|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа,**

**обработки и интерпретации больших данных.**

**Отчет**

|  |
| --- |
| **по лабораторной работе № 3** |
|  |
| **Вариант 16** |

**Название:**

Классы. Наследование и полиморфизм

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-23М | |  |  | В.И. Тарасов |
|  | | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  | |  |  |  |  |
| Преподаватель | |  |  |  | П.В. Степанов |
|  | |  |  | (Подпись, дата) | (И.О. Фамилия) |

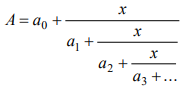
Москва, 2022

**Цель работы** –получение навыков работы с классами в Java, исследование механизмов наследования и полиморфизма.

**Ход работы**

**Задание 1:**

1. Определить класс Цепная дробь



Определить методы сложения, вычитания, умножения, деления. Вычислить значение для заданного n, x, a[n].

1. Определить класс Дробь в виде пары (m,n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения и деления дробей. Объявить массив из k дробей, ввести/вывести значения для массива дробей. Создать массив объектов и передать его в метод, который изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента массива.

Листинг 1 – Код подзадачи 1

package src.lab3\_1;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

public class ContinuedFraction {

public int n;

public int x;

public ArrayList<Integer> a;

public double value;

public ContinuedFraction(int n, int x, ArrayList<Integer> a){

this.n = n;

this.x = x;

this.a = a;

this.value = 0.0;

calc();

}

public double sum(ContinuedFraction term){

return this.value + term.value;

}

public double sub (ContinuedFraction term){

return this.value - term.value;

}

public double mul(ContinuedFraction term){

return this.value \* term.value;

}

public double div(ContinuedFraction term){

return this.value / term.value;

}

public void calc(){

double temp = 0.0;

for (int i = n-1; i >= 1; i--) {

temp = x / (temp + this.a.get(i));

}

value = temp;

}

public static void main(String[] args) {

ContinuedFraction frac = new ContinuedFraction(3, 2, new ArrayList<Integer> (Arrays.asList(1, 2, 3)));

ContinuedFraction frac2 = new ContinuedFraction(3, 2, new ArrayList<Integer> (Arrays.asList(1, 2, 3)));

System.out.println("Значение 1: ");

System.out.println(frac.value);

System.out.println("Значение 2: ");

System.out.println(frac2.value);

System.out.println("Сложение: ");

System.out.println(frac.sum(frac2));

System.out.println("Вычитание: ");

System.out.println(frac.sub(frac2));

System.out.println("Умножение: ");

System.out.println(frac.mul(frac2));

System.out.println("Деление: ");

System.out.println(frac.div(frac2));

}

}

Результаты выполнения подзадачи 1 представлены на рисунке 1.

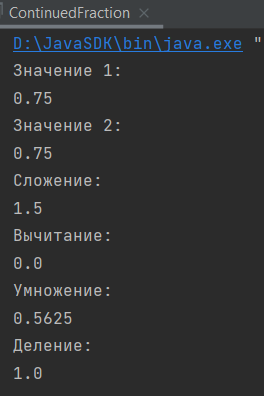


Рисунок 1 – Результат выполнения задания 1

Листинг 2 – Код класса Fraction подзадачи 2

package src.lab3\_1;

public class Fraction {

int m, n;

public Fraction() {

m = 0;

n = 1;

}

public Fraction(int m) {

this.m = m;

n = 1;

}

public Fraction(int m, int n) {

this.m = m;

this.n = n;

}

public Fraction(Fraction a) {

m = a.m;

n = a.n;

}

public Fraction Add(Fraction term) {

Fraction sum = new Fraction();

sum.n = this.n \* term.n;

sum.m = m \* term.n + term.m \* n;

return sum;

}

public Fraction Multiply(Fraction term) {

return new Fraction(

m \* term.m,

n \* term.n

);

}

public Fraction Invert() {

return new Fraction(-m, n);

}

public Fraction Subtract(Fraction term) {

Fraction inverted\_term = new Fraction(term.Invert());

return this.Add(inverted\_term);

