

## ЛАБОРАТОРНАЯ РАБОТА №5

### Цель:

Доработать простой REST сервис и добавить unit тесты

### Необходимое программное обеспечение:

IntelliJ IDEA CE, Spring Boot 2, Java 8+, Browser, Postman

### Полезные ссылки:

- 1 <https://start.spring.io/>
- 2 <https://spring.io/quickstart/>
- 3 <https://projectlombok.org/>
- 4 <https://reflectoring.io/bean-validation-with-spring-boot/>
- 5 <https://junit.org/junit5/>

### Задачи:

- 1 Доработка Сервиса 1 - изменение класса Request и добавления метода расчета заработной платы
- 2 Создание unit теста
- 3 Самостоятельно доработать приложение в соответствии с заданием
- 4 Ответить письменно на вопросы в соответствии с номером в списке
- 5 Оформить отчет и прикрепить его на сайте <https://edu.itlearn.ru/> в соответствующем курсе, в соответствующем разделе

### Содержание:

1 Доработка Сервиса 1 - изменение класса Request и добавления метода расчета заработной платы.....	2
2 Создание unit теста.....	4
3 Реализация дополнительного функционала.....	7
4 Вопросы.....	8
5 Требования к оформлению отчета.....	9



## 1 Доработка Сервиса 1 - изменение класса Request и добавления метода расчета заработной платы

Доработайте класс Request, чтобы он соответствовал новой структуре входящего сообщения.

```
{
    "uid": "3",
    "operationUid": "",
    "systemName": "ERP",
    "systemTime": "",
    "source": "",
    "position": "DEV",
    "salary": 100000,
    "bonus": 2.3,
    "workDays": 250,
    "communicationId": 100,
    "templateId": 8,
    "productCode": 1500,
    "smsCode": 10
}
```

Создайте перечисление Positions, в котором добавьте значения DEV (Developer), HR (HR-specialist), TL (Manager), как показано на рисунке 1.

