МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №5

Специальность ПО9(з)

Выполнил
Д. Н. Кухарев,
студент группы ПО9

Проверил
А. А. Крощенко,
ст. преп. кафедры ИИТ,
«___k_____2024 г.

Брест 2024

Цель работы: приобрести практические навыки в области объектно-ориентированного проектирования.

Вариант 9

Задание 1. Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов: interface Корабль ← class Грузовой Корабль ← class Танкер.

Выполнение:

Код программы

Main.java:

```
import java.util.ArrayList;

public class Main {
   public static void main(String[] args) {
      ArrayList<Ship> ship_park = new ArrayList<Ship>();
      ship_park.add(new Tanker("Hispaniola"));
      ship_park.add(new Tanker("Admiral Benbow"));
      for(Ship ship : ship_park){
         manageShip(ship);
      }
   }
   public static void manageShip(Ship ship){
      Cargo cargo = new Cargo();
      ship.call();
      ship.load(cargo);
      ship.transportTo("Kidd's Island");
      ship.goToStorage();
      System.out.println();
   }
}
```

Ship.java:

```
public interface Ship {
   public void call();
   public void load(Cargo cargo);
   public void transportTo(String destination);
   public void goToStorage();
}
```

Tanker.java:

```
public class Tanker extends CargoShip{
   String name;
   Tanker(String name){
      this.name = name;
   }
   public void call(){
      System.out.println("Tanker "+ name + " is here!");
   }
   public void load(Cargo cargo){
      cargo.loadShip();
   }
}
```

CargoShip.java:

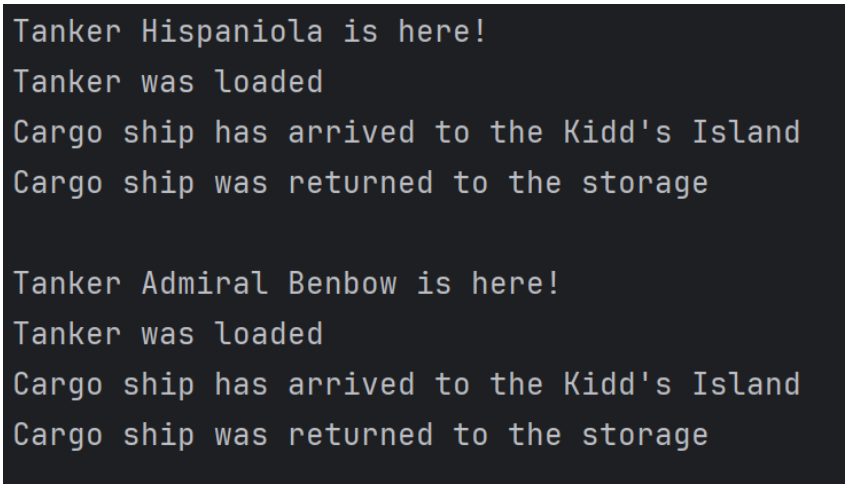
```
public abstract class CargoShip implements Ship{
   public abstract void call();
   public abstract void load(Cargo cargo);
   public void transportTo(String destination){
      System.out.println("Cargo ship has arrived to the " + destination);
   }
```

```
    public void goToStorage(){
        System.out.println("Cargo ship was returned to the storage");
    }
}
```

Cargo.java:

```
public class Cargo {
    public void loadShip(){
        System.out.println("Tanker was loaded");
    }
}
```

Рисунки с результатами работы программы



Задание 2. В следующих заданиях требуется создать суперкласс (абстрактный класс, интерфейс) и определить общие методы для данного класса. Создать подклассы, в которых добавить специфические свойства и методы. Часть методов переопределить. Создать массив объектов суперкласса и заполнить объектами подклассов. Объекты подклассов идентифицировать конструктором по имени или идентификационному номеру. Использовать объекты подклассов для моделирования реальных ситуаций и объектов. Создать суперкласс Транспортное средство и подклассы Автомобиль, Велосипед, Повозка. Подсчитать время и стоимость перевозки пассажиров и грузов каждым транспортным средством.