}

public Fraction Switch() {

return new Fraction(n, m);

}

public Fraction Divide(Fraction term) {

Fraction switched\_multiplicator = term.Switch();

return this.Multiply(switched\_multiplicator);

}

public String toString() {

return m + "/" + n;

}

}

Листинг 3 – Главная программа подзадачи 2

package src.lab3\_1;

import java.util.Scanner;

public class Program {

public static void main(String[] args) {

System.out.println("Введите k:");

Scanner console = new Scanner(System.in);

int k = console.nextInt();

Fraction[] fractions = new Fraction[k];

for (int i = 0; i < k; i++) {

System.out.println("Введите m для дроби " + (i + 1) + ":");

int m = console.nextInt();

System.out.println("Введите n для дроби " + (i + 1) + ":");

int n = console.nextInt();

fractions[i] = new Fraction(m, n);

}

PrintArray(fractions);

for (int i = 0; i < k - 1; i += 2) {

fractions[i] = fractions[i].Add(fractions[i + 1]);

}

PrintArray(fractions);

console.close();

}

private static void PrintArray(Fraction[] array) {

System.out.print("[");

for (int i = 0; i < array.length; i++) {

System.out.print(array[i].toString());

if (i != array.length - 1) System.out.print(", ");

}

System.out.print("]\n");

}

}

Результаты выполнения подзадачи 2 представлены на рисунке 2.

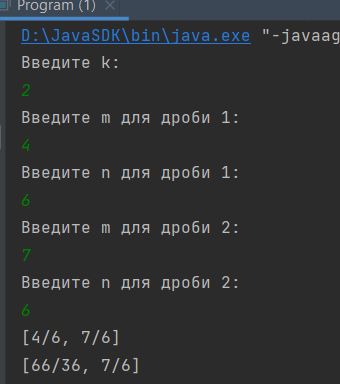


Рисунок 2 – Результат выполнения подзадачи 2

**Задание 2:**

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.

1. House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации. Создать массив объектов. Вывести: a) список квартир, имеющих заданное число комнат; b) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке; c) список квартир, имеющих площадь, превосходящую заданную.
2. Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит, Время городских и междугородных разговоров. Создать массив объектов. Вывести: a) сведения об абонентах, у которых время внутригородских разговоров превышает заданное; b) сведения об абонентах, которые пользовались междугородной связью; c) сведения об абонентах в алфавитном порядке.

Листинг 4 – Код класса House

package src.lab3\_2;

public class House {

int id;

int number;

int square;

int floor;

int room\_count;

String street, building\_type;

int lifetime;

public void setId(int id) {

this.id = id;

}

public void setBuilding\_type(String building\_type) {

this.building\_type = building\_type;

}

public void setFloor(int floor) {

this.floor = floor;

}

public void setLifetime(int lifetime) {

this.lifetime = lifetime;

}

public void setNumber(int number) {

this.number = number;

}

public void setRoom\_count(int room\_count) {

this.room\_count = room\_count;

}

public void setSquare(int square) {

this.square = square;

}

public void setStreet(String street) {

this.street = street;

}

public int getId() {

return id;

}

public double getSquare() {

return square;

}

public int getFloor() {

return floor;

}

public int getLifetime() {

return lifetime;

}

public int getNumber() {

return number;

}

public int getRoom\_count() {

return room\_count;

}

public String getBuilding\_type() {

return building\_type;

}

public String getStreet() {

return street;

}

@Override

public String toString() {

return "House{" +

"id=" + id +

", number=" + number +

", square=" + square +

", floor=" + floor +

", room\_count=" + room\_count +

", street='" + street + '\'' +

", building\_type='" + building\_type + '\'' +

", lifetime=" + lifetime +

'}';

}

}

Листинг 5 – Код класса Phone

package src.lab3\_2;

public class Phone {

int id;

String last\_name, name, patronymic, address;

long card\_number;

double debit, credit;

int intercity\_calls, intracity\_calls;

public int getId() {

return id;

}

public String getLast\_name() {

return last\_name;

