Домашнее задание по теме «Современные IDE»

Формулировка задания:

1. Преобразовать проект для ДЗ 9 (работа с машинами) в maven-проект.

- Создаем pom.xml файл в проекте с домашним заданием.
- Пишем/изменяем теги заголовка groupId, artifactId.
- Добавить зависимости Junit 5.8.1, Mockito, Lombok
- Добавляем тег <build>. Указываем в нем где находятся ресурсы, имя результирующего jar файла.

2. Добавить перед классами аннотации Lombok. Удалить лишние геттеры, сеттеры, конструкторы. Проверить, что проект не изменил работоспособность.

3. Добавить в проект CarRepository по аналогу с Промежуточной аттестацией 1. Реализовать CarRepositoryFileImpl для получения данных из файла для заданной структуры.

4. Переписать App приложения для работы с CarRepository. Входные данные добавить в файл Resources.

5. Протестировать класс CarRepository юнит-тестами. Файл для тестов добавить в Test Resources.

6. Настроить Maven для запуска unit-тестов.

7. Убедиться, что все библиотеки упакованы в файл JAR.

Программа реализуется в отдельной ветке git homeworks/homework17 с описанием хода работы по задаче.

В корне папки с программой должен быть файл .gitignore.

Программа локально коммитится и публикуется в репозиторий GitHub на проверку.

Планируемый результат:

Работа в IDE Intellij Idea с Maven проектом и JUnit-тестами.

Описания плана работы:

1. Ссылка на программу в репозитории github;
2. Отчёт со скринами выполнения задач - постановка задачи, код задачи и результат в консоли Intellij Idea. Подготовка необходимых class-файлов и последующая упаковка в архив JAR с помощью фаз Maven.

Перечень инструментов, необходимых для реализации деятельности:

Персональный компьютер, JDK 17 (либо OpenJDK 17), Intellij Idea для разработки на Java, GIT, Tortoise GIT, jUnit 5.8.1, Lombok, Apache Maven 3.9.5

Пояснение к домашнему заданию

1. Структура задачи (основная часть ДЗ 9)

Car

Базовый автомобиль обладает следующими свойствами: маркой (строка), моделью (строка), годом выпуска (int), мощностью в лошадиных силах (int), ускорением (int), подвеской (int) и долговечностью (int).

Каждый отдельный тип автомобиля дополняет эти свойства. Вот типы:
1. PerformanceCar – гоночный автомобиль.
Имеет дополнения addOns (массив строк, по умолчанию – пустой)
Увеличенная мощность двигателя на 50%.
Уменьшенная подвеска на 25%.
2. ShowCar – спортивная машина. Looking cool there, bro.
Включает поле stars (int). (по умолчанию – 0), поле для оценки популярности автомобиля.

Race

Гонка имеет следующие свойства: длина (int), маршрут (строка), призовой фонд (int) и участники (коллекция автомобилей),
• CasualRace – обычная гонка.
• DragRace – гонка за самый мощный двигатель. Идеальное переключение передач — залог победы.

• DriftRace – дрифтовая гонка.

Garage

• Garage – место, где остаются все автомобили, когда они не участвуют в гонках. Гараж также предоставляет возможность модифицировать припаркованный автомобиль. Включает parkedCars (массив объектов типа Car).

Каждый из представленных классов должен включать:

1. Конструктор пустой и с параметрами;

2. Переопределенный метод toString();

3. Геттеры и сеттеры для полей. Обратить внимание, что поля требуется сделать private;

4. У классов переопределены методы equals() и hashcode().

2. Работа с Lombok

Lombok - это плагин компилятора, который добавляет в Java новые «ключевые слова» и превращает аннотации в Java-код

Шпаргалка по аннотациям Lombok:

* @AllArgsConstructor - заменяет конструктор, в котором инициализируются все поля.

* @NoArgsConstructor - создает дефолтный/пустой конструктор.

* @Builder - позволяет создавать объект с помощью статического Билдера (строителя). Можно инициализировать либо все поля, либо определенные, либо вообще никакие.

* @Getter - у всех полей класса создается геттер.

* @Setter - у всех полей класса создается сеттер.

* @Data - содержит в себе такие аннотации как - @ToString, @EqualsAndHashCode, @Getter, @Setter и @RequiredArgsConstructor.