Выполнение:

Код программы

Main.java:

```
public class Main {
    public static void main(String[] args) {
        Automobile car = new Automobile("McLaren", 4, 100);
        car.carryPassenger(3, 100, Vehicle.TYPE_HIGHWAY);
        car.carryCargo(10, 100, Vehicle.TYPE_HIGHWAY);
        Bicycle bicycle = new Bicycle("Aist");
        bicycle.carryCargo(20, 100, Vehicle.TYPE_HIGHWAY);
        bicycle.carryPassenger(100, Vehicle.TYPE_HIGHWAY);
        Cart cart = new Cart("Povozka", 5, 100);
        cart.carryPassenger(3, 100, Vehicle.TYPE_FOREST);
        cart.carryCargo(50, 100, Vehicle.TYPE_FOREST);
    }
}
```

Vehicle.java:

```
import java.text.DecimalFormat;

public class Vehicle {
   final public static int TYPE_HIGHWAY = 1;
   final public static int TYPE_RUGGED = 2;
   final public static int TYPE_FOREST = 3;
   final public static int TYPE_HILLS = 4;
   final public static int BICYCLE_SEATS = 1;
   final public static int BICYCLE_LIFTING = 20;
   DecimalFormat decimalFormat = new DecimalFormat("#.##");
   String name;
   Vehicle(String name){
      this.name = name;
   }

}
```

Automobile.java:

```
import java.text.DecimalFormat;

public class Automobile extends Vehicle{
   int seats, lifting;
   public Automobile(String name, int seats, int lifting) {
      super(name);
      this.seats = seats;
      this.lifting = lifting;
   }
   public void carryPassenger(int passenger_amount, int distance, int ROAD_TYPE){
      System.out.println("Transportation of " + passenger_amount + " passengers over a distance of " + distance +
" km with Car '" + name + "'");
      double cost;
      if(passenger_amount > seats){
         System.out.println("Too much passengers");
         return;
      }
      switch (ROAD_TYPE){
         case TYPE_HIGHWAY:
            System.out.println("Road type: highway");
            cost = ((double) distance/100.0 * 8);
            System.out.println("Time: " + decimalFormat.format(((double) distance/120.0)) + " h");
            System.out.println("Total cost: $" + decimalFormat.format(cost));
            System.out.println("Cost per passenger: $" + decimalFormat.format(cost/passenger_amount));
            break;
         case TYPE_RUGGED:
            System.out.println("Road type: rugged terrain");
            if(Math.random()*2 > 1){
               System.out.println("Car " + name + " stuck on the rugged road\n");
            }else{
               cost = ((double) distance/100.0 * 14);
               System.out.println("Time: " + decimalFormat.format((double) distance/40.0) + " h");
               System.out.println("Total cost: $" + decimalFormat.format(cost));
               System.out.println("Cost per passenger: $" + decimalFormat.format(cost/passenger_amount));
            }
            break;
         case TYPE_FOREST:
            System.out.println("Road type: forest");
            System.out.println("Car " + name + " stuck in the forest\n");
            break;
         case TYPE_HILLS:
            System.out.println("Road type: hills");
            if(Math.random()*3 > 1) {
```

```
            System.out.println("Car " + name + " stuck on the hill\n");
          }else{
            cost = ((double) distance/100.0 * 20);
            System.out.println("Time: " + decimalFormat.format(((double) distance/20.0)) + " h");
            System.out.println("Total cost: $" + decimalFormat.format(cost));
            System.out.println("Cost per passenger: $" + decimalFormat.format(cost/passenger_amount));
          }
          break;
        default:
          System.out.println("Type doesn't exist");
          break;
      }
      System.out.println();
    }
    public void carryCargo(int cargo_weight, int distance, int ROAD_TYPE){
      System.out.println("Transportation of cargo weighing " + cargo_weight + " kg over a distance of " + distance +
" km with Car '" + name + "'");
      double cost;
      if(cargo_weight > lifting){
        System.out.println("Cargo is too heavy");
        return;
      }
      switch (ROAD_TYPE){
        case TYPE_HIGHWAY:
          System.out.println("Road type: highway");
          cost = ((double) distance/100.0 * 8.0 * (1.0 + (double)cargo_weight/(double)lifting));
          System.out.println("Time: " + decimalFormat.format(((double) distance/120.0)) + " h");
          System.out.println("Transportation cost: $" + decimalFormat.format(cost));
          break;
        case TYPE_RUGGED:
          System.out.println("Road type: rugged terrain");
          if(Math.random()*2 > 1){
            System.out.println("Car " + name + " stuck on the rugged road\n");
          }else{
            cost = ((double) distance/100.0 * 14.0 * (1.0 + (double)cargo_weight/(double)lifting));
            System.out.println("Time: " + decimalFormat.format((double) distance/40.0) + " h");
            System.out.println("Transportation cost: $" + decimalFormat.format(cost));
          }
          break;
        case TYPE_FOREST:
          System.out.println("Road type: forest");
          System.out.println("Car " + name + " stuck in the forest\n");
          break;
        case TYPE_HILLS:
          System.out.println("Road type: hills");
          if(Math.random()*3 > 1) {
            System.out.println("Car " + name + " stuck on the hill\n");
          }else{
            cost = ((double) distance/100.0 * 20.0 * (1.0 + (double)cargo_weight/(double)lifting));
            System.out.println("Time: " + decimalFormat.format(((double) distance/20.0)) + " h");
            System.out.println("Transportation cost: $" + decimalFormat.format(cost));
          }
          break;
        default:
          System.out.println("Type doesn't exist");
          break;
      }
      System.out.println();
    }
}
```

