|  |  |
| --- | --- |
| 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, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ 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**

**Название:** Классы и объекты

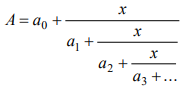
**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-12М |  |  | В.А. Трофимов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Задания:**

1.6. Определить класс Цепная дробь



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

1.7. 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.

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.

2.6. 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.

2.7. 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.

Код для решения задания 1.6:

|  |
| --- |
| package bdjava.lab3.var1;  import java.io.Serializable; import java.util.Scanner; import static java.lang.System.in; import static java.lang.System.out;  public class ContinuedFraction implements Serializable {  private final short N;  private final float[] A;  private float x;  private double bufRes;  transient private Scanner reader;  public ContinuedFraction(short n, float x){  this(n);  this.x = x;  }  public ContinuedFraction(short n){  this.A = new float[n];  this.N = n;  inputConst(n);  }  public void changeX(float x){  this.x = x;  }  private void inputConst(short n){  reader = new Scanner(in);  out.println("Введите " + n +" констант(ы), начиная с нижнего уровня цепной дроби");  for(short i=0; i<n; i++) {  A[i] = reader.nextFloat();  }  }  public double calculate(boolean detail) {  if (this.bufRes != 0)  return bufRes;  else {  double buf = this.A[0] + this.x;  if(detail)  out.println(this.A[0] + " + " + this.x + " / 1 = " + buf);  for(short i = 1; i<this.N; i++) {  if(detail)  out.print(this.A[i] + " + " + this.x + " / " + buf + " = ");  buf = this.A[i] + this.x / buf;  if(detail)  out.println(buf);  }  this.bufRes = buf;  return buf;  }  }  public String toString(){  String buf = "Созданная цепная дробь:\n";  for(short i = 1; i<=this.N; i++)  buf = buf + this.A[this.A.length-i] + " + " + this.x + "/(";  buf = buf + "1 ...)";  return buf;  } } |

Код для решения задания 1.6:

|  |
| --- |
| package bdjava.lab3.var1;   import java.io.Serializable;  public class Fraction implements Serializable {  private int m;  private int n;   public Fraction(int m){  this.m = m;  this.n = 1;  }  public Fraction(int m, int n){  this.m = m;  this.n = n;  }  Fraction(){  this(1);  }  public static Fraction addition(Fraction first, Fraction second){  Fraction bufF = new Fraction();  int bufI1, bufI2;  Fraction.fracReduction(first);  Fraction.fracReduction(second);  bufF.n = first.n \* second.n;  bufI1 = first.m \* second.n;  bufI2 = second.m \* first.n;  bufF.m = bufI1 + bufI2;  Fraction.fracReduction(bufF);  return bufF;  }  public static Fraction subtraction(Fraction first, Fraction second){  Fraction bufF = new Fraction();  int bufI1, bufI2;  Fraction.fracReduction(first);  Fraction.fracReduction(second);  bufF.n = first.n \* second.n;  bufI1 = first.m \* second.n;  bufI2 = second.m \* first.n;  bufF.m = bufI1 - bufI2;  Fraction.fracReduction(bufF);  return bufF;  }  public static Fraction multiplication(Fraction first, Fraction second){  Fraction bufF = new Fraction();  Fraction.fracReduction(first);  Fraction.fracReduction(second);  bufF.n = first.n \* second.n;  bufF.m = first.m \* second.m;  Fraction.fracReduction(bufF);  return bufF;  }  public static Fraction division(Fraction first, Fraction second){  Fraction bufF = new Fraction(), divisor = new Fraction(second.n, second.m);  Fraction.fracReduction(first);  Fraction.fracReduction(divisor);  bufF.n = first.n \* divisor.n;  bufF.m = first.m \* divisor.m;  Fraction.fracReduction(bufF);  return bufF;  }  public static void neighborAddition(Fraction[] fracs){  for (short i=0; i<fracs.length-1; i++)  if(i % 2 == 0)  fracs[i] = Fraction.addition(fracs[i], fracs[i+1]);  }  private static void fracReduction(Fraction frac){  int first = frac.m;  int second = frac.n;  int nod;  if (second == 0)  throw new ArithmeticException("Деление на ноль в операциях с обыкновенными дробями!");  else if (first != 0)  {  while (first != second)  if (first > second)  first -= second;  else  second -= first;  nod = first;  frac.m /= nod;  frac.n /= nod;  }  }  public String toString(){  return this.m + "/" + this.n;  } } |

