МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

 «БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ИИТ

ЛАБОРАТОРНАЯ РАБОТА №4

Выполнил:

Студент ФЭИС

3-го курса, группы ПО-5

Пищик А.В

Проверил:

Крощенко А. А.

Брест 2021

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

Задание 1

Реализовать указанный класс, включив в него вспомогательный внутренний класс или классы.

Реализовать 2-3 метода (на выбор). Продемонстрировать использование реализованных классов.

Создать класс Payment (покупка) с внутренним классом, с помощью объектов которого можно

сформировать покупку из нескольких товаров.

Задание 2

Реализовать агрегирование. При создании класса агрегируемый класс объявляется как атрибут

(локальная переменная, параметр метода). Включить в каждый класс 2-3 метода на выбор. Проде-

монстрировать использование разработанных классов.

1. Создать класс Звездная система, используя классы Планета, Звезда.

Задание 3

11) Система Аэрофлот. Администратор формирует летную Бригаду (пилоты, штурман, ра-

дист, стюардессы) на Рейс. Каждый Рейс выполняется Самолетом с определенной вмести-

мостью и дальностью полета. Рейс может быть отменен из-за погодных условий в Аэропорту

отлета или назначения. Аэропорт назначения может быть изменен в полете из-за технических

неисправностей, о которых сообщил командир.

**Код программы:**

import java.util.ArrayList;

import java.util.List;

public class task1\_lab4 {

public static void main(String[] args) {

Payment payment = new Payment();

Payment.Product productFirst = payment.new Product("Семена", 123);

Payment.Product productSecond = payment.new Product("Соль", 1);

Payment.Product productThird = payment.new Product("Макароны", 25);

Payment.Product productForth = payment.new Product("Водка", 6);

Payment.Product productFifth = payment.new Product("Влад", 3);

payment.buyProduct(productFirst);

payment.buyProduct(productSecond);

payment.buyProduct(productThird);

payment.buyProduct(productForth);

payment.buyProduct(productFifth);

System.out.println(payment.sum());

}

}

class Payment {

private List<Product> productList;

public Payment() {

productList = new ArrayList<>();

}

public Product buyProduct(Product product){

productList.add(product);

return product;

}

public int sum(){

return productList.stream().mapToInt(Product::getPrice).sum();

}

public class Product {

private String name;

private Integer price;

public Integer getPrice() {

return price;

}

public Product(String name, Integer price) {

this.name = name;

this.price = price; }

@Override

public String toString() {

return "Product{" +

"name='" + name + '\'' +

", price=" + price +

'}';

}

}

}

TASK2

import java.util.ArrayList;

import java.util.List;

public class task2\_lab4 {

public static void main(String[] args) {

StarsSystem starSystem = new StarsSystem("Солнечная");

starSystem.addPlanet(new Planet("Венера", false));

starSystem.addPlanet(new Planet("Сатурн", false));

starSystem.addPlanet(new Planet("Венера", false));

starSystem.addPlanet(new Planet("Меркурий", false));

starSystem.addPlanet(new Planet("Земля", true));

starSystem.addPlanet(new Planet("Марс", false));

starSystem.addStar(new Star("Солнце"));

starSystem.deletePlanet(5);

starSystem.printText();

starSystem.printAlive();

}

}

class StarsSystem {

private String name;

private List<Planet> planetList = new ArrayList<>();

private List<Star> starList = new ArrayList<>();

public StarsSystem(String name){

this.name = name;

}

public void addPlanet(Planet planet) {

planetList.add(planet);

}

public void deletePlanet(int planetId) {

planetList.remove(planetId);

}

public void addStar(Star star) {

starList.add(star);

}

public void printAlive(){

System.out.println("Планеты, населенные жизнью:");

planetList.stream().filter(Planet::getAlive).forEach(Planet::printPlanet);

}

public void deleteStar(int starId) {

starList.remove(starId);

}

public void printText() {

System.out.println("Система: " + this.name);

System.out.println("Планеты:" );

planetList.forEach(Planet::printPlanet);

System.out.println("Звезды: " );

starList.forEach(Star::printStar);