Cart.java:

```
public class Cart extends Vehicle{
   int seats, lifting;
   public Cart(String name, int seats, int lifting) {
      super(name);
      this.seats = seats;
      this.lifting = lifting;
   }
   public void carryPassenger(int passenger_amount, int distance, int ROAD_TYPE){
      System.out.println("Transportation of " + passenger_amount + " passengers over a distance of " + distance +
" km with Cart '" + name + "'");
      double cost;
      if(passenger_amount > seats){
         System.out.println("Too much passengers");
         return;
      }
      switch (ROAD_TYPE){
         case TYPE_HIGHWAY:
            System.out.println("Road type: highway");
            cost = ((double) distance/100.0 * 2.0);
            System.out.println("Time: " + decimalFormat.format(((double) distance/60.0)) + " h");
            System.out.println("Total cost: $" + decimalFormat.format(cost));
            System.out.println("Cost per passenger: $" + decimalFormat.format(cost/passenger_amount));
            break;
         case TYPE_RUGGED:
            System.out.println("Road type: rugged terrain");
            cost = ((double) distance/100.0 * 3.0);
            System.out.println("Time: " + decimalFormat.format(((double) distance/35.0)) + " h");
            System.out.println("Total cost: $" + decimalFormat.format(cost));
            System.out.println("Cost per passenger: $" + decimalFormat.format(cost/passenger_amount));
            break;
         case TYPE_FOREST:
            System.out.println("Road type: forest");
            cost = ((double) distance/100.0 * 2.5);
            System.out.println("Time: " + decimalFormat.format(((double) distance/20.0)) + " h");
            System.out.println("Total cost: $" + decimalFormat.format(cost));
            System.out.println("Cost per passenger: $" + decimalFormat.format(cost/passenger_amount));
            break;
         case TYPE_HILLS:
            System.out.println("Road type: hills");
            cost = ((double) distance/100.0 * 3.5);
            System.out.println("Time: " + decimalFormat.format(((double) distance/15.0)) + " h");
            System.out.println("Total cost: $" + decimalFormat.format(cost));
            System.out.println("Cost per passenger: $" + decimalFormat.format(cost/passenger_amount));
            break;
         default:
            System.out.println("Type doesn't exist");
            break;
      }
      System.out.println();
   }
   public void carryCargo(int cargo_weight, int distance, int ROAD_TYPE){
      System.out.println("Transportation of cargo weighing " + cargo_weight + " kg over a distance of " + distance +
" km with Cart '" + name + "'");
      double cost;
      if(cargo_weight > lifting){
         System.out.println("Cargo is too heavy");
         return;
      }
      switch (ROAD_TYPE){
         case TYPE_HIGHWAY:
            System.out.println("Road type: highway");
```

```
            cost = ((double) distance/100.0 * 2.5 * (1.0 + (double)cargo_weight/(double)lifting));
            System.out.println("Time: " + decimalFormat.format(((double) distance/60.0)) + " h");
            System.out.println("Transportation cost: $" + decimalFormat.format(cost));
            break;
         case TYPE_RUGGED:
            System.out.println("Road type: rugged terrain");
            cost = ((double) distance/100.0 * 3.5 * (1.0 + (double)cargo_weight/(double)lifting));
            System.out.println("Time: " + decimalFormat.format(((double) distance/35.0)) + " h");
            System.out.println("Transportation cost: $" + decimalFormat.format(cost));
            break;
         case TYPE_FOREST:
            System.out.println("Road type: forest");
            cost = ((double) distance/100.0 * 3.0 * (1.0 + (double)cargo_weight/(double)lifting));
            System.out.println("Time: " + decimalFormat.format(((double) distance/20.0)) + " h");
            System.out.println("Transportation cost: $" + decimalFormat.format(cost));
            break;
         case TYPE_HILLS:
            System.out.println("Road type: hills");
            cost = ((double) distance/100.0 * 4.0 * (1.0 + (double)cargo_weight/(double)lifting));
            System.out.println("Time: " + decimalFormat.format(((double) distance/15.0)) + " h");
            System.out.println("Transportation cost: $" + decimalFormat.format(cost));
            break;
         default:
            System.out.println("Type doesn't exist");
            break;
      }
      System.out.println();
   }
}
```

