МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №4

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 указанный класс, включив в него вспомогательный внутренний класс или классы. Создать класс Mobile с внутренним классом, с помощью объектов которого можно хранить информацию о моделях телефонов и их свойствах.

Выполнение:

Код программы

Main.java:

```
public class Main {
    public static void main(String[] args) {
        String[] specs = {"256 GB", "12 GB", "6.7'", "128 MP", "Black abundance", "Good phone for everyday usage. Logical evolution of 25B Pro"};
        Mobile Xiaomi = new Mobile("Xiaomi");
        Xiaomi.addPhone("26B Pro", specs, 2030);
        Xiaomi.addPhone("26B", specs, 2029);
        Xiaomi.addPhone("26B Pro Max", specs, 2031);
        Xiaomi.addPhone("26B Pro X", specs, 2032);
        Xiaomi.addPhone("26A Ultra", specs, 2032);
        Xiaomi.addPhone("27T Super Pro Max X Ultra", specs, 2035);
        Xiaomi.ShowList();
        Xiaomi.AddDescription(3);
        Xiaomi.ShowPhone(3);
    }
}
```

Mobile.java:

```
import java.util.ArrayList;
import java.util.Scanner;
public class Mobile {
    String brand;
    ArrayList<Phone> phones = new ArrayList<Phone>();
    public class Phone{
        String model;
        String[] specs;
        int release_year;
        public Phone(String model, String[] specs, int release_year){
            this.model = model;
            this.specs = specs;
            this.release_year = release_year;
```

```
    }
    public void Show(){
        System.out.print(brand + " " + model + ", (" + release_year + ").");
        System.out.println("\n\tStorage: " + specs[Const.ROM] +
            "\n\tRAM: " + specs[Const.RAM] + "\n\tScreen size: " + specs[Const.SCREEN_SIZE] +
            "\n\tColor: " + specs[Const.COLOR] + "\n\tMain camera: " +
specs[Const.MAIN_CAM_RESOLUTION] +
            "\n\tDescription: \n\t" + specs[Const.ADV_DESCRIPTION]);
        System.out.println();
    }
    public void setSpecs(String[] specs){
        this.specs = specs;
    }
    public String[] MakeDescription(){
        String[] specs = new String[Const.ADV_DESCRIPTION+1];
        Scanner in = new Scanner(System.in);
        System.out.println("Add description for the " + model + ":");
        System.out.print("Storage: "); specs[Const.ROM] = in.nextLine();
        System.out.print("ROM: "); specs[Const.RAM] = in.nextLine();
        System.out.print("Screen size: "); specs[Const.SCREEN_SIZE] = in.nextLine();
        System.out.print("Main camera: "); specs[Const.MAIN_CAM_RESOLUTION] = in.nextLine();
        System.out.print("Color: "); specs[Const.COLOR] = in.nextLine();
        System.out.print("Description: "); specs[Const.ADV_DESCRIPTION] = in.nextLine();
        return specs;
    }
}
public Mobile(String brand){
    this.brand = brand;
}
public void AddDescription(int index){
    phones.get(index).setSpecs(phones.get(index).MakeDescription());
}
public void addPhone(String model, String[] specs, int release_date){
    Phone new_phone = new Phone(model, specs, release_date);
    phones.add(new_phone);
}
public void ShowList(){
    System.out.println("Phones list:");
    for(int i = 0; i < phones.size(); ++i){
        System.out.print(i + "."); phones.get(i).Show();
```

```
        }
        System.out.println();
    }
    public void ShowPhone(int index){
        if(index < 0 || index > phones.size()){
            System.out.println("Incorrect index");
            return;
        }
        System.out.print("Model № " + index + " ");
        phones.get(index).Show();
    }
}
```

Рисунки с результатами работы программы

```
0.Xiaomi 26B Pro, (2030).
    Storage: 256 GB
    RAM: 12 GB
    Screen size: 6.7'
    Color: Black abundance
    Main camera: 128 MP
    Description:
    Good phone for everyday usage. Logical evolution of 25B Pro

1.Xiaomi 26B, (2029).
    Storage: 256 GB
    RAM: 12 GB
    Screen size: 6.7'
    Color: Black abundance
    Main camera: 128 MP
    Description:
    Good phone for everyday usage. Logical evolution of 25B Pro
```

```
Add description for the 26B Pro X:
Storage: 200 GB
ROM: 10 GB
Screen size: 7.0'
Main camera: 45 MP
Color: Aquamarine
Description: Best phone ever
Model № 3 Xiaomi 26B Pro X, (2032).
    Storage: 200 GB
    RAM: 10 GB
    Screen size: 7.0'
    Color: Aquamarine
    Main camera: 45 MP
    Description:
    Best phone ever
```

Задание 2. Реализовать агрегирование. При создании класса агрегируемый класс объявляется как атрибут. Создать класс Автомобиль, используя класс Колесо.