}

}

class Planet {

private String name;

private Boolean isAlive;

public Planet(String name, Boolean isAlive) {

this.name = name;

this.isAlive = isAlive;

}

public Boolean getAlive() {

return isAlive;

}

public void printPlanet() {

System.out.println("\t" + this.name);

}

}

class Star {

private String name;

public Star(String name) {

this.name = name;

}

public void printStar() {

System.out.println("\t" + this.name);

}

}

Task3

import java.time.\*;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import java.util.concurrent.atomic.AtomicInteger;

public class task3\_lab4 {

public static void main(String[] args) {

Aeroflot aeroflot1 = new Aeroflot("Сингапур");

Aeroflot aeroflot2 = new Aeroflot("Лондон");

Aeroflot aeroflot3 = new Aeroflot("Токио");

List<Admin> admins = new ArrayList<>();

admins.add(new Admin("1"));

aeroflot1.setAdmins(admins);

Plane plane1 = new Plane("Маленький");

Admin firstAdmin = admins.get(0);

List<String> crew = firstAdmin.setCrew(

"Саша, Макс, Ви",

"Никита",

"Андрей",

"Ира, Виталина"

);

plane1.setCrew(crew);

LocalDateTime localDate = LocalDateTime.of(2020, Month.APRIL, 15, 10, 30);

LocalDateTime departureDate = LocalDateTime.of(2020, Month.AUGUST, 12, 22, 00);

Date destinationDate = Date.from(Instant.from(localDate.atZone(ZoneId.systemDefault())));

Date departureDateUTC = Date.from(Instant.from(departureDate.atZone(ZoneId.systemDefault())));

Flight flight1 = new Flight(

"Лондон",

"Токио",

departureDateUTC,

destinationDate,

plane1

);

aeroflot1.addFlight(flight1);

aeroflot2.addFlight(flight1);

System.out.println(aeroflot1);

System.out.println(aeroflot2);

System.out.println(aeroflot3);

LocalDateTime localDate2 = LocalDateTime.of(2020, Month.DECEMBER, 13, 15, 00);

Date date2 = Date.from(Instant.from(localDate2.atZone(ZoneId.systemDefault())));

changeDestination(aeroflot2, aeroflot3, date2, flight1);

System.out.println(aeroflot1);

System.out.println(aeroflot2);

System.out.println(aeroflot3);

discardFlight(aeroflot1, aeroflot3, flight1);

System.out.println(aeroflot1);

System.out.println(aeroflot2);

System.out.println(aeroflot3);

}

private static void discardFlight(Aeroflot destination, Aeroflot departure, Flight flight) {

if(flight.getDepartureTime().before(new Date())) {

System.out.println("Вы не можете отменить полет");

return;

}

if(destination.getFlights().contains(flight) && departure.getFlights().contains(flight)) {

departure.discardFlight(flight);

destination.discardFlight(flight);

}

}

private static void changeDestination(Aeroflot oldDestination,

Aeroflot newDestination,

Date newDestinationTime,

Flight flight

) {

List<Flight> oldFlights = oldDestination.getFlights();

int oldFlightIndex = oldFlights.indexOf(flight);

Flight oldFlight = oldFlights.get(oldFlightIndex);

if(oldFlight.getDestinationTime().before(new Date())) {

System.out.println("Вы не можете изменить пункт назначения");

return;

}

flight.setDestination(newDestination.getName());

flight.setDestinationTime(newDestinationTime);

oldDestination.discardFlight(flight);

newDestination.addFlight(flight);

System.out.println("Пункт назначения изменен");

}

}

class Admin {

private static final AtomicInteger count = new AtomicInteger(1);

private int id;

private String name;

public Admin(String name) {

this.id = count.incrementAndGet();

this.name = name;

}

public List<String> setCrew(String pilots, String navigator, String operator, String stewardesses) {

List<String> crew = new ArrayList<String>();

crew.add(pilots);

crew.add(navigator);

crew.add(operator);

crew.add(stewardesses);

return crew;

}

@Override

public String toString() {

return "Администратор {" +

"id=" + id +

",имя=" + name +

"}";