Bicycle.java:

```
import java.text.DecimalFormat;

public class Bicycle extends Vehicle{
   public Bicycle(String name) {
      super(name);
   }
   public void carryPassenger(int distance, int ROAD_TYPE){
      System.out.println("Transportation of passenger over a distance of " + distance + " km with Bicycle '" + name
+ "'");
      double cost;
      switch (ROAD_TYPE){
         case TYPE_HIGHWAY:
            System.out.println("Road type: highway");
            cost = ((double) distance/100.0 * 1);
            System.out.println("Time: " + decimalFormat.format(((double) distance/35.0)) + " h");
            System.out.println("Cost for passenger: $" + decimalFormat.format(cost));
            break;
         case TYPE_RUGGED:
            System.out.println("Road type: rugged terrain");
            cost = ((double) distance/100.0 * 1.5);
            System.out.println("Time: " + decimalFormat.format(((double) distance/20.0)) + " h");
            System.out.println("Cost for passenger: $" + decimalFormat.format(cost));
            break;
         case TYPE_FOREST:
            System.out.println("Road type: forest");
            cost = ((double) distance/100.0 * 1.2);
            System.out.println("Time: " + decimalFormat.format(((double) distance/10.0)) + " h");
            System.out.println("Cost for passenger: $" + decimalFormat.format(cost));
            break;
         case TYPE_HILLS:
            System.out.println("Road type: hills");
```

```
            cost = ((double) distance/100.0 * 2.0);
            System.out.println("Time: " + decimalFormat.format(((double) distance/15.0)) + " h");
            System.out.println("Cost for passenger: $" + decimalFormat.format(cost));
            break;
         default:
            System.out.println("Type doesn't exist");
            break;
      }
      System.out.println();
   }
   public void carryCargo(int cargo_weight, int distance, int ROAD_TYPE){
      System.out.println("Transportation of cargo weighing " + cargo_weight + " kg over a distance of " + distance +
" km with Bicycle '" + name + "'");
      double cost;
      if(cargo_weight > BICYCLE_LIFTING){
         System.out.println("Cargo is too heavy");
         return;
      }
      switch (ROAD_TYPE){
         case TYPE_HIGHWAY:
            System.out.println("Road type: highway");
            cost = ((double) distance/100.0 * 1.0 * (1.0 + (double)cargo_weight/(double)BICYCLE_LIFTING));
            System.out.println("Time: " + decimalFormat.format(((double)
distance/(35.0/(1+(double)cargo_weight/(double)BICYCLE_LIFTING)))) + " h");
            System.out.println("Transportation cost: $" + decimalFormat.format(cost));
            break;
         case TYPE_RUGGED:
            System.out.println("Road type: rugged terrain");
            cost = ((double) distance/100.0 * 1.5 * (1.0 + (double)cargo_weight/(double)BICYCLE_LIFTING));
            System.out.println("Time: " + decimalFormat.format(((double)
distance/(20.0/(1+(double)cargo_weight/(double)BICYCLE_LIFTING)))) + " h");
            System.out.println("Transportation cost: $" + decimalFormat.format(cost));
            break;
         case TYPE_FOREST:
            System.out.println("Road type: forest");
            cost = ((double) distance/100.0 * 1.2 * (1.0 + (double)cargo_weight/(double)BICYCLE_LIFTING));
            System.out.println("Time: " + decimalFormat.format(((double)
distance/(10.0/(1+(double)cargo_weight/(double)BICYCLE_LIFTING)))) + " h");
            System.out.println("Transportation cost: $" + decimalFormat.format(cost));
            break;
         case TYPE_HILLS:
            System.out.println("Road type: hills");
            cost = ((double) distance/100.0 * 2.0 * (1.0 + (double)cargo_weight/(double)BICYCLE_LIFTING));
            System.out.println("Time: " + decimalFormat.format(((double)
distance/(15.0/(1+(double)cargo_weight/(double)BICYCLE_LIFTING)))) + " h");
            System.out.println("Transportation cost: $" + decimalFormat.format(cost));
            break;
         default:
            System.out.println("Type doesn't exist");
            break;
      }
      System.out.println();
   }
}
```