Выполнение:

Код программы

Main.java:

```
public class Main {
   public static void main(String[] args) {
      Wheel normal_wheel1 = new Wheel("Michelin", 16, "passenger", 2.1, 100);
      Wheel normal_wheel2 = new Wheel("Michelin", 16, "passenger", 2.1, 100);
      Wheel invalid_wheel = new Wheel("Michelin", 10, "passenger", 2.1, 100);
      Wheel weak_wheel = new Wheel("Michelin", 15, "until", 2.1, 10);
      Wheel lower_wheel = new Wheel("Michelin", 16, "passenger", 1.001, 100);
      Automobile car1 = null;
      Automobile car2 = null;
      try{
         car1 = new Automobile("McLaren", normal_wheel1, normal_wheel2, invalid_wheel, lower_wheel);
      }catch (IllegalArgumentException ex){
         System.out.println(ex.getMessage());
      }
      try{
         car2 = new Automobile("Нива", normal_wheel1, normal_wheel2, weak_wheel, lower_wheel);
      }catch (IllegalArgumentException ex){
         System.out.println(ex.getMessage());
      }
      if(car2 != null){
         car2.drive(2000);
         car2.maintenance();
         car2.drive(100);
         car2.maintenance();
         car2.drive(20000);
         car2.maintenance();
         car2.OverPump(2);
         car2.OverPump(2);
         car2.OverPump(2);
      }
   }
}
```

Automobile.java:

```
public class Automobile {
   private String name;
   private String car_type;
   private Wheel[] wheels;
```

```
    private boolean isUntil; //докатка
    Automobile(String name, Wheel wheel1, Wheel wheel2, Wheel wheel3, Wheel wheel4){
        if(!compareWheels(wheel1, wheel2) || !compareWheels(wheel2, wheel3) || !compareWheels(wheel3,
wheel4)){
            throw new IllegalArgumentException("Разные диаметры колес "+name+" недопустимы!\n");
        }
        this.name = name;
        this.car_type = "passenger";
        wheels = new Wheel[4];
        wheels[0] = wheel1; wheels[1] = wheel2; wheels[2] = wheel3; wheels[3] = wheel4;
        if(wheel1.getType().equals("until") || wheel2.getType().equals("until") ||
            wheel3.getType().equals("until") || wheel4.getType().equals("until")){
            isUntil = true;
        }else{
            isUntil = false;
        }
        System.out.println(name + " готов к поездке!\n");
    }
    Automobile(String name, Wheel wheel1, Wheel wheel2, Wheel wheel3, Wheel wheel4, Wheel wheel5, Wheel
wheel6){
        if(!compareWheels(wheel1, wheel2) || !compareWheels(wheel2, wheel3) || !compareWheels(wheel3,
wheel4) || !compareWheels(wheel4, wheel5) || !compareWheels(wheel5, wheel6)){
            throw new IllegalArgumentException("Разные диаметры колес недопустимы!");
        }
        this.name = name;
        this.car_type = "cargo";
        wheels = new Wheel[6];
        wheels[0] = wheel1; wheels[1] = wheel2; wheels[2] = wheel3; wheels[3] = wheel4; wheels[4] = wheel5;
wheels[5] = wheel6;
        if(wheel1.getType().equals("until") || wheel2.getType().equals("until") ||
            wheel3.getType().equals("until") || wheel4.getType().equals("until") ||
            wheel5.getType().equals("until") || wheel6.getType().equals("until")){
            isUntil = true;
        }else{
            isUntil = false;
        }
    }
    public void drive(int distance){
        System.out.println("В путь");
        for(int i = 0; i < wheels.length; ++i){
            if(wheels[i].getState() == Wheel.UNTORN){
                System.out.println("Шина изношена, машина не может ехать\n");
                return;
            }else if(wheels[i].getState() == Wheel.LOWERED){
                System.out.println("Шина спущена, машина не может ехать\n");
                return;
```

```
        }
      }
      System.out.println("Едем!");
      for(int i = 0; i < wheels.length; ++i){
        if(wheels[i].Wear(distance) == 1){
          System.out.println("Шина "+(i+1)+" износилась");
        }
        if(wheels[i].PressureDown(distance) == 1){
          System.out.println("Шина "+(i+1)+" спущена");
        }
      }
      System.out.println();
    }
    public void maintenance(){
      Wheel until;
      int count_new = 0, count_pump = 0;
      for(int i = 0; i < wheels.length; ++i){
        if(wheels[i].getState() == Wheel.UNTORN){
          System.out.println("Шина "+(i+1)+" заменена");
          ++count_new;
          until = new Wheel();
          wheels[i] = until;
        }
        if(wheels[i].getState() == Wheel.LOWERED){
          ++count_pump;
          wheels[i].PumpUp();
        }
        System.out.println("Шина "+(i+1)+": Все в порядке");
      }
      System.out.println("Количество услуг: " + (count_new+count_pump)+"\n-
"+(count_new*150+count_pump*20+10)+"$\n");
      System.out.println();
    }
    public void OverPump(int wheel_num){
      wheels[wheel_num].PumpUp();
    }
    private boolean compareWheels(Wheel wheel1, Wheel wheel2){
      if((wheel1.getDiameter() == wheel2.getDiameter())){
        return true;
      }else if((wheel1.getDiameter() != wheel2.getDiameter()) && (wheel1.getType().equals("until") ||
wheel2.getType().equals("until"))){
        return true;
      }else{
        return false;
      }
    }
```

