Домашнее задание

по теме «Инкапсуляция. Модификаторы доступа в Java».

Формулировка задания:

Основная структура программы должна включать следующие элементы:

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().

Работу с классами проверить в методе main класса App.

Программа реализуется в отдельной ветке git homeworks/homework09. При сохранении состояния программы (коммиты) пишется сообщение 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.

```java
package homeworks.homework09;

import java.util.Objects;

public class Car {  19 usages  2 inheritors  new *

    private String brand;  7 usages
    private String model;  7 usages
    private int yearOfRelease;  7 usages
    private int power;  7 usages
    private int acceleration;  7 usages
    private int suspension;  7 usages
    private int durability;  7 usages

    public Car() {  4 usages  new *
    }

    public Car(String brand, String model, int yearOfRelease, int power,  2 usages  new *
               int acceleration, int suspension, int durability) {

        this.brand = brand;
        this.model = model;
        this.yearOfRelease = yearOfRelease;
        this.power = power;
        this.acceleration = acceleration;
        this.suspension = suspension;
        this.durability = durability;
    }
```

```java

    public String getBrand() {  4 usages  new *
        return brand;
    }

    public String getModel() {  2 usages  new *
        return model;
    }

    public int getYearOfRelease() {  2 usages  new *
        return yearOfRelease;
    }

    public int getPower() {  5 usages  new *
        return power;
    }

    public int getAcceleration() {  2 usages  new *
        return acceleration;
    }

    public int getSuspension() {  2 usages  new *
        return suspension;
    }

```

```java
    public int getDurability() {  2 usages  new *
        return durability;
    }

    public void setBrand(String brand) {  no usages  new *
        this.brand = brand;
    }

    public void setModel(String model) {  no usages  new *
        this.model = model;
    }

    public void setYearOfRelease(int yearOfRelease) {  no usages  new *
        this.yearOfRelease = yearOfRelease;
    }

    public void setPower(int power) {  2 usages  1 override  new *
        this.power = power;
    }

    public void setAcceleration(int acceleration) {  no usages  new *
        this.acceleration = acceleration;
    }

    public void setSuspension(int suspension) {  1 usage  1 override  new *
        this.suspension = suspension;
    }
```

```java

    public void setDurability(int durability) {  no usages  new *
        this.durability = durability;
    }

    @Override  2 overrides  new *
    public boolean equals(Object o) {
        if (o == null || getClass() != o.getClass()) return false;
        Car car = (Car) o;
        return yearOfRelease == car.yearOfRelease && power == car.power &&
                acceleration == car.acceleration && suspension == car.suspension &&
                durability == car.durability && Objects.equals(brand, car.brand) &&
                Objects.equals(model, car.model);
    }

    @Override  2 overrides  new *
    public int hashCode() {
        return Objects.hash(brand, model, yearOfRelease, power, acceleration,
                suspension, durability);
    }

    @Override  2 overrides  new *
    public String toString() {
        return "Car{" +
                "brand='" + brand + '\'' +
                ", model='" + model + '\'' +
                ", yearOfRelease=" + yearOfRelease +
                ", power=" + power +
```

```
109                 ", acceleration=" + acceleration +
110                 ", suspension=" + suspension +
111                 ", durability=" + durability +
112                 '}';
113         }
114     }
```

```java
package homeworks.homework09;

import java.util.Arrays;
import java.util.Objects;

public class PerformanceCar extends Car {  6 usages  new *

    private String[] addons = {};  8 usages

    public PerformanceCar() {  no usages  new *
    }

    public PerformanceCar(String[] addons) {  no usages  new *
        this.addons = addons;
    }

    public PerformanceCar(String brand, String model, int yearOfRelease, int power,  2 usages  new *
                          int acceleration, int suspension, int durability, String[] addons) {

        super(brand, model, yearOfRelease, (int) (power * 1.5), acceleration,
                (int) (suspension * 0.75), durability);

        this.addons = addons;
    }

    public String[] getAddons() {  no usages  new *
        return addons;
    }

```

```java
    public void setAddons(String[] addons) {  no usages  new *
        this.addons = addons;
    }

    @Override  2 usages  new *
    public void setPower(int power) {
        super.setPower((int) (power * 1.5));
    }

    @Override  1 usage  new *
    public void setSuspension(int suspension) {
        super.setSuspension((int) (suspension * 0.75));
    }

    @Override  new *
    public boolean equals(Object o) {
        if (o == null || getClass() != o.getClass()) return false;
        if (!super.equals(o)) return false;
        PerformanceCar that = (PerformanceCar) o;
        return Objects.deepEquals(addons, that.addons);
    }

    @Override  new *
    public int hashCode() {
        return Objects.hash(super.hashCode(), Arrays.hashCode(addons));
    }