Код для решения задания 1.6:

|  |
| --- |
| package bdjava.lab3.var1; import static java.lang.System.\*; import java.lang.Math; /\* 6. Определить класс Цепная дробь Определить методы сложения, вычитания, умножения, деления. Вычислить значение для заданного n, x, a[n]. 7. 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.  \*/ public class Program {  public static void main(String[] args) {  out.println("Часть 1:");  ContinuedFraction frac1 = new ContinuedFraction((short) 5,3.6f);  out.println("Результат: " + frac1.calculate(true));  out.println(frac1);  out.println();  out.println("Часть 2:");  Fraction frac2 = new Fraction(232,8);  Fraction frac3 = new Fraction(32,7);  Fraction sum = Fraction.addition(frac2, frac3);  Fraction sub = Fraction.subtraction(frac2, frac3);  Fraction mul = Fraction.multiplication(frac2, frac3);  Fraction div = Fraction.division(frac2, frac3);  out.println(frac2 + " + " + frac3 + " = " + sum);  out.println(frac2 + " - " + frac3 + " = " + sub);  out.println(frac2 + " \* " + frac3 + " = " + mul);  out.println(frac2 + " / " + frac3 + " = " + div);  Fraction[] fracs = new Fraction[15];  out.println("Сгенерированный набор дробей:");  for(short i=0; i<15; i++) {  fracs[i] = new Fraction((int) (Math.random() \* 20), (int) (Math.random() \* 20));  out.println(fracs[i]);  }  Fraction.neighborAddition(fracs);  out.println("Набор после преобразования:");  for(short i=0; i<15; i++)  out.println(fracs[i]);  } } |

Код для решения задания 1.7:

|  |
| --- |
| package bdjava.lab3.var2;  import java.io.Serializable; import java.util.ArrayList; import java.util.HashMap; import bdjava.lab5.Lab5Exceptions;  public class House implements Serializable {  private int ID;  private short num;  private float square;  private short floor;  private short numberOfRooms;  private String street;  private String buildingType;  private short serviceLifeInYears;  private static int numberOfApartments;  public House(short num, float square, short floor, short numberOfRooms, String street,  String buildingType, short serviceLifeInYears) throws Lab5Exceptions.NegativeNumberException, Lab5Exceptions.EmptyLineException {  if (num <= 0)  throw new Lab5Exceptions.NegativeNumberException("House(...num...)", Integer.toString(this.num));  else this.num = num;  if (square <= 0)  throw new Lab5Exceptions.NegativeNumberException("House(...square...)", Float.toString(this.square));  else this.square = square;  if (floor <= 0)  throw new Lab5Exceptions.NegativeNumberException("House(...floor...)", Integer.toString(this.floor));  else this.floor = floor;  if (numberOfRooms <= 0)  throw new Lab5Exceptions.NegativeNumberException("House(...numberOfRooms...)", Integer.toString(this.numberOfRooms));  else this.numberOfRooms = numberOfRooms;  if (street.isEmpty())  throw new Lab5Exceptions.EmptyLineException("House(...street...)");  else this.street = street;  if (buildingType.isEmpty())  throw new Lab5Exceptions.EmptyLineException("House(...buildingType...)");  else this.buildingType = buildingType;  if (serviceLifeInYears <= 0)  throw new Lab5Exceptions.NegativeNumberException("House(...serviceLifeInYears...)", Integer.toString(this.numberOfRooms));  else this.serviceLifeInYears = serviceLifeInYears;  this.ID = House.numberOfApartments;  House.numberOfApartments++;  }   public HashMap<String, String> getHouse(){  HashMap<String, String> buf = new HashMap<String, String>();  buf.put("num", String.valueOf(this.num));  buf.put("square", String.valueOf(this.square));  buf.put("floor", String.valueOf(this.floor));  buf.put("numberOfRooms", String.valueOf(this.numberOfRooms));  buf.put("street", this.street);  buf.put("buildingType", this.buildingType);  return buf;  }  public void setHouse(HashMap<String, String> set){  String buf = set.get("num");  if(buf != null)  this.num = Short.parseShort(buf);  buf = set.get("square");  if(buf != null)  this.square = Short.parseShort(buf);  buf = set.get("floor");  if(buf != null)  this.floor = Short.parseShort(buf);  buf = set.get("numberOfRooms");  if(buf != null)  this.numberOfRooms = Short.parseShort(buf);  buf = set.get("street");  if(buf != null)  this.street = buf;  buf = set.get("buildingType");  if(buf != null)  this.buildingType = buf;  }  @Override  public String toString() {  return "ID: " + this.ID + " NUM: " + this.num + " SQUARE: " + this.square + " FLOOR: " + this.floor +  " NUMBER\_OF\_ROOMS: " + this.numberOfRooms + " STREET: " + this.street + " BUILDING\_TYPE: " +  this.buildingType;  }   //Число комнат, минимальная шлощадь, минимальный этаж, максимальный этаж  public static ArrayList<House> search(ArrayList<House> apartments, Short numberOfRooms, Float minSquare, Short minLvl, Short maxLvl){  ArrayList<House> buf = new ArrayList<>();  for(House ap : apartments){  boolean hit = true;  if(numberOfRooms != null)  if(ap.numberOfRooms != numberOfRooms)  hit = false;  if(minSquare != null)  if(ap.square < minSquare)  hit = false;  if(minLvl != null)  if(ap.floor < minLvl)  hit = false;  if(maxLvl != null)  if(ap.floor > maxLvl)  hit = false;  if(!hit)  continue;  buf.add(ap);  }  return buf;  } } |