}

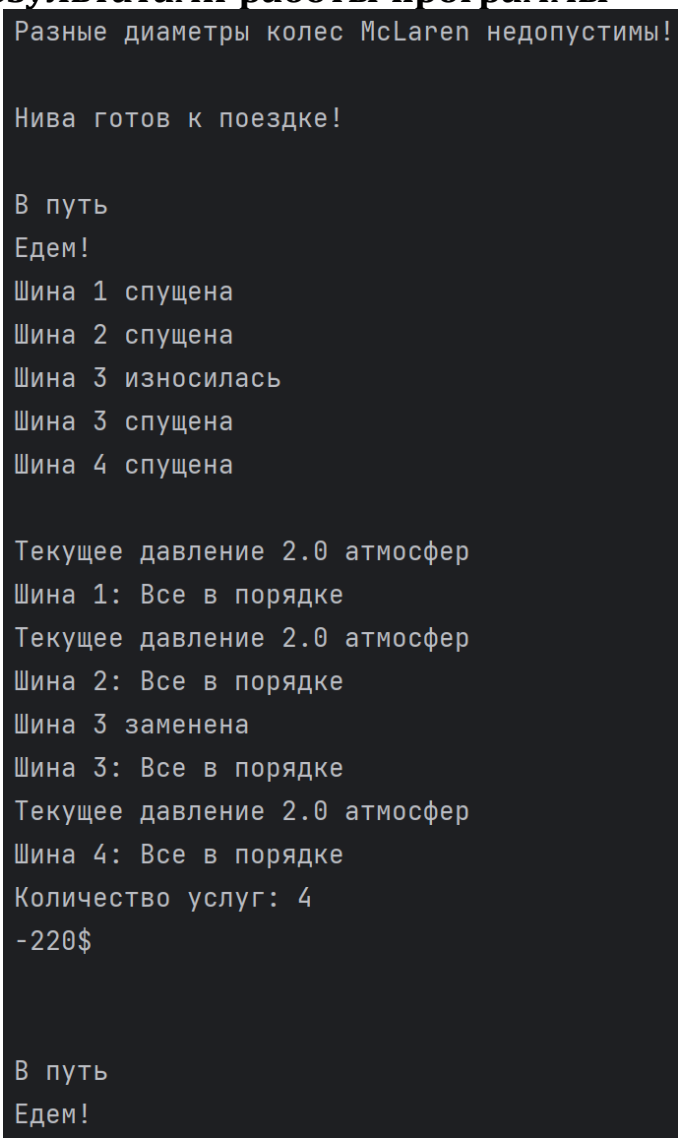
Wheel.java:

```
public class Wheel {
    final static int OK = 0;
    final static int UNTORN = 1;
    final static int LOWERED = 2;
    private String brand;
    private String mark;
    private String type;
    private int diameter;
    private double pressure;
    private double resource;
    private boolean isUntorn;
    private boolean isLowered;
    Wheel(){
        this.brand = "БелШина";
        this.mark = "195/65 R15 91 T RSC";
        this.type = "until";
        this.diameter = 15;
        this.pressure = 2.0;
        this.resource = 100;
        isUntorn = false;
        isLowered = false;
    }
    Wheel(String brand, int diameter, String type, double pressure, double resource){
        this.brand = brand;
        if(type.equals("until")){
            this.mark = "195/65 R"+diameter+" 91 T RSC";
        }else{
            this.mark = "195/65 R"+diameter+" 91 T XL";
        }
        this.type = type;
        this.diameter = diameter;
        this.pressure = pressure;
        this.resource = resource;
        isUntorn = false;
        isLowered = false;
    }
    public String getType(){
        return type;
```

```
}
public int getDiameter(){
   return diameter;
}
public int getState(){
   if(isUntorn){
      return UNTORN;
   }else if(isLowered){
      return LOWERED;
   }else {
      return OK;
   }
}
public int Wear(double distance){//изнашивание в пути
   resource -= distance/100;
   if(resource < 0){
      isUntorn = true;
      return 1;
   }
   return 0;
}
public int PressureDown(int distance){
   pressure -= distance/1000;
   if(pressure < 1.0){
      isLowered = true;
      return 1;
   }
   return 0;
}
public void PumpUp(){
   if(pressure < 2.0){
      pressure = 2.0;
      System.out.println("Текущее давление 2.0 атмосфер");
   }else if (pressure < 2.3){
      pressure = 2.3;
      System.out.println("Текущее давление 2.3 атмосферы");
   }else if(pressure < 3.0){
      pressure = 3.0;
      System.out.println("Текущее давление 3.0 атмосфер, шина перекачана");
   }else {
```

```
            System.out.println("Шина лопнула");
            isUntorn = true;
        }
        isLowered = false;
    }
}
```

