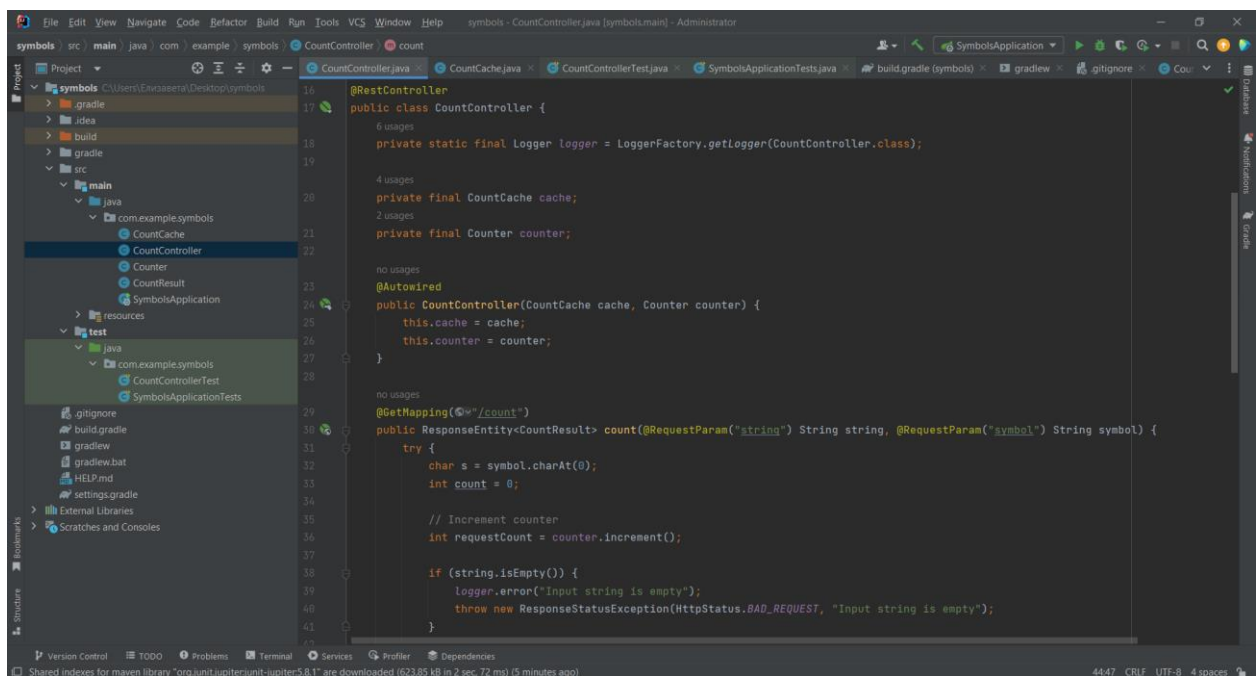


Рисунок 27

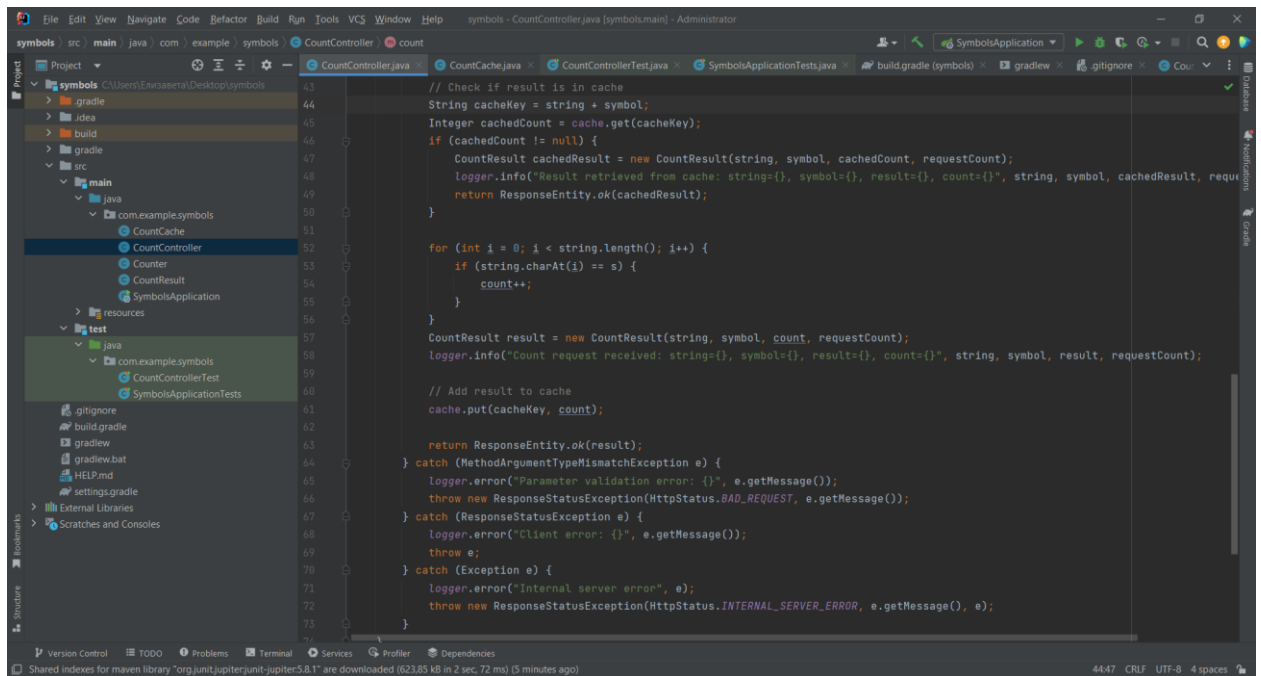


Рисунок 28