}

public String getName() {

return name;

}

public String getPatronymic() {

return patronymic;

}

public String getAddress() {

return address;

}

public long getCard\_number() {

return card\_number;

}

public double getDebit() {

return debit;

}

public double getCredit() {

return credit;

}

public int getIntercity\_calls() {

return intercity\_calls;

}

public int getIntracity\_calls() {

return intracity\_calls;

}

public void setAddress(String address) {

this.address = address;

}

public void setCard\_number(long card\_number) {

this.card\_number = card\_number;

}

public void setCredit(double credit) {

this.credit = credit;

}

public void setDebit(double debit) {

this.debit = debit;

}

public void setId(int id) {

this.id = id;

}

public void setIntercity\_calls(int intercity\_calls) {

this.intercity\_calls = intercity\_calls;

}

public void setIntracity\_calls(int intracity\_calls) {

this.intracity\_calls = intracity\_calls;

}

public void setLast\_name(String last\_name) {

this.last\_name = last\_name;

}

public void setName(String name) {

this.name = name;

}

public void setPatronymic(String patronymic) {

this.patronymic = patronymic;

}

@Override

public String toString() {

return "Phone{" +

"id=" + id +

", last\_name='" + last\_name + '\'' +

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

", patronymic='" + patronymic + '\'' +

", address='" + address + '\'' +

", card\_number=" + card\_number +

", debit=" + debit +

", credit=" + credit +

", intercity\_calls=" + intercity\_calls +

", intracity\_calls=" + intracity\_calls +

'}';

}

}

Листинг 6 – Код основной программы

package src.lab3\_2;

import java.util.Arrays;

import java.util.Comparator;

public class Program {

public static void main(String[] args) {

Phone[] phones = new Phone[3];

phones[0] = new Phone();

phones[0].setAddress("Балашиха");

phones[0].setCard\_number(4000111122223333L);

phones[0].setCredit(1000);

phones[0].setDebit(500);

phones[0].setId(1);

phones[0].setIntercity\_calls(0);

phones[0].setIntracity\_calls(50);

phones[0].setLast\_name("Santalov");

phones[0].setName("Vadim");

phones[0].setPatronymic("Evgenievich");

phones[1] = new Phone();

phones[1].setAddress("Птичное");

phones[1].setCard\_number(4000111122223333L);

phones[1].setCredit(1000);

phones[1].setDebit(500);

phones[1].setId(1);

phones[1].setIntercity\_calls(10);

phones[1].setIntracity\_calls(500);

phones[1].setLast\_name("Eliseev");

phones[1].setName("Octoslav");

phones[1].setPatronymic("Mnogoslavovich");

phones[2] = new Phone();

phones[2].setAddress("Пущино");

phones[2].setCard\_number(4000111122223333L);

phones[2].setCredit(1000);

phones[2].setDebit(500);

phones[2].setId(1);

phones[2].setIntercity\_calls(10);

phones[2].setIntracity\_calls(100);

phones[2].setLast\_name("Tarasov");

phones[2].setPatronymic("Igorevich");

phones[2].setName("Viktor");

System.out.println("Пользователи, превысившие лимиты: ");

intracityOverGiven(phones, 100);

System.out.println("Пользователи, использовавшие междугородние звонки: ");

intercityCallsUsed(phones);

System.out.println("В алфавитном порядке");

alphabetOrderUsers(phones);

House[] houses = new House[3];

houses[0] = new House();

houses[0].setBuilding\_type("Офисное");

houses[0].setFloor(1);

houses[0].setNumber(10);

houses[0].setId(1);

houses[0].setLifetime(20);

houses[0].setRoom\_count(10);

houses[0].setSquare(150);

houses[0].setStreet("Пушкина");

houses[1] = new House();

houses[1].setBuilding\_type("Жилое");

houses[1].setFloor(7);

houses[1].setNumber(11);

houses[1].setId(2);

houses[1].setLifetime(15);

houses[1].setRoom\_count(5);

houses[1].setSquare(100);

houses[1].setStreet("Колотушкина");

houses[2] = new House();

houses[2].setBuilding\_type("Жилое");

houses[2].setFloor(3);

houses[2].setNumber(15);

houses[2].setId(3);

houses[2].setLifetime(15);

houses[2].setRoom\_count(5);

houses[2].setSquare(20);

houses[2].setStreet("Улицына");