Рисунки с результатами работы программы



Задание 3. Построить модель программной системы с применением отношений (обобщения, агрегации, ассоциации, реализации) между классами. Задать атрибуты и методы классов.

Система Железнодорожная касса. Пассажир делает Заявку на станцию назначения, время и дату поездки. Система регистрирует Заявку и осуществляет поиск подходящего Поезда.

Пассажир делает выбор Поезда и получает Счет на оплату. Администратор вводит номера поездов, промежуточные и конечные станции, цены.

Выполнение:

Код программы

Main.java:

```
import RailwayTicketOffice.*;

import java.io.Console;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;

public class Main {
   static ArrayList<Administrator> admins;
   public static void main(String[] args) {
      admins = new ArrayList<Administrator>();
      Scanner read = new Scanner(System.in);
      System.out.println("Enter 'start' to begin");
      String action = "";
      while(!action.equals("quit")){
         action = read.next();
         Menu(action);
      }}
   public static void Menu(String action){
      Scanner read = new Scanner(System.in);
      String name = "";
      int index = 0;
      Console console = System.console();
      if (console == null) {
         System.err.println("Консоль недоступна");
         System.exit(1);
      }
      switch (action){
         case "quit":
            return;
         case "1":
            System.out.print("Enter administrator's name: ");
            name = read.next();
            System.out.println();
            char[] passwordArray = console.readPassword("Enter new password: ", "*");
            String password = new String(passwordArray);
            admins.add(new Administrator(name, password));
            Arrays.fill(passwordArray, ' ');

            break;
         case "2":
            showAdministrators();
            break;
         case "3":
            showAdministrators();
            System.out.print("Enter administrator's number: ");
            index = read.nextInt()-1;
            Administrator admin;
            try{
               admin = admins.get(index);
            }catch (Exception ex){
               System.out.println("No such administrator");
               Menu("2");
               break;
            }
```

```
            admin.Action();
            break;
         case "4":
            Passenger user = new Passenger();
            user.Action();
            break;
         default:
            System.out.println("1 - Add administrator\n2 - Show administrators\n3 - Login as 'Administrator'\n4 -
Enter as user\n%any% - help\nquit to exit");
            break;
      }
   }
   public static void showAdministrators(){
      System.out.println("Administrators list: ");
      int i = 0;
      for(Administrator current : admins){
         ++i;
         System.out.println("\t"+i+". "+current.getName());
      }
   }

}
```

RailwayTicketOffice:

Administrator.java:

```
   package RailwayTicketOffice;

import RailwayTicketOffice.Person;

import java.util.Scanner;

public class Administrator extends Person {
   private String name;
   private String password;
   public Administrator(String name, String password){
      type = Const.ADMINISTRATOR;
      this.name = name;
      this.password = password;
   }
   public void Action(){
      Scanner read = new Scanner(System.in);
      String action = "";
      System.out.print("Enter password: ");
      action = read.next();
      System.out.println();
      if(!password.equals(action)){
         System.out.println("Wrong password");
         return;
      }
      System.out.println("\tADMINISTRATOR");
      do{
         action = read.next();
         Menu(action);
      }
      while(!action.equals("quit"));
   }
   public void Menu(String action){
      Scanner read = new Scanner(System.in);
      switch (action.toLowerCase()){
         case "1":
            RailwayOffice.showSchedule();
            break;
```

```
        case "2":
          RailwayOffice.AddParagraph(this);
          break;
        case "3":
          StationTree.show();
          break;
        case "4":
          StationTree.show();
          System.out.println("Enter parent station");
          String parent = read.nextLine();
          System.out.println("Enter new station name");
          String new_station = read.nextLine();
          StationTree.insert(parent, new_station);
          break;
        case "5":
          RailwayOffice.addTrain();
          break;
        case "6":
          ShowTrainList();
          break;
        case "7":
          ShowTrainList();
          System.out.println("Enter train number in list to delete (-1 to cancel)");
          int number = read.nextInt();
          if(number == -1){
            return;
          }
          RailwayOffice.trains.remove(number);
          break;
        case "quit":
          return;
        default:
          System.out.println("1 - Show schedule\n2 - Add paragraph\n3 - Show stations tree\n4 - Add station\n5 -
Add train\n6 - Show train list\n7 - Remove train\n%any% - help\nquit to exit");
          break;
      }
    }
    public String getName(){
      return name;
    }
    void ShowTrainList(){
      for(int i = 0; i < RailwayOffice.trains.size(); ++i){
        System.out.println(i + ". Train number: " + RailwayOffice.trains.get(i).getNumber() + " with capacity: " +
RailwayOffice.trains.get(i).getCapacity());
      }
    }

}
```