Код для решения задания 1.7:

|  |
| --- |
| package bdjava.lab3.var2; import java.io.Serializable; import bdjava.lab5.Lab5Exceptions;  import java.util.ArrayList; import java.util.Collections; import java.util.HashMap;  public class Phone implements Comparable<Phone>, Serializable{  private int ID;  private String name;  private String surname;  private String patronymic;  private String address;  private String creditCardNumber;  private int debit;  private int credit;  private short cityTalkTime;  private short longDistanceCallTime;  private static int numberOfSubscriber;  public Phone(String name, String surname, String patronymic, String address, String creditCardNumber,  int debit, int credit, short cityTalkTime, short longDistanceCallTime)  throws Lab5Exceptions.NegativeNumberException, Lab5Exceptions.EmptyLineException{  if (name.isEmpty())  throw new Lab5Exceptions.EmptyLineException("Phone(...name...)");  else this.name = name;  if (surname.isEmpty())  throw new Lab5Exceptions.EmptyLineException("Phone(...surname...)");  else this.surname = surname;  if (patronymic.isEmpty())  throw new Lab5Exceptions.EmptyLineException("Phone(...patronymic...)");  else this.patronymic = patronymic;  if (address.isEmpty())  throw new Lab5Exceptions.EmptyLineException("Phone(...address...)");  else this.address = address;  this.creditCardNumber = creditCardNumber;  this.debit = debit;  this.credit = credit;  this.cityTalkTime = cityTalkTime;  this.longDistanceCallTime = longDistanceCallTime;  this.ID = Phone.numberOfSubscriber;  Phone.numberOfSubscriber++;  }   public HashMap<String, String> getPhone(){  HashMap<String, String> buf = new HashMap<String, String>();  buf.put("name", this.name);  buf.put("surname", this.surname);  buf.put("patronymic", this.patronymic);  buf.put("address", this.address);  buf.put("creditCardNumber", this.creditCardNumber);  buf.put("debit", String.valueOf(this.debit));  buf.put("credit", String.valueOf(this.credit));  buf.put("cityTalkTime", String.valueOf(this.cityTalkTime));  buf.put("longDistanceCallTime", String.valueOf(this.longDistanceCallTime));  return buf;  }  public void setPhone(HashMap<String, String> set){  String buf = set.get("name");  if(buf != null)  this.name = buf;  buf = set.get("surname");  if(buf != null)  this.surname = buf;  buf = set.get("patronymic");  if(buf != null)  this.patronymic = buf;  buf = set.get("address");  if(buf != null)  this.address = buf;  buf = set.get("creditCardNumber");  if(buf != null)  this.creditCardNumber = buf;  buf = set.get("debit");  if(buf != null)  this.debit = Integer.parseInt(buf);  buf = set.get("credit");  if(buf != null)  this.credit = Integer.parseInt(buf);  buf = set.get("cityTalkTime");  if(buf != null)  this.cityTalkTime = Short.parseShort(buf);  buf = set.get("longDistanceCallTime");  if(buf != null)  this.longDistanceCallTime = Short.parseShort(buf);  }   @Override  public int compareTo(Phone p) {  int buf = this.name.compareToIgnoreCase(p.name);  if(buf == 0)  buf = this.surname.compareToIgnoreCase(p.surname);  if(buf == 0)  buf = this.patronymic.compareToIgnoreCase(p.patronymic);  return buf;  }   @Override  public String toString() {  return "ID: " + this.ID + " name: " + this.name + " " + this.surname + " " + this.patronymic +  " address: " + this.address + " creditCardNumber: " + this.creditCardNumber + " debit: " +  this.debit + " credit: " + this.credit + " cityTalkTime: " + this.cityTalkTime +  " longDistanceCallTime: " + this.longDistanceCallTime;  }   //Минимальное время внутригородских разговоров, кто использовал междугородную связь, вывод в алфовитном порядке  public static ArrayList<Phone> search(ArrayList<Phone> subscribers, Short minTimeForCityCalls, Boolean isUsedLongDistanceCall,  Boolean isSort){  ArrayList<Phone> buf = new ArrayList<Phone>();  for(Phone sub : subscribers){  boolean hit = true;  if(minTimeForCityCalls != null)  if(sub.cityTalkTime < minTimeForCityCalls)  hit = false;  if(isUsedLongDistanceCall != null)  if(sub.longDistanceCallTime == 0 && isUsedLongDistanceCall)  hit = false;   if(!hit)  continue;  buf.add(sub);  }  if(isSort != null)  if(isSort)  Collections.sort(buf);  return buf;  } } |