System.out.println("Имеющие заданное число комнат: ");

roomCountGiven(houses,5);

System.out.println("Имеющие заданное число комнат и в диапазоне этажей: ");

roomCountAndFloor(houses, 5, 4, 8);

System.out.println("Превосходящие площадь: ");

squareExceedsGiven(houses, 90);

}

public static void intracityOverGiven(Phone[] phones, int limit) {

Arrays.stream(phones).filter(c -> c.getIntracity\_calls() > limit).forEach(c -> System.out.println(c.toString()));

}

public static void intercityCallsUsed(Phone[] phones){

Arrays.stream(phones).filter(c -> c.getIntercity\_calls() > 0).forEach(c -> System.out.println(c.toString()));

}

public static void alphabetOrderUsers(Phone[] phones){

Arrays.sort(phones, Comparator.comparing(a -> a.last\_name));

Arrays.stream(phones).forEach(c -> System.out.println(c.toString()));

}

public static void roomCountGiven(House[] houses, int count){

Arrays.stream(houses).filter(c -> c.getRoom\_count() == count).forEach(c -> System.out.println(c.toString()));

}

public static void roomCountAndFloor(House[] houses, int count, int bottom, int top){

Arrays.stream(houses)

.filter(c -> c.getRoom\_count() == count)

.filter(c -> c.getFloor() >= bottom)

.filter(c -> c.getFloor() <= top)

.forEach(c -> System.out.println(c.toString()));

}

public static void squareExceedsGiven(House[] houses, int square){

Arrays.stream(houses).filter(c -> c.getSquare() > square).forEach(c -> System.out.println(c.toString()));

}

}

Результаты выполнения представлены на рисунке 3.

