|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | |  | | | | | |
|  | | |
|  | Институт информационных технологий (ИТ) |
|  |  |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ** | |
| **по дисциплине** | |
| **«Java»** | |
|  | |
| Выполнил студент группы **ИНБО-06-19** | *Сахневич К.Р.* |
| Принял преподаватель | *Иванов М.Е..* |

|  |  |  |
| --- | --- | --- |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_2020г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2020г. | *(подпись руководителя)* |

Москва 2020

Практическая работа 1.

1. Рандомная генерация чисел.

public class randGen {  
 public static void main(String[] arg){  
 *randGen*();  
 }  
  
 public static int[] arr(int[] arrSum) {  
 boolean sorted = false;  
 int temp;  
 while (!sorted){  
 sorted = true;  
 for (int i = 0; i < arrSum.length - 1; i++){  
 if (arrSum[i] > arrSum[i+1]){  
 temp = arrSum[i];  
 arrSum[i] = arrSum[i+1];  
 arrSum[i +1] = temp;  
 sorted = false;  
 }  
 }  
 }  
 return arrSum;  
 }  
  
 public static void randGen()  
 {  
 int n = 10;  
 int[] arrSum = new int[n];  
 System.*out*.println("Рандомные числа в массиве:");  
 for (int i = 0; i < n; i++)  
 {  
 arrSum[i] = (int)(Math.*random*() \* 100);  
 System.*out*.println(arrSum[i]);  
 }  
 arrSum = *arr*(arrSum);  
  
 System.*out*.println("Массив после сортировки:");  
 for (int i = 0; i < n; i++)  
 {  
 System.*out*.println(arrSum[i]);  
 }  
 }  
}

1. Написать программу, в результате которой считается сумма элементов целочисленного массива с помощью циклов for, while, do while, результат выводится на экран.

public class practical\_1 {  
 public static void main(String[] arg) {  
 int n = 10;  
 int[] arrSum = new int[n];  
 for (int i = 0; i < n; i++)  
 {  
 arrSum[i] = (int)(Math.*random*() \* 100);  
 System.*out*.print(arrSum[i] + " ");  
 }  
 System.*out*.println();  
 int sum = 0;  
 for (int i = 0; i < n; i++) {  
 sum += arrSum[i];  
 }  
 System.*out*.println(sum);  
  
 sum = 0;  
 int i = 0;  
 while (i < n) {  
 sum += arrSum[i];  
 i++;  
 }  
 System.*out*.println(sum);  
  
 sum = 0;  
 i = 0;  
 do {  
 sum += arrSum[i];  
 i++;  
 } while (i < n);  
 System.*out*.println(sum);  
 }  
}

1. Написать программу, в результате работы которой выводятся на экран первые 10 чисел гармонического ряда (форматировать вывод). Вывести сумму гармонического ряда.

public class garmonic {  
 public static void main(String[] arg){  
 int n = 10;  
 double sum = 0;  
 for (int i = 1; i <= n; i++)  
 {  
 double k = (double) 1/i;  
 System.*out*.println(k);  
 sum += k;  
 }  
 System.*out*.println("Сумма всех чисел гармонического ряда: " + sum);  
 }  
}

1. Написать программу, которая с помощью метода, вычисляет факториал числа (использовать управляющую конструкцию цикла), проверить работу метода

import java.util.Scanner;  
  
public class Factorial {  
 public static void main(String[] arg) {  
  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Введите число, факториал которого хотите найти:");  
 int n = scanner.nextInt();  
 int N = 1;  
 for (int i = 1; i <= n; i++)  
 {  
 N \*= i;  
 }  
 System.*out*.println("Факториал числа " + n + " = " + N);  
 }  
}

Практическая работа 2.

1. Разработайте и реализуйте класс Dog (Собака), поля класса описывают кличку и возраст собаки. Необходимо выполнить следующие действия: определить конструктор собаки, чтобы принять и инициализировать данные экземпляра., включить стандартные методы (геттеры/сеттеры) для получения и установки для имени и возраста, включить метод для перевода возраста собаки в "человеческий " возраст (возраст семь раз собаки), включите метод ToString, который возвращает описание экземпляра собаки в виде строки. Создайте массив собак с несколькими элементами.

import java.util.Scanner;  
  
public class main {  
 public static void main(String[] arg) {  
 Scanner s = new Scanner(System.*in*);  
 System.*out*.println("Введите имя собаки: ");  
 String name = s.nextLine();  
 System.*out*.println("Введите возраст собаки: ");  
 int age = s.nextInt();  
 Dog d = new Dog(name, age);  
 boolean flag = true;  
 while (flag) {  
 System.*out*.println("Варианты: 1 - изменить имя; 2 - изменить возраст; 3 - показать возраст собаки на \"Человеческом\"; 4 - показать описание собаки; 5 - показать имя собаки; 6 - показать возраст собаки; 7 - закончить работу.");  
 int i = s.nextInt();  
 switch (i){  
 case 1:  
 {  
 d.setName(name = s.next());  
 break;  
 }  
 case 2:  
 {  
 d.setAge(age = s.nextInt());  
 break;  
 }  
 case 3:  
 {  
 d.translate();  
 break;  
 }  
 case 4:  
 {  
 System.*out*.println(d);  
 break;  
 }  
 case 5:  
 {  
 d.getName();  
 break;  
 }  
 case 6:  
 {  
 d.getAge();  
 break;  
 }  
 case 7:  
 {  
 flag = false;  
 break;  
 }  
 }  
 }  
 }  
}