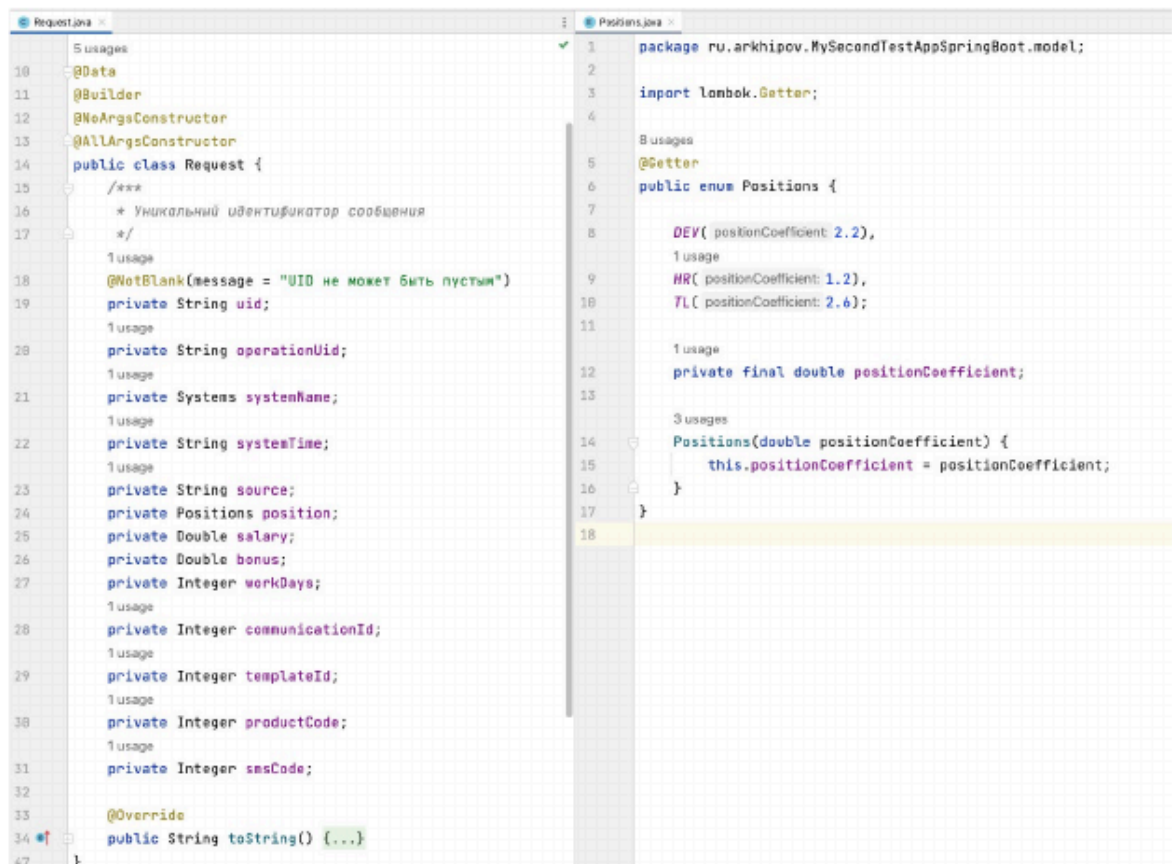


Рисунок 1 – Расширение класса Request



## Доработайте класс Response

```
Response.java x
1 package ru.arkhipov.MySecondTestAppSpringBoot.model;
2
3 import ...
4
5 12 usages
6
7 @Data
8 @Builder
9 public class Response {
10
11     private String uid;
12     private String operationUid;
13     private String systemTime;
14     private Codes code;
15     private Double annualBonus;
16     private ErrorCodes errorCode;
17     private ErrorMessages errorMessage;
18
19 }
20
```

Создайте сервис расчета годовой премии AnnualBonusService и класс его реализующий, как показано на рисунке 2.

```
AnnualBonusService.java x
1 package ru.arkhipov.MySecondTestAppSpringBoot.service;
2
3 import org.springframework.stereotype.Service;
4 import ru.arkhipov.MySecondTestAppSpringBoot.model.Positions;
5
6 4 usages 1 implementation
7 @Service
8 public interface AnnualBonusService {
9
10     2 usages 1 implementation
11     double calculate(Positions positions, double salary, double bonus, int workDays);
12
13 }
14
```

```
AnnualBonusServiceImpl.java x
1 package ru.arkhipov.MySecondTestAppSpringBoot.service;
2
3 import org.springframework.stereotype.Service;
4 import ru.arkhipov.MySecondTestAppSpringBoot.model.Positions;
5
6 1 usage
7 @Service
8 public class AnnualBonusServiceImpl implements AnnualBonusService {
9
10     2 usages
11     @Override
12     public double calculate(Positions positions, double salary, double bonus, int workDays) {
13         return salary * bonus * 365 * positions.getPositionCoefficient() / workDays ;
14     }
15 }
16
```



Рисунок 2 – Сервис расчета годовой премия





## 2 Создание unit теста

Поставьте курсор на название класса и вызовите контекстное меню, как показано на рисунке 3 и нажмите на «Show context actions»