Код для решения задания 1.7:

|  |
| --- |
| package bdjava.lab3.var2; import org.jetbrains.annotations.NotNull;  import java.lang.Short; import java.lang.Float; import static java.lang.System.\*; import java.util.ArrayList; import java.util.Collections; import java.util.HashMap; import java.lang.Math;  /\* 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. 6. 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. 7. 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.  \*/ public class Program {  public static void main(String[] args) {  /\* out.println("Часть 1:");  ArrayList<House> buf;  ArrayList<House> l = new ArrayList<>();  for(short i=0; i<20; i++)  l.add(new House((short) (Math.random() \* 100 + 1), (float) (Math.random() \* 100 + 1),  (short) (Math.random() \* 40 + 1), (short) (Math.random() \* 10 + 1), "Вешняки",  "Жилое строение", (short) (Math.random() \* 100 + 1)));  out.println("Сгенерированные квартиры:");  for(House ap : l)  out.println(ap);  out.println();  out.println("Поиск по номеру квартиры (3):");  buf = House.search(l, (short)3, null, null, null);  for(House ap : buf)  out.println(ap);  out.println();  out.println("Квартиры, имеющие заданное число комнат (3) и расположенных на этаже, " +  "который находится в заданном промежутке (2,25):");  buf = House.search(l, (short)3, null, (short)2, (short)25);  for(House ap : buf)  out.println(ap);  out.println();  out.println("Квартиры, имеющие площадь, превосходящую 20 кв.м:");  buf = House.search(l, null, 20f, null, null);  for(House ap : buf)  out.println(ap);  out.println();\*/ // out.println("Часть 2:"); // ArrayList<String> names = new ArrayList<String>(); // names.add("Дима"); // names.add("Вова"); // names.add("Ваня"); // names.add("Лёня"); // names.add("Гоша"); // ArrayList<String> surname = new ArrayList<String>(); // surname.add("Иванов"); // surname.add("Борисов"); // surname.add("Вадисов"); // surname.add("Петросян"); // surname.add("Степанов"); // ArrayList<Phone> l2 = new ArrayList<>(); // for(short i=0; i<10; i++) // l2.add(new Phone(names.get((int)(Math.random()\*5)), surname.get((int)(Math.random()\*5)), "Александрович", // "Старый Гай 28 кв 5", "232354353", 21312, 6564, (short) (Math.random()\*300), // (short) (Math.random()\*2))); // out.println("Сгенерированные абоненты:"); // for(Phone ph : l2) // out.println(ph); // out.println(); // out.println("Сведения об абонентах, у которых время внутригородских разговоров превышает заданное (150):"); // ArrayList<Phone> buf2 = Phone.search(l2, (short)150, null, null); // for(Phone ph : buf2) // out.println(ph); // out.println(); // out.println("Сведения об абонентах, которые пользовались междугородной связью:"); // buf2 = Phone.search(l2, null, true, null); // for(Phone ph : buf2) // out.println(ph); // out.println(); // out.println("Сведения об абонентах в алфавитном порядке:"); // buf2 = Phone.search(l2, null, null, true); // for(Phone ph : buf2) // out.println(ph);  } } |