```

```
57      @Override  new *
58      public String toString() {
59          return "PerformanceCar{" +
60                  "brand='" + super.getBrand() + '\'' +
61                  ", model='" + super.getModel() + '\'' +
62                  ", yearOfRelease=" + super.getYearOfRelease() +
63                  ", power=" + super.getPower() +
64                  ", acceleration=" + super.getAcceleration() +
65                  ", suspension=" + super.getSuspension() +
66                  ", durability=" + super.getDurability() +
67                  ", addons=" + Arrays.toString(addons) +
68                  '}';
69      }
70  }
```

```java
package homeworks.homework09;

import java.util.Objects;

public class ShowCar extends Car {  5 usages  new *

    private int stars = 0;  8 usages
    private int popularity;  8 usages

    public ShowCar() {  no usages  new *
    }

    public ShowCar(int stars, int popularity) {  no usages  new *
        this.stars = stars;
        this.popularity = popularity;
    }

    public ShowCar(String brand, String model, int yearOfRelease, int power,  1 usage  new *
                   int acceleration, int suspension, int durability, int stars,
                   int popularity) {

        super(brand, model, yearOfRelease, power, acceleration, suspension, durability);
        this.stars = stars;
        this.popularity = popularity;
    }

    public int getStars() {  no usages  new *
        return stars;
    }
```

```java

    public void setStars(int stars) {  no usages  new *
        this.stars = stars;
    }

    public int getPopularity() {  no usages  new *
        return popularity;
    }

    public void setPopularity(int popularity) {  no usages  new *
        this.popularity = popularity;
    }

    @Override  new *
    public boolean equals(Object o) {
        if (o == null || getClass() != o.getClass()) return false;
        if (!super.equals(o)) return false;
        ShowCar showCar = (ShowCar) o;
        return stars == showCar.stars && popularity == showCar.popularity;
    }

    @Override  new *
    public int hashCode() {
        return Objects.hash(super.hashCode(), stars, popularity);
    }

    @Override  new *
    public String toString() {
```

```
58          return "ShowCar{" +
59                  "brand='" + super.getBrand() + '\'' +
60                  ", model='" + super.getModel() + '\'' +
61                  ", yearOfRelease=" + super.getYearOfRelease() +
62                  ", power=" + super.getPower() +
63                  ", acceleration=" + super.getAcceleration() +
64                  ", suspension=" + super.getSuspension() +
65                  ", durability=" + super.getDurability() +
66                  ", stars=" + stars +
67                  ", popularity=" + popularity +
68                  '}';
69      }
70  }
```

```java
package homeworks.homework09;

import java.util.Arrays;
import java.util.Objects;

public class Race {  6 usages  3 inheritors  new *

    private int distance;  7 usages
    private String route;  7 usages
    private int prizeFund;  7 usages
    private Car[] raceParticipants;  7 usages

    public Race() {  3 usages  new *
    }

    public Race(int distance, String route, int prizeFund, Car[] raceParticipants) {
        this.distance = distance;
        this.route = route;
        this.prizeFund = prizeFund;
        this.raceParticipants = raceParticipants;
    }

    public String startRace (Car[] raceCar) {  2 usages  new *
        int winNum = 0;
        int maxPower = 0;
        for (int i = 0; i < raceCar.length; i++) {
            if (maxPower < raceCar[i].getPower()) {
                maxPower = raceCar[i].getPower();
                winNum = i;
```

```
30              }
31          }
32          return "Победил гонщик на " + raceCar[winNum].getBrand();
33      }
34
35      public int getDistance() {  no usages  new *
36          return distance;
37      }
38
39      public void setDistance(int distance) {  no usages  new *
40          this.distance = distance;
41      }
42
43      public String getRoute() {  no usages  new *
44          return route;
45      }
46
47      public void setRoute(String route) {  no usages  new *
48          this.route = route;
49      }
50
51      public int getPrizeFund() {  no usages  new *
52          return prizeFund;
53      }
54
```

```java
    public void setPrizeFund(int prizeFund) {  no usages  new *
        this.prizeFund = prizeFund;
    }

    public Car[] getRaceParticipants() {  no usages  new *
        return raceParticipants;
    }

    public void setRaceParticipants(Car[] raceParticipants) {  no usages  new *
        this.raceParticipants = raceParticipants;
    }

    @Override  new *
    public boolean equals(Object o) {
        if (o == null || getClass() != o.getClass()) return false;
        Race race = (Race) o;
        return distance == race.distance && prizeFund == race.prizeFund &&
                Objects.equals(route, race.route) && Objects.deepEquals(raceParticipants,
                race.raceParticipants);
    }

    @Override  new *
    public int hashCode() {
        return Objects.hash(distance, route, prizeFund, Arrays.hashCode(raceParticipants));
    }