Рисунки с результатами работы программы

```
Road type: highway
Time: 0,83 h
Total cost: $8
Cost per passenger: $2,67

Transportation of cargo weighing 10 kg over a distance of 100 km with Car 'McLaren'
Road type: highway
Time: 0,83 h
Transportation cost: $8,8

Transportation of cargo weighing 20 kg over a distance of 100 km with Bicycle 'Aist'
Road type: highway
Time: 5,71 h
Transportation cost: $2

Transportation of passenger over a distance of 100 km with Bicycle 'Aist'
Road type: highway
Time: 2,86 h
Cost for passenger: $1

Transportation of 3 passengers over a distance of 100 km with Cart 'Povozka'
Road type: forest
Time: 5 h
Total cost: $2,5
Cost per passenger: $0,83

Transportation of cargo weighing 50 kg over a distance of 100 km with Cart 'Povozka'
Road type: forest
Time: 5 h
Transportation cost: $4,5
```

Задание 3. В задании 3 ЛР No4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

Выполнение:

Так как в лабораторной работе No4 уже были использованы абстрактные классы, заменим абстрактный класс Person на интерфейс.

Код программы

Main.java:

```
package RailwayTicketOffice;

public interface Person {
   abstract public void Action();
   abstract public void Menu(String action);
}
```

Вывод и остальные функции не изменяются.

Вывод: приобрел практические навыки в области объектно-ориентированного проектирования.