}

}

class Aeroflot {

private static final AtomicInteger count = new AtomicInteger(1);

private int id;

private String name;

private List<Admin> admins;

private List<Flight> flights;

public Aeroflot(String name) {

this.id = count.incrementAndGet();

this.name = name;

this.admins = new ArrayList<>();

this.flights = new ArrayList<>();

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public List<Admin> getAdmins() {

return admins;

}

public void setAdmins(List<Admin> admins) {

this.admins = admins;

}

public List<Flight> getFlights() {

return flights;

}

public void setFlights(List<Flight> flights) {

this.flights = flights;

}

public void addFlight(Flight flight) {

this.flights.add(flight);

}

public void addAdmin(Admin admin) {

this.admins.add(admin);

}

public void discardFlight(Flight flight) {

this.flights.remove(flight);

}

@Override

public String toString() {

return "Аэрофлот {" +

"\n\t id=" + id +

",\n\t имя='" + name + '\'' +

",\n\t администратор=" + admins +

",\n\t полеты=" + flights +

"\n}";

}

}

class Flight {

private String destination;

private String departure;

private Date destinationTime;

private Date departureTime;

private Plane plane;

public Flight(String destination, String departure, Date departureTime, Date destinationTime, Plane plane) {

this.destination = destination;

this.departure = departure;

this.destinationTime = destinationTime;

this.departureTime = departureTime;

this.plane = plane;

}

public String getDestination() {

return destination;

}

public void setDestination(String destination) {

this.destination = destination;

}

public String getDeparture() {

return departure;

}

public void setDeparture(String departure) {

this.departure = departure;

}

public Date getDestinationTime() {

return destinationTime;

}

public void setDestinationTime(Date destinationTime) {

this.destinationTime = destinationTime;

}

public Date getDepartureTime() {

return departureTime;

}

public void setDepartureTime(Date departureTime) {

this.departureTime = departureTime;

}

public Plane getPlane() {

return plane;

}

public void setPlane(Plane plane) {

this.plane = plane;

}

@Override

public String toString() {

return "\n\t\t Полеты {" +

"\n\t\t\t Пункт назначения='" + destination + '\'' +

",\n\t\t\t Вылет из='" + departure + '\'' +

",\n\t\t\t Время прибытия=" + destinationTime +

",\n\t\t\t Время вылета=" + departureTime +

",\n\t\t\t Самолет=" + plane +

"\n\t}";

}

}

class Plane {

private static final AtomicInteger count = new AtomicInteger(1);

private int id;

private String planeType;

private List<String> crew;

public Plane(String planeType) {

this.id = count.incrementAndGet();

this.planeType = planeType;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getTypeOfPlane() {

return planeType;

}

public void setTypeOfPlane(String planeType) {

this.planeType = planeType;

}

public List<String> getCrew() {

return crew;

}

public void setCrew(List<String> crew) {

this.crew = crew;

}

@Override

public String toString() {

return "\n\t\t\t\t Самолет {" +

" \n\t\t\t\t\t id=" + id +

",\n\t\t\t\t\t Тип самолета=" + planeType +

",\n\t\t\t\t\t Персонал=" + crew +

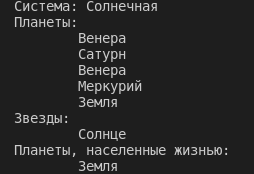
"\n\t\t\t\t}";

}

}

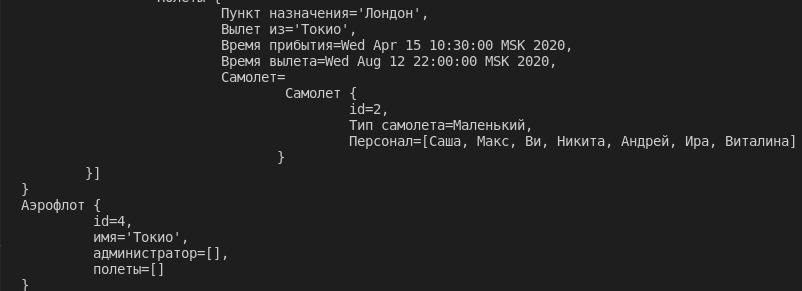
**Вывод работы:**











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