МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

Факультет информационных технологий и программирования

Кафедра информационных систем

Лабораторная работа №3

Симулятор гонок

Выполнил студент группы М3201:

Дымчикова Аюна

САНКТ-ПЕТЕРБУРГ

2020

Задание

Разработать примитивный движок для фэнтезийного симулятора гонок.

В симуляторе присутствуют несколько типов транспортных средств:

- двугорбый верблюд;
- верблюд-быстроход;
- кентавр;
- ботинки-вездеходы;
- ковер-самолет;
- ступа;
- метла.

Можно добавить свои собственные типы.

При этом все типы транспортных средств делятся на два класса:

- наземные;
- воздушные.

Наземные типы транспорта обладают следующими характеристиками:

- скорость, в условных единицах;
- время движения до отдыха, в условных единицах;
- длительность отдыха, в условных единицах, задается формулой (зависит от номера остановки по счету).

Воздушные типы транспорта обладают следующими характеристиками:

- скорость, в условных единицах;
- коэффициент сокращения расстояния за счет перелетов, в % от расстояния, задается формулой (зависит от расстояния).

В симуляторе присутствуют несколько типов гонок:

- только для наземного транспорта;
- только для воздушного транспорта;
- для любого типа транспорта.

Движок должен иметь возможность:

1. Создать гонку;
2. Зарегистрировать на гонку транспортное средство в соответствии с допустимым классом транспортного средства (нельзя зарегистрировать воздушное транспортное средство на гонку только для наземных транспортных средств и наоборот);
3. Запустить гонку (определить победителя).

Таблица с характеристиками наземных транспортных средств:

Тип транспортного средства	**Скорость (Speed), усл. ед.**	**Время до отдыха (RestInterval), усл. ед.**	**Длительность отдыха (RestDuration), усл. ед.**
двугорбый верблюд	10	30	первый раз - 5, все последующие разы - 8
верблюд-быстроход	40	10	первый раз - 5, второй раз - 6.5, все последующие разы - 8
кентавр	15	8	всегда 2
ботинки-вездеходы	6	60	первый раз - 10, все последующие разы - 5

Таблица с характеристиками воздушных транспортных средств:

Тип транспортного средства	**Скорость (Speed), усл. ед.**	**Сокращение расстояния (DistanceReducer), %**
ковер-самолет	10	дистанция до 1000 - без сокращения, до 5000 - 3%, до 10000 - 10%, больше 10000 - 5%
ступа	8	всегда 6%
метла	20	равномерно сокращается на 1% за каждую 1000 усл. единиц расстояния

Ход рассуждений

В результате выполнения работы была реализована система классов, где от абстрактного класса Vehicle наследуются классы AirVehicle и LandVehicle, на основе которых реализованы классы для представленных в лабораторной работе видов транспортных средств.

Vehicle - абстрактный класс, выделяющий основные черты транспортных средств, в котором необходимо особо выделить функцию для получения времени завершения гонки, которая переопределяется в его подклассах.

AirVehicle и LandVehicle - абстрактные классы, созданные для разделения воздушных и наземных транспортных средств и выделения черт, характерных для каждого из классов, а также добавления необходимых атрибутов,отличных у этих классов.

Классы Broom, Centaur и т. п. - классы, экземпляры которых используются в качестве непосредственных участников гонки, а также переопределяют функции для подсчета времени завершения гонки в зависимости от характеристик данного вида средства.

Таким образом, реализован примитивный движок для фентезийного симулятора гонок с помощью представленной системы классов, которая удовлетворяет условиям данной лабораторной работы.

Листинг

Файл AirVehicle.java

```
public abstract class AirVehicle extends Vehicle {
    public AirVehicle(int _id, int _speed) throws Exception {
        super(_id, _speed);
    }

    protected abstract double getDistanceReducer(int dist);

    public double getFinishTime(int dist) throws Exception {
        if(dist <= 0) {
            throw new Exception("Wrong distance: " + dist);
        }
        return (getDistanceReducer(dist) * dist) / this.getSpeed();
    }
}
```

Файл AllTerrainBoots.java

```
public class AllTerrainBoots extends LandVehicle {
    public AllTerrainBoots(int _id) throws Exception {
        super(_id, 6, 60);
    }
    protected double getRestDuration(int number) {
        if (number == 1) {
            return 10;
        }
        return 5;
    }
}
```

Файл BactrianCamel.java

```
public class BactrianCamel extends LandVehicle {
    public BactrianCamel(int _id) throws Exception {
        super(_id, 10, 30);
    }
    protected double getRestDuration(int number) {
        if (number == 1) {
            return 5;
        }
        return 8;
    }
}
```

Файл Broom.java

```
public class Broom extends AirVehicle{
        public Broom(int _id) throws Exception {
            super(_id, 20);
        }
        protected double getDistanceReducer(int dist) {
            int percent = dist / 1000;
            return 1 - (double) percent / 100;
        }
    }
```

Файл Centaur.java

```
public class Centaur extends LandVehicle {
    public Centaur(int _id) throws Exception {
        super(_id, 15, 8);
    }

    protected double getRestDuration(int number) {
        return 2;
    }
}
```