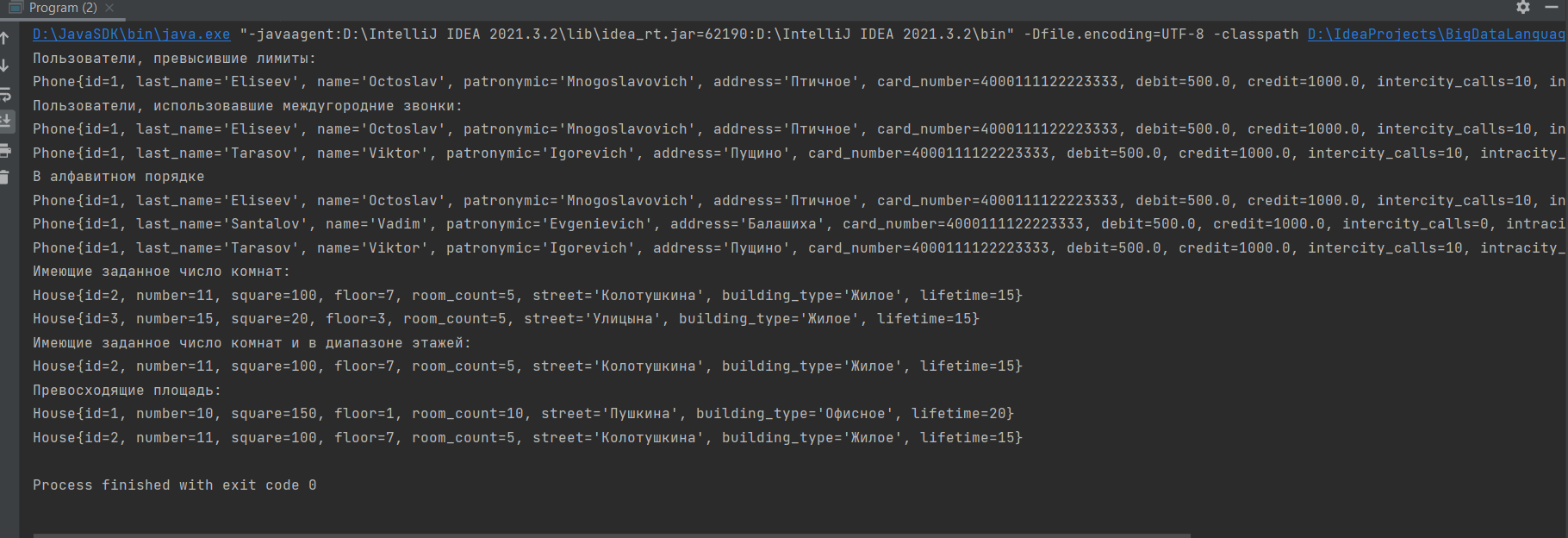


Рисунок 3 – Результат выполнения задания 2

**Задание 3:**

Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString().

1. Создать объект класса Дом, используя классы Окно, Дверь. Методы: закрыть на ключ, вывести на консоль количество окон, дверей.
2. Создать объект класса Роза, используя классы Лепесток, Бутон. Методы: расцвести, завять, вывести на консоль цвет бутона.

Листинг 7 – Код класса Bud

package src.lab3\_3;

import java.util.Objects;

public class Bud {

public String color;

public Petal[] petals;

public Bud(String color) {

this.color = color;

petals = new Petal[5];

for(int i = 0; i < 5; i++){

petals[i] = new Petal();

}

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Bud bud = (Bud) o;

return color.equals(bud.color);

}

@Override

public int hashCode() {

return Objects.hash(color);

}

@Override

public String toString() {

return "Bud{" +

"color='" + color + '\'' +

'}';

}

}

Листинг 8 – Код класса Petal

package src.lab3\_3;

import java.util.Objects;

public class Petal {

public String shape;

public Petal(){

this.shape = "Basic";

}

public Petal(String shape){

this.shape = shape;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Petal petal = (Petal) o;

return shape.equals(petal.shape);

}

@Override

public int hashCode() {

return Objects.hash(shape);

}

@Override

public String toString() {

return "Petal{" +

"shape='" + shape + '\'' +

'}';

}

}

Листинг 9 – Код класса FlowerState

package src.lab3\_3;

public enum FlowerState {

BLOOM, WITHER, GROWING

}

Листинг 10 – Код класса Rose

package src.lab3\_3;

import java.util.Objects;

public class Rose {

Bud bud;

FlowerState state = FlowerState.GROWING;

public Rose(String color){

bud = new Bud(color);

}

public void bloom(){

this.state = FlowerState.BLOOM;

}

public void wither(){

state = FlowerState.WITHER;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Rose rose = (Rose) o;

return bud.equals(rose.bud) && state == rose.state;

}

@Override

public int hashCode() {

return Objects.hash(bud, state);

}

@Override

public String toString() {

return "Rose{" +

"bud=" + bud +

", state=" + state +

'}';

}

public void printColor(){

System.out.println(bud.color);

}

}

Листинг 11 – Код класса Door

package src.lab3\_3;

import java.util.Objects;

public class Door {

DoorState state = DoorState.OPEN;

public void useKey(){

if (state == DoorState.OPEN){

state = DoorState.CLOSE;

}

else {

state = DoorState.OPEN;

}

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Door door = (Door) o;

return state == door.state;

}

@Override

public int hashCode() {

return Objects.hash(state);

}

}

Листинг 12 – Код класса DoorState

package src.lab3\_3;

public enum DoorState {

OPEN, CLOSE

}

Листинг 13 – Код класса House

package src.lab3\_3;

import java.util.Arrays;

public class House {

Door[] doors = new Door[2];

Window[] windows = new Window[3];

public House(){

for (int i = 0; i < 2; i++){

doors[i] = new Door();

}

for (int i = 0; i < 3; i++){

windows[i] = new Window(i);

}

}

public void getDoorsCount(){

System.out.println("Количество дверей: " + doors.length);