Код для решения задания 2.6:

|  |
| --- |
| package bdjava.lab3.var3; import static java.lang.System.\*; import java.util.ArrayList;  /\* Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString(). 5. Создать объект класса Дом, используя классы Окно, Дверь. Методы: закрыть на ключ, вывести на консоль количество окон, дверей. 6. Создать объект класса Роза, используя классы Лепесток, Бутон. Методы: расцвести, завять, вывести на консоль цвет бутона.  \*/ public class Part1 {  public static void main(String[] args) {  House h1 = new House((short) 3,(short) 4);  House h2 = new House((short) 3,(short) 4);  House h3 = new House((short) 5,(short) 4);  out.println("Сравнение идентичных квартир: " +h1.equals(h2));  out.println("Сравнение неидентичных квартир: " +h1.equals(h3));  out.println("Информация по дому h1:");  out.println(h1);  h1.accessChangeDoor(1);  h1.accessChangeWindow(2);  out.println("Информация по дому h1 после открытия окна и двери:");  out.println(h1);  out.println("Хеши идентичных домов: " + h1.hashCode() + " И " + h2.hashCode());   } }  class House{  private ArrayList<Window> windows;  private ArrayList<Door> doors;  private final int ID;  private static int numOfHouses;  House(){  this.windows = new ArrayList<Window>();  this.windows.add(new Window());  this.doors = new ArrayList<Door>();  this.doors.add(new Door());  this.ID = House.numOfHouses;  House.numOfHouses++;  }  House(short numOfWindows, short numOfDoors){  this.windows = new ArrayList<Window>(numOfWindows);  this.doors = new ArrayList<Door>(numOfDoors);  for (short i=0; i<numOfWindows; i++)  this.windows.add(new Window());  for (short i=0; i<numOfWindows; i++)  this.doors.add(new Door());  this.ID = House.numOfHouses;  House.numOfHouses++;  }  public void accessChangeWindow(int id){  this.windows.get(id).accessChange();  }  public void accessChangeDoor(int id){  this.doors.get(id).accessChange();  }  public int getNumberOfWindows(){  return this.windows.size();  }  public int getNumberOfDoor(){  return this.doors.size();  }  @Override  public boolean equals(Object obj) {  if (this == obj) return true;  if (obj == null || this.getClass() != obj.getClass()) return false;  House h = (House) obj;  if (this.getNumberOfDoor() != h.getNumberOfDoor()) return false;  if (this.getNumberOfWindows() != h.getNumberOfWindows()) return false;  return true;  }  @Override  public int hashCode() {  return 31 \* (this.getNumberOfDoor() + this.getNumberOfWindows());  }  @Override  public String toString() {  StringBuilder buf = new StringBuilder("Дом номер: " + this.ID + "\n");  buf.append("Двери:\n");  for(Door d : this.doors)  buf.append(d + "\n");  buf.append("Окна:\n");  for(Window w : this.windows)  buf.append(w + "\n");  return buf.toString();  } } class Window{  private final int ID;  private boolean isOpen;  private static int numOfWindows;  Window(){  this.ID = Window.numOfWindows;  Window.numOfWindows++;  }  public void accessChange(){  this.isOpen = !this.isOpen;  }  public boolean getIsOpen(){  return isOpen;  }  @Override  public boolean equals(Object obj) {  if (this == obj) return true;  if (obj == null || this.getClass() != obj.getClass()) return false;  Window w = (Window) obj;  if (this.ID != w.ID) return false;  return true;  }  @Override  public int hashCode() {  return this.ID\*31;  }  @Override  public String toString() {  return "Окно номер " + this.ID + (this.isOpen ? "\nОкно открыто\n" : "\nОкно закрыто\n");  } } class Door{  private final int ID;  private boolean isOpen;  private static int numOfDoors;  Door(){  this.ID = Door.numOfDoors;  Door.numOfDoors++;  }  public boolean getIsOpen(){  return isOpen;  }  public void accessChange(){  this.isOpen = !this.isOpen;  }  @Override  public boolean equals(Object obj) {  if (this == obj) return true;  if (obj == null || this.getClass() != obj.getClass()) return false;  Door d = (Door) obj;  if (this.ID != d.ID) return false;  return true;  }  @Override  public int hashCode() {  return this.ID\*31;  }  @Override  public String toString() {  return "Древь номер " + this.ID + (this.isOpen ? "\nДверь открыта\n" : "\nДверь закрыта\n");  } } |