public class Dog {  
 private String name;  
 private int age;  
  
 public Dog(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void translate() {  
 System.*out*.println("Человеческий возраст собаки: " + 7\*age);  
 }  
  
 public String toString() {  
 return "Имя собаки: " + name +" Возраст собаки: "+ age +" \"Человеческий\" возраст собаки: " + 7\*age;  
 }  
}

1. Реализуйте простейший класс «Книга»

public class book {  
  
 private String author;  
 private String name;  
  
 public void setName(String author) {  
 this.author = author;  
 }  
 public void setAge(String name) {  
 this.name = name;  
 }  
  
 public String getAge() {  
 return name;  
 }  
  
 public String getName() {  
 return author;  
 }  
  
 public String toString() {  
 return "Автор книги: " + author + " Название книги: " + name;  
 }  
}

Практическая работа 3

1. Создать абстрактный класс, описывающий посуду (Dish). С помощью наследования реализовать различные виды посуды, имеющие свои свойства и методы. Протестировать работу классов.

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] arg) {  
 Scanner s = new Scanner(System.*in*);  
 System.*out*.println("Выберите вид посуды: 1 - кухонная; 2 - столовая");  
 int f = s.nextInt();  
 if(f == 1) {  
 CookingWare a = new CookingWare();  
 System.*out*.println("Введите материалы из которых выполнена посуда: ");  
 a.setMaterial(s.next());  
 System.*out*.println("Введите тип кухонной посуды(это могут быть: кастрюли, сковородки и т.д.): ");  
 a.setType(s.next());  
 System.*out*.println("Введите общее число посудных изделий для кухни: ");  
 a.setCount(s.nextInt());  
 System.*out*.println("Введите особенности посуды: ");  
 a.setFeatures(s.next());  
 System.*out*.println(a.getInfo());  
 }  
 else {  
 TableWare b = new TableWare();  
 System.*out*.println("Введите материалы из которых выполнена посуда: ");  
 b.setMaterial(s.next());  
 System.*out*.println("Введите тип столовой посуды(это могут быть: вилки, бакалы, стаканы и т.д.): ");  
 b.setType(s.next());  
 System.*out*.println("Введите общее число посудных изделий для столовой: ");  
 b.setCount(s.nextInt());  
 System.*out*.println("Введите особенности посуды: ");  
 b.setFeatures(s.next());  
 System.*out*.println(b.getInfo());  
 }  
 }  
}

import Base.Dish;  
  
public class TableWare extends Dish {  
  
 private String features;  
  
 public void setFeatures(String features){  
 this.features = features;  
 }  
  
 public String getFeatures() {  
 return features;  
 }  
  
 @Override  
 public void setMaterial(String material) {  
 this.material = material;  
 }  
  
 @Override  
 public void setType(String type) {  
 this.type = type;  
 }  
  
 @Override  
 public void setCount(int count) {  
 this.count = count;  
 }  
  
 @Override  
 public String getInfo() {  
 return " Материалы: " + material + " Тип: " + type + " Кол-во: " + count + " Особенности: " + features;  
 }  
  
}

import Base.Dish;  
  
public class CookingWare extends Dish {  
  
 private String features;  
  
 public void setFeatures(String features){  
 this.features = features;  
 }  
  
 public String getFeatures() {  
 return features;  
 }  
  
  
 @Override  
 public void setMaterial(String material) {  
 this.material = material;  
 }  
  
 @Override  
 public void setType(String type) {  
 this.type = type;  
 }  
  
 @Override  
 public void setCount(int count) {  
 this.count = count;  
 }  
  
 @Override  
 public String getInfo() {  
 return " Материалы: " + material + " Тип: " + type + " Кол-во: " + count + " Особенности: " + features;  
 }  
  
}

package Base;  
  
public abstract class Dish {  
  
 protected String material;  
 protected String type;  
 protected int count;  
  
 public abstract void setCount(int count);  
  
 public abstract void setType(String type);  
  
 public abstract void setMaterial(String material);  
  
 public abstract String getInfo();  
  
  
}

1. Создать абстрактный класс, описывающий собак (Dog). С помощью наследования реализовать различного вида пород собак. Протестировать работу классов.