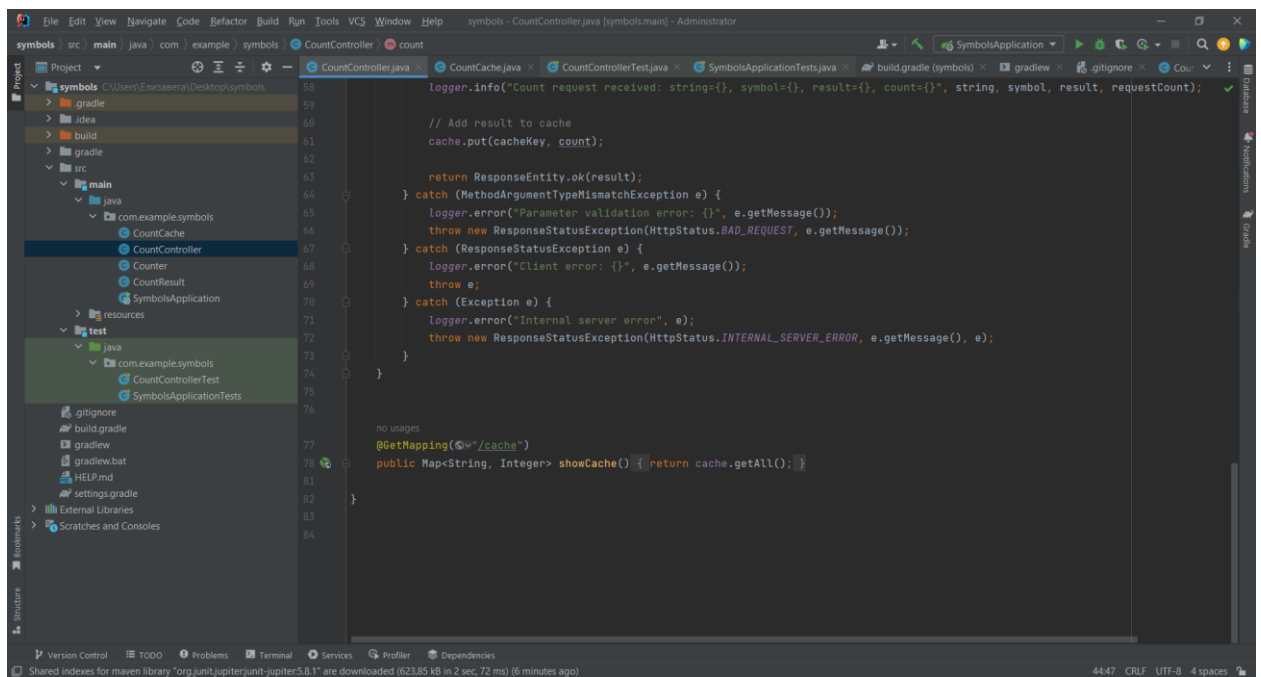


Рисунок 29

Рассмотрим, как работает программа (см.рис.30,31,32):

- Если 3 итерации (рис.30):