Файл FastCamel.java

```
public class FastCamel extends LandVehicle {
    public FastCamel(int _id) throws Exception {
        super(_id, 40, 10);
    }

    protected double getRestDuration(int number) {
        if (number == 1) {
            return 5;
        } else if (number == 2) {
            return 6.5;
        }
        return 8;
    }
}
```

Файл FlyingCarpet.java

```
public class FlyingCarpet extends AirVehicle{
    public FlyingCarpet(int _id) throws Exception {
        super(_id, 10);
    }
    protected double getDistanceReducer(int dist) {
        if(dist < 1000) {
            return 1;
        } else if(dist < 5000) {
            return 0.97;
        } else if(dist < 10000) {
            return 0.9;
        } else {
            return 0.95;
        }
    }
}
```

Файл LandVehicle.java

```
public abstract class LandVehicle extends Vehicle{
    private int restInterval = 1;
    public LandVehicle(int _id, int _speed, int _restInterval) throws Exception {
        super(_id, _speed);
        restInterval = _restInterval;
    }

    protected abstract double getRestDuration(int number);

    protected double getTotalRestDuration(int number) {
        double res = 0;
        for(int i = 1; i <= number; i++) {
            res += getRestDuration(i);
        }
        return res;
    }
    public double getFinishTime(int dist) throws Exception {
        if(dist <= 0) {
            throw new Exception("Wrong distance: " + dist);
        }
        double time = (double) dist / this.getSpeed();
        return time + getTotalRestDuration((int) ((time - 1e-9) / restInterval));
    }
}
```

Файл Main.java

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

        try {
            FlyingCarpet carpet = new FlyingCarpet(0);
            Mortar mortar = new Mortar(1);
            Broom broom = new Broom(2);
            Race<AirVehicle> airRace = new Race<>(1000, carpet, mortar);
            System.out.println("Winner in air race with distance 1000 is " + airRace.getWinner().getId());
            airRace.setDistance(1000);
            System.out.println("Winner in air race with distance 9000 is " + airRace.getWinner().getId());
            airRace.setDistance(11000);
            airRace.addParticipant(broom);
            System.out.println("Winner in air race with distance 11000 is " + airRace.getWinner().getId());

            BactrianCamel bCamel = new BactrianCamel(3);
            FastCamel fCamel = new FastCamel(4);
            Centaur centaur = new Centaur(5);
            AllTerrainBoots boots = new AllTerrainBoots(6);
            Race<LandVehicle> landRace = new Race<>(1000, bCamel, centaur, boots);
            System.out.println("Winner in land race with distance 1000 is " + landRace.getWinner().getId());
            landRace.setDistance(400);
            System.out.println("Winner in land race with distance 400 is " + landRace.getWinner().getId());
            landRace.setDistance(50000);
            landRace.addParticipant(fCamel);
            System.out.println("Winner in land race with distance 50000 is " + landRace.getWinner().getId());

            Race<Vehicle> race = new Race<>(1000, carpet, mortar, broom, bCamel, fCamel, centaur, boots);
            System.out.println("Winner in race for all with distance 1000 is " + race.getWinner().getId());
            race.setDistance(50000);
            System.out.println("Winner in race for all with distance 50000 is " + race.getWinner().getId());

        } catch (Exception e) {
            System.out.println(e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Файл Mortar.java

```
public class Mortar extends AirVehicle {
    public Mortar(int _id) throws Exception {
        super(_id, 8);
    }

    protected double getDistanceReducer(int dist) {
        return 0.94;
    }
}
```

Файл Race.java

```
import java.util.ArrayList;
import java.util.Arrays;

public class Race<T extends Vehicle> {
    private int distance = 0;
    private ArrayList <T> participants;
    public Race(int dist) throws Exception {
        checkDistance(dist);
        participants = new ArrayList<>();
        distance = dist;
    }
    public Race(int dist, T ... challengers) throws Exception {
        checkDistance(dist);
        participants = new ArrayList<>();
        participants.addAll(Arrays.asList(challengers));
        distance = dist;

    }

    public void setDistance(int dist) throws Exception {
        checkDistance(dist);
        distance = dist;
    }

    private void checkDistance(int dist) throws Exception {
        if(dist <= 0) {
            throw new Exception("Wrong distance: " + dist);
        }
    }

    public void addParticipant(T participant) {
        participants.add(participant);
    }
    public T getWinner() throws Exception {
        double bestTime = Double.MAX_VALUE;
        T winner = null;
        for(T participant : participants) {
            double curTime = participant.getFinishTime(distance);
            if(curTime < bestTime) {
                bestTime = curTime;
                winner = participant;
            }
        }
        if(winner != null) {
            return winner;
        } else {
            throw new Exception("No winner found");
        }
    }
```

```
}
```

Файл Vehicle.java

```
public abstract class Vehicle {
    private int speed = 1;
    private int id = 0;
    public Vehicle(int _id, int _speed) throws Exception {
        if(_speed <= 0) {
            throw new Exception("Speed can't be negative");
        }
        speed = _speed;
        id = _id;
    }


    public abstract double getFinishTime(int dist) throws Exception;
    public int getSpeed() {
        return speed;
    }

    public int getId() {
        return id;
    }
}
```