Код для решения задания 2.7:

|  |
| --- |
| package bdjava.lab3.var3;  import java.util.ArrayList;  import static java.lang.System.out;  public class Part2 {  public static void main(String[] args) {  Rose r1 = new Rose(5);  Rose r2 = new Rose(5);  Rose r3 = new Rose(2);  out.println("Сравнение идентичных роз: " + r1.equals(r2));  out.println("Сравнение неидентичных роз: " + r1.equals(r3));  out.println("Информация по розе r1:");  out.println(r1);  out.println("Хеши идентичных роз: " + r1.hashCode() + " И " + r2.hashCode());  r1.wither();  out.println("Информация по розе r1 после того, как она завяла:");  out.println(r1);  out.println("Хеши ранее идентичных роз, после того как первая роза завяла: " + r1.hashCode() + " И " + r2.hashCode());  out.println();  r1.getBudColor(true);  } }  class Rose {  private final int ID;  private Bud bud;  private static int numOfRoses;  Rose() {  this.ID = Rose.numOfRoses;  Rose.numOfRoses++;  bud = new Bud();  }  Rose (int numOfPetals) {  this.ID = Rose.numOfRoses;  Rose.numOfRoses++;  bud = new Bud(numOfPetals);  }  public void bloom () {  this.bud.bloom();  }  public void wither () {  this.bud.wither();  }  public String getBudColor (boolean toConsole) {  if (toConsole)  out.println(this.bud.getColor());  return this.bud.getColor();  }  @Override  public String toString() {  return "Роза номер " + this.ID + "\n" + this.bud.toString();  }  public Bud getBud () { return this.bud; }  @Override  public boolean equals(Object obj) {  return this.bud.equals(((Rose) obj).getBud());  }  @Override  public int hashCode() {  return this.bud.hashCode();  } } class Bud {  private String color;  private ArrayList<Petal> petals;  Bud () {  short n = (short) (Math.random()\*20+5);  this.petals = new ArrayList<Petal>(n);  for (short i=0; i<n; i++)  this.petals.add(new Petal());  this.color = "Red";  }  Bud (int numOfPetals) {  this.petals = new ArrayList<Petal>(numOfPetals);  for (short i=0; i<numOfPetals; i++)  this.petals.add(new Petal());  this.color = "Red";  }  public void bloom () {  this.color = "Red";  for (Petal p : petals)  p.bloom();  }  public void wither () {  this.color = "Brown";  for (Petal p : petals)  p.wither();  }  public String getColor () {  return this.color;  }  @Override  public String toString() {  StringBuilder buf = new StringBuilder("Цвет бутона: " + this.getColor() + "\n");  for (Petal p : petals)  buf.append(p.toString());  return buf.toString();  }  @Override  public boolean equals(Object obj) {  if (this == obj) return true;  if (obj == null || this.getClass() != obj.getClass()) return false;  Bud b = (Bud) obj;  if (this.color != b.color) return false;  if(this.petals.size() != b.petals.size()) return false;  for (int i=0; i<this.petals.size(); i++)  if (!this.petals.get(i).equals(b.petals.get(i))) return false;  return true;  }  @Override  public int hashCode() {  int res = 0;  for (Petal p : petals)  res += p.hashCode();  return 31 \* (res);  } } class Petal {  private final int ID;  private static int numOfPetals;  private String color;  Petal () {  this.color = "Red";  this.ID = Petal.numOfPetals;  Petal.numOfPetals++;  }  public String getColor () {  return this.color;  }  public void bloom () {  this.color = "Red";  }  public void wither () {  this.color = "Brown";  }  @Override  public String toString() {  return "Лепесток номер " + this.ID + "\nЦвет: " + this.getColor() + "\n";  }  @Override  public boolean equals(Object obj) {  if (this == obj) return true;  if (obj == null || this.getClass() != obj.getClass()) return false;  Petal p = (Petal) obj;  if (this.color != p.color) return false;  return true;  }  @Override  public int hashCode() {  return this.color == "Red" ? 2 : 1;  } } |

**Вывод:**

В ходе выполнения заданий был созданы классы цепной и обычной дроби, дома и телефона, и выполнены задания с использованием этих классов.