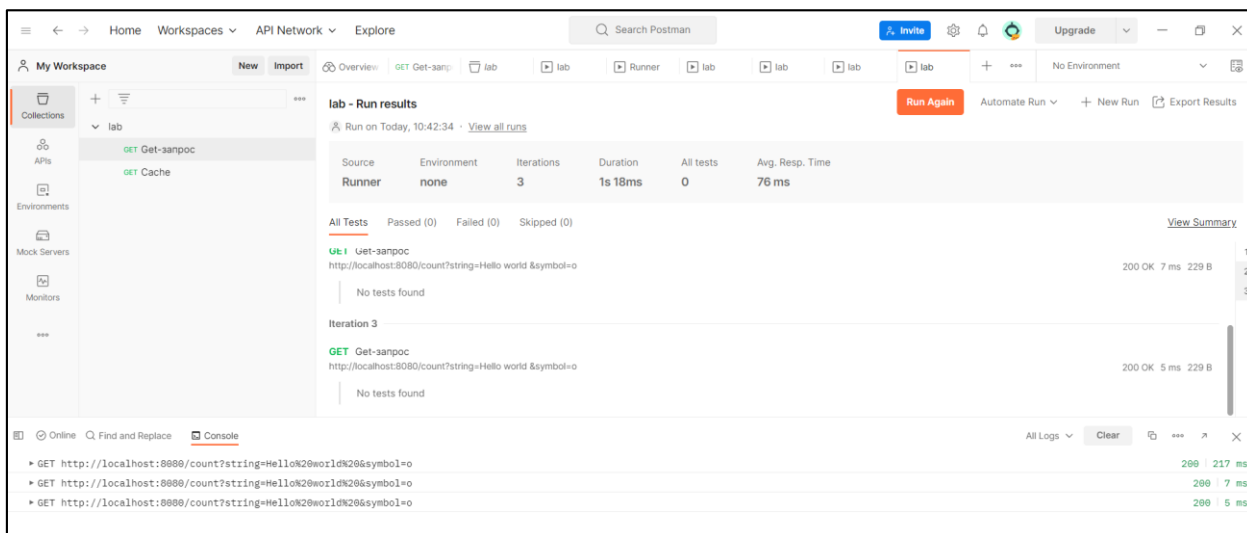


Рисунок 30

- В это время счётчик (рис.31):

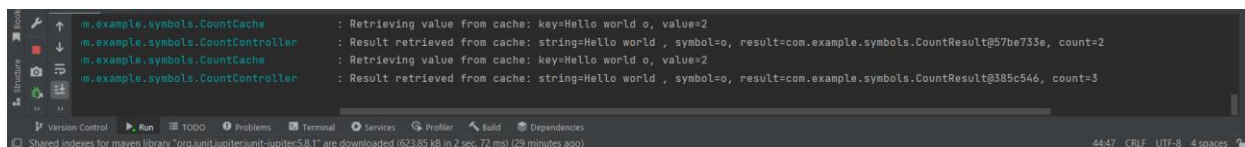


Рисунок 31

Если мы снова проведём нагрузочный тест, добавив ещё 4 итерации, при всё том же запущенном коде, то count будет равен 7.

Пробовала и на большом кол-ве итерации, а именно – 1000+. Всё работало так же, как и при маленьком, и при среднем кол-ве. Счётчик также считал корректно.

Для примера взяла 201 итерацию (рис.32) на 2 GET-запроса всё так же, как и до этого отработало без ошибок.

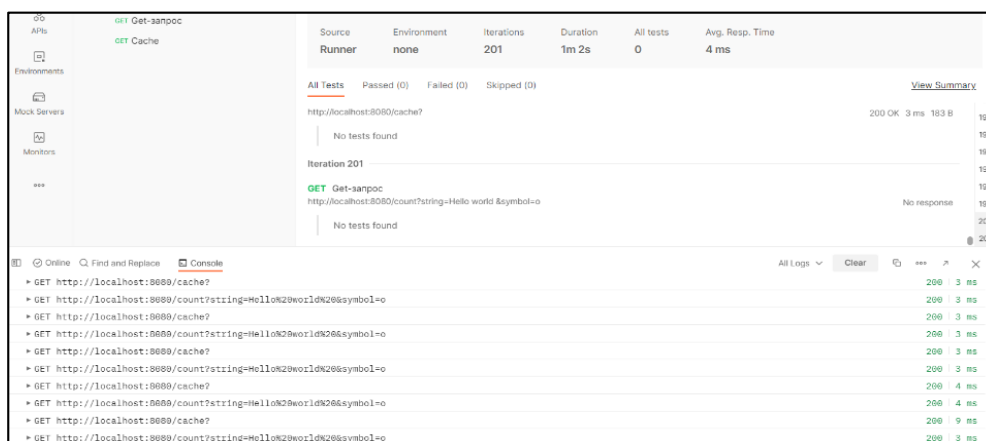


Рисунок 32