Passenger.java:

```
  package RailwayTicketOffice;

import RailwayTicketOffice.Person;

import javax.sound.midi.Soundbank;
import java.util.Scanner;
import java.util.concurrent.ScheduledFuture;

public class Passenger extends Person {
    public Passenger(){
      type = Const.USER;
    }
```

```
    public void Action(){
        System.out.println("\tUser");
        Scanner read = new Scanner(System.in);
        String action = "";
        while(!action.equals("quit")){
            action = read.next();
            Menu(action);
        }
    }
    public void Menu(String action){//реализация
        switch (action){
            case "quit":
                return;
            case "1":
                RailwayOffice.showSchedule();
                break;
            case "2":
                RailwayOffice.findsuitableTrain();
                break;
            case "3":
                RailwayOffice.chooseTicket(RailwayOffice.findsuitableTrain(), this);
                break;
            default:
                System.out.println("1 - Show schedule\n2 - Find suitable trains\n3 - Make request\n%any% - help\nquit
to exit");
                break;
        }
    }
}
```

Person.java:

```
    package RailwayTicketOffice;

import com.sun.tools.javac.Main;

import java.util.Scanner;

abstract public class Person {
    int type;
    abstract public void Action();
    abstract public void Menu(String action);
}
```

RailwayOffice.java:

```
package RailwayTicketOffice;

import javax.sound.midi.Soundbank;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Locale;
import java.util.Scanner;

public class RailwayOffice {
    static public ArrayList<Train> trains = new ArrayList<Train>();
    static int bill_number = 0;

    public static void AddParagraph(Person person){
        if(person.type != Const.ADMINISTRATOR){
            System.out.println("You don't have a permission");
            return;
```

```
    }
    isExit = false;
    System.out.println("\tAdd paragraph ('-1' to exit)");
    Schedule.addParagraph(readStation(), readTrain(), readDateTime(), readTicketsAmount(), readTicketPrice());
}

public static String readStation(){
    Scanner read = new Scanner(System.in);
    System.out.print("Enter station name: ");
    String station_name = read.next();
    if(station_name.equals("-1")){
        isExit = true;
        return null;
    }
    if(StationTree.findParent(station_name) == null){
        System.out.println(station_name + " station doesn't exist, but you can create it.");
        return readStation();
    }else{
        return station_name;
    }
}
public static Train readTrain(){
    int train_number = 0;
    if(isExit){
        return null;
    }
    Scanner read = new Scanner(System.in);
    System.out.print("Enter train number: ");
    try{
        train_number = read.nextInt();
    }catch(Exception ex){
        System.out.println("Wrong number");
        return readTrain();
    }
    if(train_number == -1){
        isExit = true;
        return null;
    }
    int find = findTrain(train_number);
    if(find < 0){
        System.out.println("Train №"+train_number+" doesn't exist, but you can create it.");
        return readTrain();
    }
    return trains.get(find);
}
public static LocalDateTime readDateTime(){
    if(isExit){
        return null;
    }
    Scanner read = new Scanner(System.in);
    System.out.print("Enter date in next format '2024-01-01': ");
    String dateString = read.next();
    if(dateString.equals("-1")){
        isExit = true;
        return null;
    }
    System.out.print("Enter time in next format '12:00:00': ");
    String timeString = read.next();
    if(timeString.equals("-1")){
        isExit = true;
        return null;
    }
```

```
    LocalDate date;
    LocalTime time;
    LocalDateTime dateTime = LocalDateTime.now();
    try{
        date = LocalDate.parse(dateString);
    }
    catch (Exception ex){
        System.out.println("Wrong date format");
        return readDateTime();
    }
    try{
        time = LocalTime.parse(timeString);
    }
    catch (Exception ex){
        System.out.println("Wrong time format");
        return readDateTime();
    }
    dateTime = LocalDateTime.of(date, time);
    return dateTime;
}
public static int readTicketsAmount(){
    if(isExit){
        return -1;
    }
    int tickets_amount = 0;
    Scanner read = new Scanner(System.in);
    System.out.print("Enter tickets amount: ");
    try{
        tickets_amount = read.nextInt();
    }catch(Exception ex){
        System.out.println("Wrong number");
        return readTicketsAmount();
    }
    if(tickets_amount == -1){
        isExit = true;
        return tickets_amount;
    }
    return tickets_amount;
}
public static int readTicketPrice(){
    int ticket_price = 0;
    if(isExit){
        return -1;
    }
    Scanner read = new Scanner(System.in);
    System.out.print("Enter ticket price: ");
    try{
        ticket_price = read.nextInt();
    }catch(Exception ex){
        System.out.println("Wrong number");
        return readTicketPrice();
    }
    if(ticket_price == -1){
        isExit = true;
        return ticket_price;
    }
    return ticket_price;
}
public static void addTrain(){
    Scanner read = new Scanner(System.in);
    int capacity = 0;
    System.out.print("Enter train capacity: ");
```

```
        try{
            capacity = read.nextInt();
        }catch(Exception ex){
            System.out.println("Wrong capacity");
            addTrain();
        }
        Train new_train = new Train(capacity);
        trains.add(new_train);
    }
    public static void removeTrainByNumber(int train_number){
        int index = findTrain(train_number);
        if(index > -1){
            trains.remove(index);
        }
    }
    public static int findTrain(int number){
        if(trains == null){
            return -1;
        }
        for(int i = 0; i < trains.size(); ++i){
            if(trains.get(i).getNumber() == number){
                return i;
            }
        }
        return -1;
    }
    public static ArrayList<Integer> findsuitableTrain(){
        LocalDate currentDate = LocalDate.now();
        LocalTime currentTime = LocalTime.now();
        String station;
        String date, time;
        Scanner read = new Scanner(System.in);
        System.out.println("Ticket search:");
        System.out.print("Enter station name: "); station = read.next();
        if(StationTree.findParent(station) == null){
            System.out.println("Can't find station");
