**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

отчет

**по лабораторной работе №7**

**по дисциплине «Объектно-ориентированное программирование»**

Тема: Модульное тестирование приложения

|  |  |  |
| --- | --- | --- |
| Студент гр. 9308 |  | Дементьев Д.П. |
| Преподаватель |  | Гречухин М.Н. |

Санкт-Петербург

2021

# Цель работы

Знакомство с технологией модульного тестирования Java приложений с использованием системы JUnit.

# Методы тестируемые в приложении

Так как логика изначально проектировалась через использование методов класса Bus, то всё тестирование было проведено в классе BusTest, подробнее:

1. **Выборочно для сеттеров (и геттеров одновременно)**

setRegistr() – установка регистрационного номера

setCapacity() – установка вместительности автобуса

1. **Проверка правильности выстроенных связей между сущностями**

hireToDriver() – наём работника для автобуса

chooseRoute() – выбор рабочего маршрута для автобуса

setViolation() – фиксация нарушения работы на маршруте

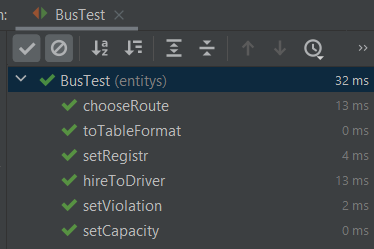
1. **Корректность формирования запроса для отображения**

toTableFormat() – формирование строки данных для добавления в таблицу

Были выбраны методы, включающие в себя наибольшее количество действий программы, чтобы проверить максимальное количество возможных случаев ошибок при минимальном количестве тестов.

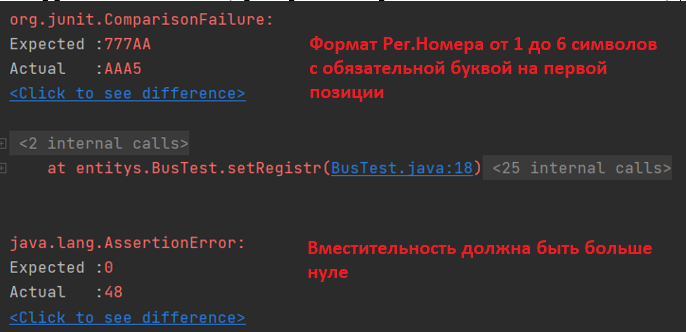
# Демонстрация выполнения тестов

Сперва приведём результат всех успешно пройденных тестов:



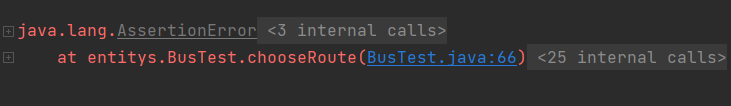
*Рисунок 1. Успешное прохождение тестов*

Далее попробуем «завалить» несколько тестов. Сеттеры проверим вводом некорректных данных (проверка геттеров включена в эти же тесты):



*Рисунок 2. Некорректные данные для сеттеров*

Попробуем выбрать маршрут для автобуса без водителя:

**

*Рисунок 3. Недопустимая операция построения связей между объектами*

# Исходный текст класса тестов

package entitys;  
  
import org.junit.Test;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import static org.junit.Assert.\*;  
  
public class BusTest {  
  
 @Test  
 public void setRegistr() {  
 // формат номера - от 1 до 6 символов, первый символ обязательно буква  
 Bus model = new Bus();  
 String new\_reg = "777AA";  
 model.setRegistr(new\_reg);  
 *assertEquals*(new\_reg, model.getRegistr());  
 }  
  
 @Test  
 public void setCapacity() {  
 // допускается 0 < capacity  
 Bus model = new Bus();  
 int new\_capacity = 0;  
 *assertFalse*(model.setCapacity(new\_capacity));  
 *assertEquals*(new\_capacity, model.getCapacity()); // значение по-умолчанию  
 }  
  
 @Test  
 public void hireToDriver() {  
 // при наёме работника прошлый работник(если он был) должен быть уволен  
 Bus model = new Bus();  
 Driver person\_one = new Driver(), person\_two = new Driver();  
 model.hireToDriver(person\_one);  
 model.hireToDriver(person\_two);  
 *assertEquals*(model.getDriver().getName(), person\_two.getName());  
 *assertNull*(person\_one.getBus());  
 }  
  
 @Test  
 public void toTableFormat() {  
 // проверка на корректность преобразования к формату для занесения в таблицу DefaultTableModel  
 Bus model = new Bus("A123BC", 96);  
 Driver person = new Driver();  
 model.hireToDriver(person);  
 String[] expected = {Integer.*toString*(model.getId()),"A123BC", "96", person.getName(), "Отсутствует", "Ок"},  
 received = model.toTableFormat();  
 for (int i = 0; i < expected.length; ++i) {  
 *assertEquals*(expected[i], received[i]);  
 }  
 }  
  
 @Test  
 public void chooseRoute() {  
 // все автобусы выставленные на маршрут должны оказаться в списке объекта Маршрут и совпасть ссылки на них  
 // также нельзя выставить автобус на маршрут без водителя  
 Bus model1 = new Bus(),  
 model2 = new Bus(),  
 model3 = new Bus();  
 Driver person1 = new Driver(),  
 person2 = new Driver(),  
 person3 = new Driver();  
 Route route = new Route();  
 // проверка попытки выставить автобус на маршрут без водителя  
 *assertFalse*(model1.chooseRoute(route));  
 *assertNull*(model1.getRoute());  
 // совпадение ссылок  
 model1.hireToDriver(person1); model2.hireToDriver(person2); model3.hireToDriver(person3);  
 model1.chooseRoute(route); model2.chooseRoute(route); model3.chooseRoute(route);  
 List<Bus> list = new ArrayList<>();  
 list.add(model1); list.add(model2); list.add(model3);  
 for (int i = 0; i < list.size(); ++i) {  
 *assertSame*(list.get(i), route.getBuses().get(i));  
 }  
 }  
  
 @Test  
 public void setViolation() {  
 Bus model = new Bus();  
 Driver person = new Driver();  
 Route route = new Route();  
  
 model.hireToDriver(person);  
 model.chooseRoute(route);  
 *assertEquals*(model.getRoute(), route); // успешно выставлен на маршрут  
  
 Violation repair = new Violation("Ремонт");  
 route.setViolation(repair);  
 *assertNull*(model.getRoute()); // из-за ремонта на маршруте - все автобусы снимаются с него  
 }  
}

# Выводы

При выполнении лабораторной работы была изучена технология модульного тестирования, с помощью системы JUnit-тестов. Ошибок выявлено не было.