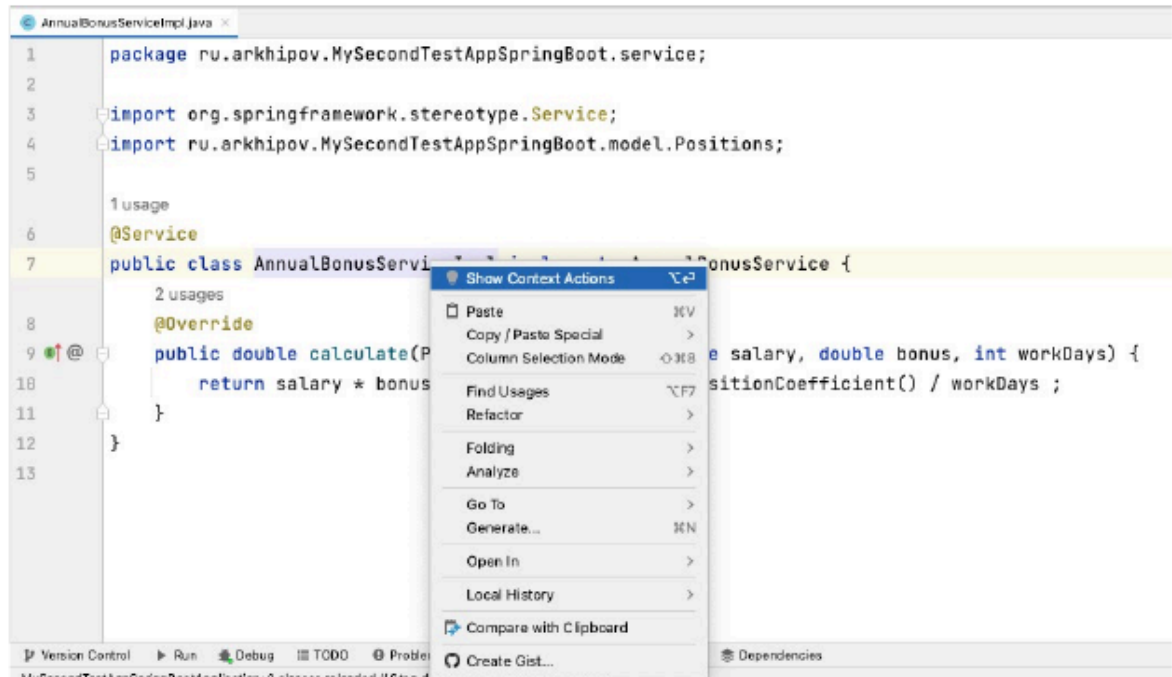


Рисунок 3 – Вызов контекстного меню

В открывшемся дополнительном контекстном меню, как показано на рисунке 4 выберите «Create Test»

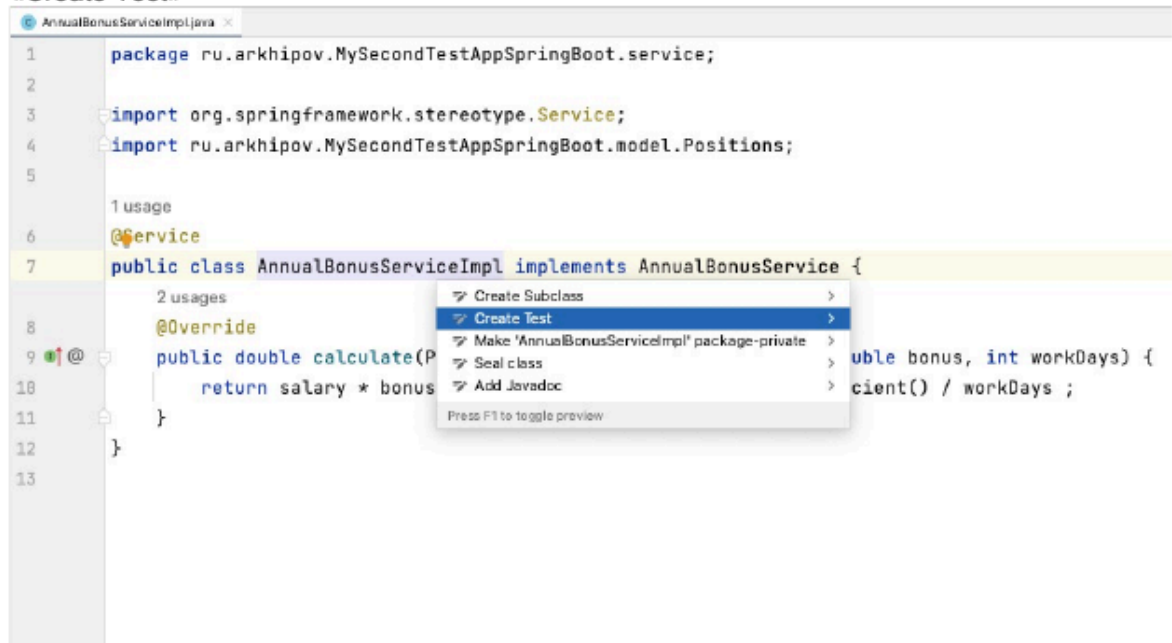


Рисунок 4 – Дополнительное контекстное меню



В открывшемся окне мастера создания теста выберите в поле Testing library JUnit5 и в списке методов поставьте галочку около метода calculate(), нажмите «OK», как показано на рисунке 5