            findsuitableTrain();
        }
        System.out.print("Enter date in format \"2000-01-01\" or press [Enter] to choose current: "); date =
read.next();
        if(date.length() > 8){
            try {
                currentDate = LocalDate.parse(date);
            }catch(Exception ex){
                System.out.println("Wrong date");
                findsuitableTrain();
            }
        }
        System.out.print("Enter time in format \"12:00:00\" or press [Enter] to choose current: "); time = read.next();
        if(date.length() > 6){
            try {
                currentTime = LocalTime.parse(time);
            }catch(Exception ex){
                System.out.println("Wrong time");
                findsuitableTrain();
            }
        }
        LocalDateTime dateTime = LocalDateTime.of(currentDate, currentTime);
        return Schedule.findTicket(station, dateTime);
    }
    public static void chooseTicket(ArrayList<Integer> indexes, Passenger passenger){
        int number = 0; boolean isHas = false;
```

```
    String bill = "";
    Scanner read = new Scanner(System.in);
    System.out.print("Enter ticket number (-1 to exit): ");
    try{
        number = read.nextInt()-1;
    }catch (Exception ex){
        System.out.println("Wrong number");
    }
    if(number == -1){
        return;
    }
    for(int i = 0; i < indexes.size(); ++i){
        if(number == indexes.get(i)){
            isHas = true;
        }
    }
    if(!isHas){
        System.out.println("Index not in list");
        chooseTicket(indexes, passenger);
    }
    System.out.println("Your route: ");
    ArrayList<String> path = new ArrayList<String>();
    path = StationTree.showToRoot(Schedule.station.get(number));
    ++bill_number;
    DateTimeFormatter date_formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
    DateTimeFormatter time_formatter = DateTimeFormatter.ofPattern("HH:mm:ss");

    Schedule.setTicketsAmount(number, Schedule.getTicketsAmount(number)-1);
    bill = "\tBill №000" + bill_number + "\nTicket number: " + Schedule.tickets_amount.get(number) +
            "\nDate: " + Schedule.date_time.get(number).format(date_formatter) +
            "\nTime: " + Schedule.date_time.get(number).format(time_formatter) +
            "\nDestination: " + Schedule.station.get(number) + "\nRoute: ";
    for(int i = path.size()-1; i >= 0; --i){
        if(i > path.size()-2){
            bill += path.get(i)+" -> ";
        }else{
            bill += path.get(i);
        }
    }
    bill += "\nPrice: $" + Schedule.ticket_price.get(number);

    System.out.println(bill);
    System.out.println("\nDo you want to print your 'Bill'(y/n)?");
    if(!read.next().toLowerCase().equals("n")){
        IFile.WriteFile(bill, "output.txt");
    }
}

public static void showSchedule(){
    Schedule.ShowSchedule();
}
private static class Schedule{
    static private ArrayList<String> station = new ArrayList<String>();
    static private ArrayList<Train> train = new ArrayList<Train>();
    static private ArrayList<LocalDateTime> date_time = new ArrayList<LocalDateTime>();
    static private ArrayList<Integer> tickets_amount = new ArrayList<Integer>();
    static private ArrayList<Integer> ticket_price = new ArrayList<Integer>();

    public static void ShowSchedule(){
        if(station == null){
            System.out.println("Schedule is empty");
            return;
```

```
        }
        System.out.println("\tSchedule");
        DateTimeFormatter date_formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        DateTimeFormatter time_formatter = DateTimeFormatter.ofPattern("HH:mm:ss");
        String formattedDate, formattedTime;
        for(int i = 0; i < station.size(); ++i){
           formattedDate = date_time.get(i).format(date_formatter);
           formattedTime = date_time.get(i).format(time_formatter);
           System.out.println((i+1)+"."+station.get(i)+"\n -Date: "+formattedDate+"\n -Time: "+formattedTime+"\n
-Ticket price: "+ticket_price.get(i) +"\n -Tickets amount: "+tickets_amount.get(i) + "\n -Train number:
"+train.get(i).getNumber());
        }
     }
     public static void addParagraph(String newstation, Train newtrain, LocalDateTime dateTime, Integer
newtickets_amount, Integer newticket_price){
        if(isExit){
           System.out.println("Exit");
           isExit = false;
           return;
        }
        boolean isAdded = false;
        for(int i = 0; i < station.size(); ++i){
           if(dateTime.isBefore(date_time.get(i)) && !isAdded){
              isAdded = true;
              station.add(i, newstation);
              train.add(i, newtrain);
              date_time.add(i, dateTime);
              tickets_amount.add(i, newtickets_amount);
              ticket_price.add(i, newticket_price);
           }
        }
        if(!isAdded){
           station.add(newstation);
           train.add(newtrain);
           date_time.add(dateTime);
           tickets_amount.add(newtickets_amount);
           ticket_price.add(newticket_price);
        }
        System.out.println("Paragraph added");
     }
     public static void setTicketsAmount(int index, int value){
        tickets_amount.set(index, value);
     }
     public static int getTicketsAmount(int index){
        return tickets_amount.get(index);
     }
     private static ArrayList<Integer> findTicket(String station_name, LocalDateTime time){
        ArrayList<Integer> indexes = new ArrayList<Integer>();
        DateTimeFormatter date_formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        DateTimeFormatter time_formatter = DateTimeFormatter.ofPattern("HH:mm:ss");
        String formattedDate, formattedTime;
        boolean isFound = false;
        for(int i = 0; i < station.size(); ++i){
           if(station.get(i).equals(station_name) && date_time.get(i).isAfter(time) && tickets_amount.get(i) > 0){
              isFound = true; indexes.add(i);
              formattedDate = date_time.get(i).format(date_formatter);
              formattedTime = date_time.get(i).format(time_formatter);
              System.out.println((i+1)+"."+station.get(i)+"\n -Date: "+formattedDate+"\n -Time:
"+formattedTime+"\n -Ticket price: "+ticket_price.get(i) +"\n -Tickets amount: "+tickets_amount.get(i) + "\n -Train
number: "+train.get(i).getNumber());
           }
        }
```

```
            if(!isFound){
                System.out.println("Cannot find match");
            }
            return indexes;
        }
    }
}
```