}

public void getWindowsCount(){

System.out.println("Количество окон: " + windows.length);

}

public void closeDoors(){

System.out.println("Закрываем все двери...");

Arrays.stream(doors).filter(c -> c.state == DoorState.OPEN).forEach(Door::useKey);

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

House house = (House) o;

return Arrays.equals(doors, house.doors) && Arrays.equals(windows, house.windows);

}

@Override

public int hashCode() {

int result = Arrays.hashCode(doors);

result = 31 \* result + Arrays.hashCode(windows);

return result;

}

}

Листинг 14 – Код класса Window

package src.lab3\_3;

import java.util.Objects;

public class Window {

int glassesCount;

public Window(int glassesCount){

this.glassesCount = glassesCount;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Window window = (Window) o;

return glassesCount == window.glassesCount;

}

@Override

public int hashCode() {

return Objects.hash(glassesCount);

}

}

Листинг 15 – Код основной программы

package src.lab3\_3;

public class Program {

public static void main(String[] args) {

House house = new House();

house.getDoorsCount();

house.getWindowsCount();

System.out.println("Состояние дверей: ");

for (int i = 0; i < house.doors.length; i++){

System.out.println(house.doors[i].state);

}

house.closeDoors();

System.out.println("Состояние дверей: ");

for (int i = 0; i < house.doors.length; i++){

System.out.println(house.doors[i].state);

}

}

}

Результаты выполнения задания 3 представлены на рисунке 4.

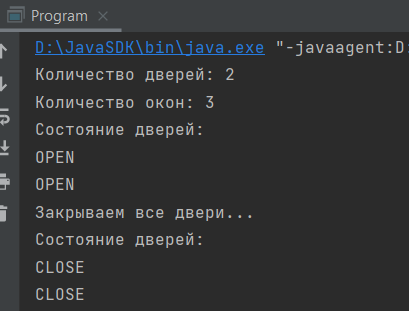
****

Рисунок 4 – Результаты выполнения задания 3

**Задание 4:**

Построить модель программной системы.

1. Система Конструкторское бюро. Заказчик представляет Техническое Задание (ТЗ) на проектирование многоэтажного Дома. Конструктор регистрирует ТЗ, определяет стоимость проектирования и строительства, выставляет Заказчику Счет за проектирование и создает Бригаду Конструкторов для выполнения Проекта.
2. Система Телефонная станция. Абонент оплачивает Счет за разговоры и Услуги, может попросить Администратора сменить номер и отказаться от услуг. Администратор изменяет номер, Услуги и временно отключает Абонента за неуплату.

Листинг 16 – Код класса Bill

package src.lab3\_4;

import java.util.HashMap;

public class Bill {

public HashMap<Integer, Service> activeServices = new HashMap<>();

public int sum;

public void addService(int id, Service service){

activeServices.put(id, service);

sum += service.price;

}

public void removeService(int id){

sum -= activeServices.get(id).price;

activeServices.remove(id);

}

public int getSum() {

return sum;

}

}

Листинг 17 – Код класса Client

package src.lab3\_4;

public class Client {

public boolean status = true;

public int number;

public int balance = 0;

public Bill bill = new Bill();

public Client(int number){

this.number = number;

}

public void switchStatus(){

status = !status;

}

public void setNumber(int number){

this.number = number;

}

public void pay(int sum){

balance += sum;

}

public int getBalance() {

return balance;

}

public boolean checkBalance(){

if (balance >= bill.sum){

balance -= bill.sum;

status = true;

}

else {

status = false;

}

return status;

}

}

Листинг 18 – Код класса Service

package src.lab3\_4;

public class Service {

public int price;

public String name;

public Service(int price, String name){

this.price = price;

this.name = name;

}

}

Листинг 19 – Код класса Bill

package src.lab3\_4\_2;

public class Bill {

int building\_cost;

int designing\_cost;

public Bill(int building\_cost, int designing\_cost){

this.building\_cost = building\_cost;

this.designing\_cost = designing\_cost;