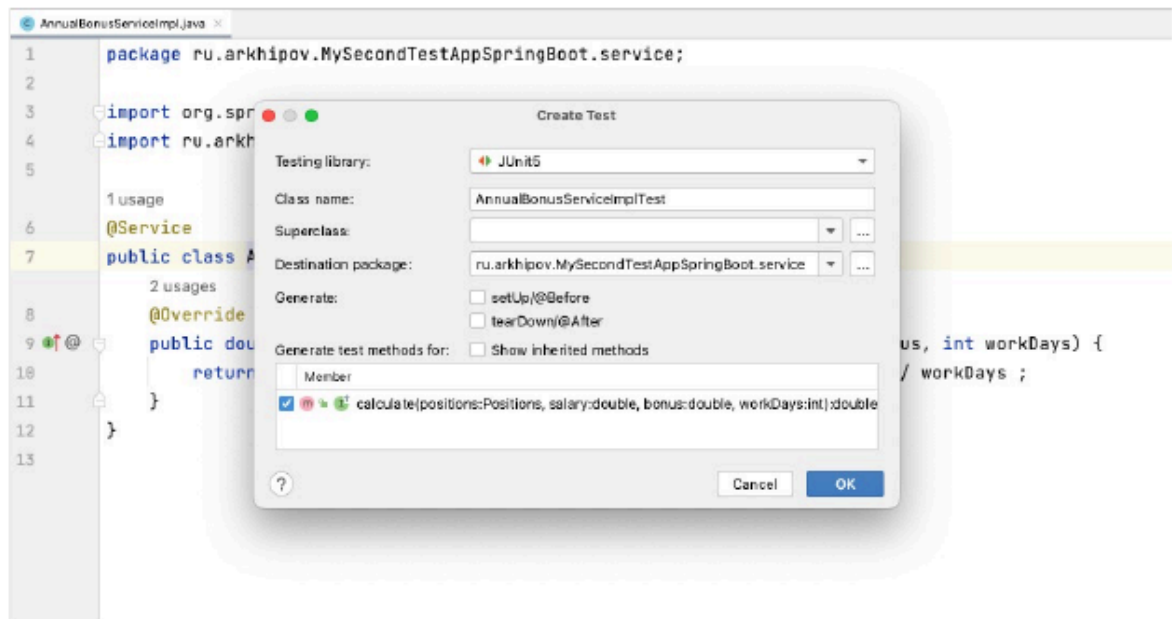


Рисунок 5 – Мастер создания unit теста

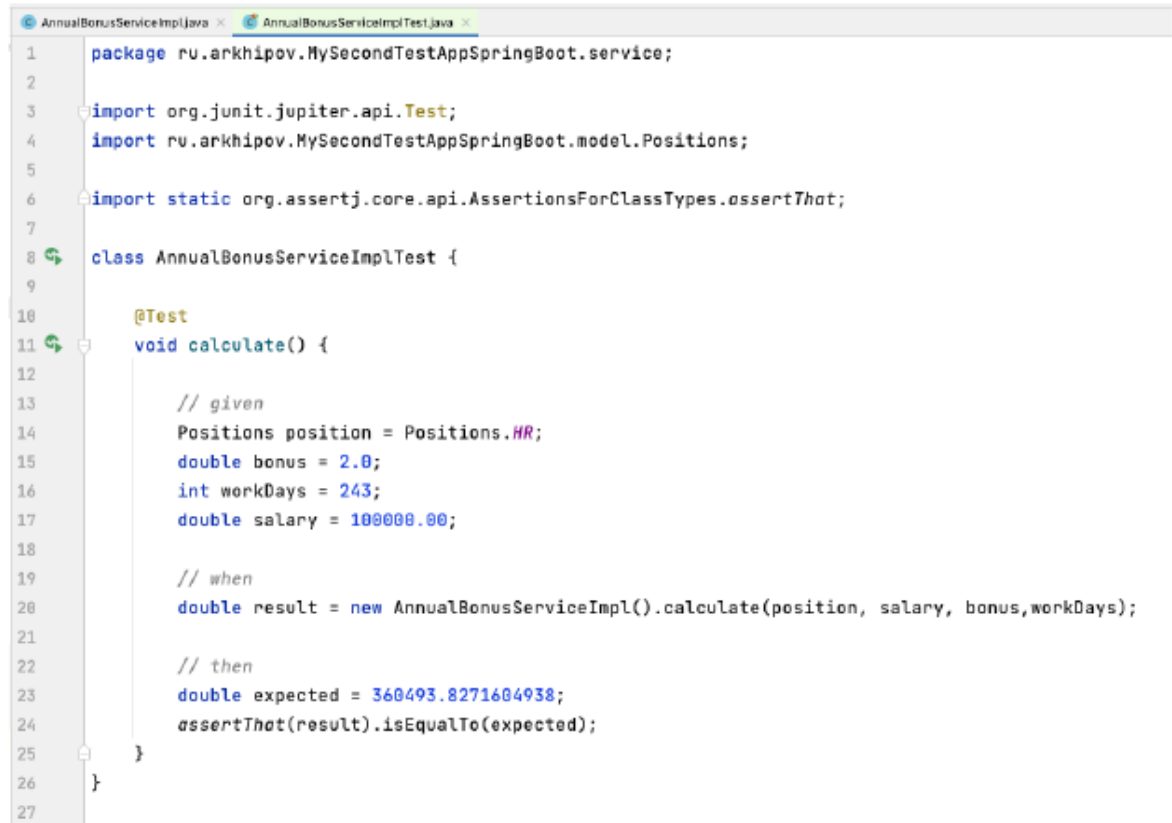
В результате будет сгенерирован тест в папке test проекта, как показано на рисунке 6