RailwayOffice.java:

```
package RailwayTicketOffice;

import java.awt.image.AreaAveragingScaleFilter;
import java.util.ArrayList;

public class StationTree {
    static private Node root = root = new Node("Technical University");//корневая станция
    static private int level = 0;
    public StationTree(){

    }
    public static void insert(String name, String station){
        Node parent = findParent(name);
        if(parent == null){
            System.out.println("Station not found");
            return;
        }
        Node child = findParent(station);
        if(child != null){
            System.out.println("Station already exist");
            return;
        }
        Node new_node = new Node(station);
        new_node.setParent(parent);
        parent.setChild(new_node);
        System.out.println("Success");
    }
    public static Node findParent(String name){
        if(name.equals(root.getStation())){
            return root;
        }
        return findParent(root, name);
    }
    public static Node findParent(Node current, String name){
        if (current == null) {
            return null;
        }
        if (current.getStation().equals(name)) {
            return current;
        }
        ArrayList<Node> child = current.getChildren();
        if(child == null){
            return null;
        }
        for (int i = 0; i < child.size(); ++i) {
            Node foundNode = findParent(child.get(i), name);
            if (foundNode != null) {
                return foundNode;
            }
        }
        return null;
    }
    public static void show(){
```

```
        show(root);
    }
    public static void show(Node current){
        String symb = " ├─";
        ++level;
        if (current == null) {
            return;
        }
        System.out.println(current.getStation());
        ArrayList<Node> child = current.getChildren();

        if(child == null){
            return;
        }
        for (int i = 0; i < child.size(); ++i) {
            if(i == child.size()-1){
                symb = " └─";
            }
            for(int j = 0; j < level*2; ++j)
                System.out.print("  ");
            System.out.print(symb);
            show(child.get(i));
            //System.out.print(child.get(i).getStation());
        }
        --level;
        return;
    }
    public static ArrayList<String> showToRoot(String name){
        Node parent = findParent(name);
        if(parent == null){
            System.out.println("Station not found");
            return null;
        }
        ArrayList<String> path = new ArrayList<String>();
        System.out.println("Path to station '" + name + "': ");
        path.add(name);
        showToRoot(parent, path);
        String tab = "";
        for(int i = path.size()-1; i > -1; --i){
            if(i == path.size()-1){
                System.out.println(path.get(i));
            }else{
                tab+="  ";
                System.out.println(tab+" └─"+path.get(i));
            }
        }
        return path;
    }
    public static void showToRoot(Node grandParent, ArrayList<String> path){
        Node parent = grandParent.getParent();
        path.add(parent.getStation());
        if(!parent.equals(root)){
            showToRoot(parent, path);
        }
        return;
    }
}
```