}

}

Листинг 20 – Код класса Brigade

package src.lab3\_4\_2;

public class Brigade {

int persons\_number;

public Brigade(int persons\_number){

this.persons\_number = persons\_number;

}

}

Листинг 21 – Код класса Client

package src.lab3\_4\_2;

public class Client {

Specification spec;

int balance;

Bill bill;

public Client(Specification spec, int balance){

this.spec = spec;

this.balance = balance;

}

public void setBill(Bill bill) {

this.bill = bill;

}

}

Листинг 22 – Код класса House

package src.lab3\_4\_2;

public class House {

public int floor\_count;

public MaterialType material;

public House(int floor\_count, MaterialType material){

this.floor\_count = floor\_count;

this.material = material;

}

}

Листинг 23 – Код класса MaterialType

package src.lab3\_4\_2;

public enum MaterialType {

BASIC(100),

ADVANCED(1000),

EXCQUISITE(10000);

private int cost;

MaterialType(int cost){

this.cost = cost;

}

public int getCost() {

return cost;

}

}

Листинг 24 – Код класса Project

package src.lab3\_4\_2;

public class Project {

Specification spec;

Brigade brigade;

public Project(Specification spec, Brigade brigade){

this.spec = spec;

this.brigade = brigade;

}

}

Листинг 25 – Код класса Specification

package src.lab3\_4\_2;

public class Specification {

House house;

public Specification(int floor\_count, MaterialType material){

house = new House(floor\_count, material);

}

}

Листинг 26 – Код главной программы подзадачи 1

package src.lab3\_4\_2;

public class Program {

public static void main(String[] args) {

Specification specification = new Specification(5, MaterialType.ADVANCED);

Client client = new Client(specification, 500000);

createBill(client);

Project project = createProject(specification);

}

public static void createBill(Client client){

int designing\_cost = client.spec.house.floor\_count \* client.spec.house.material.getCost();

int building\_cost = designing\_cost \* 2;

Bill bill = new Bill(building\_cost, designing\_cost);

client.setBill(bill);

}

public static Project createProject(Specification spec){

int persons\_number = spec.house.floor\_count;

Brigade brigade = new Brigade(persons\_number);

return new Project(spec, brigade);

}

}

Листинг 27 – Код главной программы подзадачи 2

package src.lab3\_4;

import java.util.HashMap;

public class Program {

public static void main(String[] args) {

HashMap<Integer, Service> services = new HashMap<>();

services.put(0, new Service(10, "Кабельное ТВ"));

services.put(1, new Service(20, "Белый IP"));

services.put(2, new Service(30, "Выделенный канал"));

Client client1 = new Client(1234);

Client client2 = new Client(5678);

client1.bill.addService(0, services.get(0));

client1.bill.addService(1, services.get(1));

client2.bill.addService(1, services.get(1));

client2.bill.addService(2, services.get(2));

client1.pay(20);

client2.pay(40);

removeClientService(client1, 1);

changeNumber(client2, 91011);

System.out.println(client1.checkBalance());

System.out.println(client2.checkBalance());

}

public static void changeNumber(Client client, int number){

client.setNumber(number);

}

public static void removeClientService(Client client, int id){

client.bill.removeService(id);

}

}

В результате выполнения задания 4 были разработаны классы, взаимодействующие между собой в рамках построенной программной системы.

**Местоположение проектных файлов** –файлы проекта расположены в репозитории GitHub по следующим адресам:

<https://github.com/Fregatty/BigDataLanguages/tree/main/src/lab3_1>

<https://github.com/Fregatty/BigDataLanguages/tree/main/src/lab3_2>

<https://github.com/Fregatty/BigDataLanguages/tree/main/src/lab3_3>

<https://github.com/Fregatty/BigDataLanguages/tree/main/src/lab3_4>

<https://github.com/Fregatty/BigDataLanguages/tree/main/src/lab3_4_2>

**Вывод** – в результате выполнения лабораторной работы были получены навыки работы с классами Java. Были изучены и применены на практике механизмы полиморфизма и наследования.