```

```java
    @Override  3 overrides  new *
    public String toString() {
        return "Race{" +
                "distance=" + distance +
                ", route='" + route + '\'' +
                ", prizeFund=" + prizeFund +
                ", raceParticipants=" + Arrays.toString(raceParticipants) +
                '}';
    }
}
```

```java
package homeworks.homework09;

public class CasualRace extends Race {  1 usage  new *

    public CasualRace() {  no usages  new *
    }

    public CasualRace(int distance, String route, int prizeFund, Car[] raceParticipants) {
        super(distance, route, prizeFund, raceParticipants);
    }

    @Override  new *
    public String toString() {
        return "CasualRace{}";
    }
}
```

```java
package homeworks.homework09;

public class DragRace extends Race {  no usages  new *

    public DragRace() {  no usages  new *
    }

    public DragRace(int distance, String route, int prizeFund, Car[] raceParticipants) {
        super(distance, route, prizeFund, raceParticipants);
    }

    @Override  new *
    public String toString() {
        return "DragRace{}";
    }
}
```

```java
package homeworks.homework09;

public class DriftRace extends Race {  no usages  new *

    public DriftRace() {  no usages  new *
    }

    public DriftRace(int distance, String route, int prizeFund, Car[] raceParticipants) {
        super(distance, route, prizeFund, raceParticipants);
    }

    @Override  new *
    public String toString() {
        return "DriftRace{}";
    }
}
```

```java
package homeworks.homework09;

import java.util.Arrays;
import java.util.Objects;
import java.util.Scanner;

public class Garage {  4 usages  new *

    private Car[] parkedCars;  12 usages

    public Garage() {  1 usage  new *
    }

    public Garage(Car[] parkedCars) {  no usages  new *
        this.parkedCars = parkedCars;
    }

    public void modifyCar(int numberCar) {  1 usage  new *
        Car car;
        numberCar--;

        if (parkedCars[numberCar] instanceof PerformanceCar) {
            car = (PerformanceCar) parkedCars[numberCar];
        } else if (parkedCars[numberCar] instanceof ShowCar) {
            car = (ShowCar) parkedCars[numberCar];
        } else {
            car = parkedCars[numberCar];
        }

```

```
30          System.out.print("Какая мощность будет теперь у " + car.getBrand()
31          + "? ");
32
33          Scanner scanner = new Scanner(System.in);
34          car.setPower(scanner.nextInt());
35      }
36
37      public Car[] getParkedCars() {  1 usage  new *
38          return parkedCars;
39      }
40
41      public void setParkedCars(Car[] parkedCars) {  1 usage  new *
42          this.parkedCars = parkedCars;
43      }
44
45      @Override  new *
46      public boolean equals(Object o) {
47          if (o == null || getClass() != o.getClass()) return false;
48          Garage garage = (Garage) o;
49          return Objects.deepEquals(parkedCars, garage.parkedCars);
50      }
51
52      @Override  new *
53      public int hashCode() {
54          return Arrays.hashCode(parkedCars);
55      }
```

```
56
57      @Override  new *
58      public String toString() {
59          return "Garage{" +
60                  "parkedCars=" + Arrays.toString(parkedCars) +
61                  '}';
62      }
63  }
```

```java
package homeworks.homework09;

import java.util.Scanner;

public class App {
    public static void main(String[] args) {

        String winner;
        Garage garage = new Garage();
        Car[] arrayCar = new Car[3];

        arrayCar[0] = new PerformanceCar("Nissan", "Z (RZ34)", 2025, 400, 4, 30,
                14, new String[0]);
        arrayCar[1] = new ShowCar("McLaren", "750S", 2025, 750, 3, 20, 14, 0, 0);
        arrayCar[2] = new PerformanceCar("Toyota", "GR Supra (A90/A91)", 2025,
                382, 2, 30, 14, new String[0]);

        Race race = new CasualRace(1000, "Linear track", 1000, arrayCar);

        winner = race.startRace(arrayCar); // Для простоты эксперимента
                                           // сравнение идёт только по мощности
        System.out.println("\n" + winner + "\n");
        System.out.print("Автомобиль 1, 2 или 3 вы хотите улучшить? ");

        Scanner scanner = new Scanner(System.in);
        garage.setParkedCars(arrayCar);
        garage.modifyCar(scanner.nextInt());

```

```
30
31          System.out.println("\nПовторный заезд...");
32          winner = race.startRace(garage.getParkedCars());
33          System.out.println(winner);
34      }
35  }
```

```
"C:\Program Files\Java\jdk-21.0.2\bin\java.exe" -javaagent:G:\JetBrains\IntelliJIdea2025.1\lib\idea_rt
 .jar=50926 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath
 D:\JavaProject\JavaDevHomeworks\out\production\JavaDevHomeworks homeworks.homework09.App

Победил гонщик на McLaren

Автомобиль 1, 2 или 3 вы хотите улучшить? 1
Какая мощность будет теперь у Nissan? 800

Повторный заезд...
Победил гонщик на Nissan

Process finished with exit code 0
```