RailwayOffice.java:

```
package RailwayTicketOffice;
public class Train {
    static private int next_number = 0;
```

```
    final private int number;
    private int capacity;

    public Train(int capacity){
        ++next_number;
        number = next_number;
        this.capacity = capacity;
    }
    public int getNumber(){
        return number;
    }
    public int getCapacity(){
        return capacity;
    }
    public void boardPassenger(){
        if(capacity < 1){
            System.out.println("Train is full");
            return;
        }
        --capacity;
    }
}
```

Рисунки с результатами работы программы

```
start
1 - Add administrator
2 - Show administrators
3 - Login as 'Administrator'
4 - Enter as user
%any% - help
quit to exit
1
Enter administrator's name: Admi

Enter new password:
/
1 - Add administrator
2 - Show administrators
3 - Login as 'Administrator'
4 - Enter as user
%any% - help
quit to exit
3
Administrators list:
        1. Admin
Enter administrator's number: 1
Enter password: adm1n
```

```
        ADMINISTRATOR
/
1 - Show schedule
2 - Add paragraph
3 - Show stations tree
4 - Add station
5 - Add train
6 - Show train list
7 - Remove train
%any% - help
quit to exit
5
Enter train capacity: 100
```

```
4
Technical University
    ├─Moskovskaya
    └─Zelenaya
        └─Savushkin
Enter parent station
Savushkin
Enter new station name
Kobrin
Success
```

```
2
        Add paragraph ('-1' to exit)
Enter station name: Moskovskaya
Enter train number: 3
Enter date in next format '2024-01-01': 2024-01-01
Enter time in next format '12:00:00': 23:00:23
Enter tickets amount: 15
Enter ticket price: 4
Paragraph added
```

```
1 - Add administrator
2 - Show administrators
3 - Login as 'Administrator
4 - Enter as user
%any% - help
quit to exit
4
        User
/
1 - Show schedule
2 - Find suitable trains
3 - Make request
%any% - help
quit to exit
```

```
2
Ticket search:
Enter station name: Kobrin
Enter date in format "2000-01-01" or press [Enter] to choose current: 2024-01-01
Enter time in format "12:00:00" or press [Enter] to choose current: 18:00:00
2.Kobrin
 -Date: 2024-01-01
 -Time: 18:50:12
 -Ticket price: 14
 -Tickets amount: 20
 -Train number: 2
3.Kobrin
 -Date: 2024-01-01
 -Time: 19:12:22
 -Ticket price: 15
 -Tickets amount: 15
 -Train number: 1
```

```
3
Ticket search:
Enter station name: Kobrin
Enter date in format "2000-01-01" or press [Enter] to choose current: 2024-01-01
Enter time in format "12:00:00" or press [Enter] to choose current: 18:50:00
2.Kobrin
 -Date: 2024-01-01
 -Time: 18:50:12
 -Ticket price: 14
 -Tickets amount: 20
 -Train number: 2
3.Kobrin
 -Date: 2024-01-01
 -Time: 19:12:22
 -Ticket price: 15
 -Tickets amount: 15
 -Train number: 1
Enter ticket number (-1 to exit): 2
Your route:
Path to station 'Kobrin':
Technical University
  └─Zelenaya
    └─Savushkin
      └─Kobrin
        Bill №0001
Ticket number: 19
Date: 2024-01-01
Time: 18:50:12
Destination: Kobrin
Route: Technical University -> ZelenayaSavushkinKobrin
Price: $14

Do you want to print your 'Bill'(y/n)?
```

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