Рисунок 6 -Автоматически сгенерированный класс с тестом



Напишите простой тест метода calculate, как показано на рисунке 7



```
1 package ru.arkhipov.MySecondTestAppSpringBoot.service;
2
3 import org.junit.jupiter.api.Test;
4 import ru.arkhipov.MySecondTestAppSpringBoot.model.Positions;
5
6 import static org.assertj.core.api.AssertionsForClassTypes.assertThat;
7
8 class AnnualBonusServiceImplTest {
9
10     @Test
11     void calculate() {
12
13         // given
14         Positions position = Positions.HR;
15         double bonus = 2.0;
16         int workDays = 243;
17         double salary = 100000.00;
18
19         // when
20         double result = new AnnualBonusServiceImpl().calculate(position, salary, bonus, workDays);
21
22         // then
23         double expected = 360493.8271604938;
24         assertThat(result).isEqualTo(expected);
25     }
26 }
27
```

Рисунок 7 – Реализация простого unit теста для метода calculate



### 3 Реализация дополнительного функционала

Необходимо самостоятельно доработать приложение:

1. Доработайте метод `calculate`, чтобы в нем вычислялось количество дней в году, в зависимости от того високосный год или нет.
2. Расширьте `enum Positions`, добавьте три дополнительных позиции, а также добавьте поле `isManager` с типом данных `boolean`, для менеджеров установите значение `true`, в прочих случаях `false`.
3. Напишите метод, вычисляющий квартальную премию, метод работает только для менеджеров и иных управленцев (те, у кого `isManager == true`), добавьте проверку в метод, на признак управленец это или нет.
4. Разработайте `unit` тест для метода расчета квартальной премии.
5. Произведите рефакторинг приложения – добавьте комментарии к каждому полю в классах `Request` и `Response` в соответствии с описанием из лабораторной работы №2, пример комментария к полю показан на рисунке 1 текущей работы.
6. Сделайте так, чтобы в классе `MyController` стало меньше кода, т.е. приберитесь в нем.





#### 4 Вопросы

Письменно ответьте на вопросы, если ваш номер в списке группы четный, то нужно ответить на четные вопросы, если нечетный, то нужно ответить на нечетные вопросы.

1. Что такое unit тестирование?
2. Что такое smoke тестирование?
3. Что такое интеграционное тестирование?
4. Какие библиотеки используются в Spring Boot для unit тестирования?
5. Что такое Assert? Какие бывают Assert?
6. Какие аннотации наиболее распространены при использовании библиотеки JUnit 5?



## 5 Требования к оформлению отчета

1. Титульный лист
2. Цель работы
3. Описание задачи
4. Ход выполнения (содержит код программы или скриншоты кода программы, ответы на вопросы)
5. Ссылку на репозиторий git на сайте [github.com](https://github.com)
6. Вывод

Оформление:

- а) шрифт Times New Roman
- б) размер шрифта 12 или 14
- в) межстрочный интервал 1,5

Отчет выполняется индивидуально.