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] arg) {  
 Scanner s = new Scanner(System.*in*);  
 System.*out*.println("Выберите породу собаки: 1 - Бордер колли; 2 - Немецкая овчарка");  
 int f = s.nextInt();  
 if(f == 1) {  
 BorderCollie a = new BorderCollie();  
  
 System.*out*.println("Введите имя собаки: ");  
 String name = s.next();  
 a.setName(name);  
  
 System.*out*.println("Введите вес собкаи: ");  
 String weight = s.next();  
 a.setWeight(weight);  
  
 System.*out*.println("Введите кол-во собак данной породы: ");  
 int count = s.nextInt();  
 a.setCount(count);  
  
 System.*out*.println("Введите особенности собаки: ");  
 String features = s.next();  
 a.setFeatures(features);  
  
 System.*out*.println(a.getInfo());  
 }  
 else {  
 GermanShepherd b = new GermanShepherd();  
  
 System.*out*.println("Введите имя собаки: ");  
 String name = s.next();  
 b.setName(name);  
  
 System.*out*.println("Введите вес собкаи: ");  
 String weight = s.next();  
 b.setWeight(weight);  
  
 System.*out*.println("Введите кол-во собак данной породы: ");  
 int count = s.nextInt();  
 b.setCount(count);  
  
 System.*out*.println("Введите особенности собаки: ");  
 String features = s.next();  
 b.setFeatures(features);  
 System.*out*.println(b.getInfo());  
 }  
 }  
}

import Base.Dog;  
  
public class GermanShepherd extends Dog {  
 private String features;  
  
 public void setFeatures(String features){  
 this.features = features;  
 }  
  
 public String getFeatures() {  
 return features;  
 }  
  
 @Override  
 public void setName(String name) {  
 this.name = name;  
 }  
 @Override  
 public void setWeight(String weight) {  
 this.weight = weight;  
 }  
  
 @Override  
 public void setCount(int count) {  
 this.count = count;  
 }  
  
 @Override  
 public String getInfo() {  
 return "Имя: " + name + " Вес: " + weight + " Кол-во: " + count + " Особенности: " + features;  
 }  
}

package Base;  
  
public abstract class Dog {  
  
 protected String name;  
 protected String weight;  
 protected int count;  
  
 public abstract void setName(String name);  
  
 public abstract void setCount(int count);  
  
 public abstract void setWeight(String weight);  
  
 public abstract String getInfo();  
  
  
}

import Base.Dog;  
  
  
public class BorderCollie extends Dog {  
 private String features;  
  
 public void setFeatures(String features){  
 this.features = features;  
 }  
  
 public String getFeatures() {  
 return features;  
 }  
  
 @Override  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public void setWeight(String weight) {  
 this.weight = weight;  
 }  
  
 @Override  
 public void setCount(int count) {  
 this.count = count;  
 }  
  
 @Override  
 public String getInfo() {  
 return "Имя: " + name + " Вес: " + weight + " Кол-во: " + count + " Особенности: " + features;  
 }  
}

1. Создать абстрактный класс, описывающий мебель. С помощью наследования реализовать различные виды мебели. Реализовать класс магазина мебели (FurnitureShop), реализовать каталог мебели, с возможность покупки. Создать консольный интерфейс.

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 FurnitureShop furnitureShop = new FurnitureShop();  
 furnitureShop.setCatalog();  
  
 int choice;  
 int number;  
  
 boolean f = true;  
  
 while (f) {  
  
 System.*out*.println("В нашем ассортименте есть: \n 1) Диваны \n 2) Стулья \n 3) Столы \n 4) Перейти в корзину \n 5) Выйти измагазина \n Выберите номер интересующего вас варианта и нажмите enter: ");  
  
 choice = in.nextInt();  
  
  
 switch (choice){  
 case 1:  
 furnitureShop.getSofasCatalog();  
 System.*out*.println("Выберите номер дивана для покупки: ");  
 number = in.nextInt();  
 furnitureShop.addSofaToBasket(number);  
 furnitureShop.deleteSofa(number);  
 break;  
 case 2:  
 furnitureShop.getChairsCatalog();  
 System.*out*.println("Выберите номер стула для покупки: ");  
 number = in.nextInt();  
 furnitureShop.addChairToBasket(number);  
 furnitureShop.deleteChair(number);  
 break;  
 case 3:  
 furnitureShop.getTablesCatalog();  
 System.*out*.println("Выберите номер стола для покупки: ");  
 number = in.nextInt();  
 furnitureShop.addTableToBasket(number);  
 furnitureShop.deleteTable(number);  
 break;  
 case 4:  
 furnitureShop.getBasket();  
  
 System.*out*.println("Желаете приобрести выбранные товары? 1) Да 2) Нет");  
  
 number = in.nextInt();  
 if (number == 1) {  
 furnitureShop.deleteBasket();  
 }  
 else if (number == 2) {  
 furnitureShop.restoreCatalog();  
 }  
 break;  
 case 5:  
 System.*out*.println("Всегда Вам рады, обращайтесь еще!");  
 f = false;  
 break;  
  
 }  
 }  
 }  
}

import java.util.ArrayList;  
import java.util.List;  
  
public class FurnitureShop {  
 private List<Chair> chairs;  
 private List<Table> tables;  
 private List<Sofa> sofas;  
 private ArrayList<Chair> chairsBasket;  
 private ArrayList<Table> tablesBasket;  
 private ArrayList<Sofa> sofasBasket;  
  
 public FurnitureShop() {  
 chairs = new ArrayList<Chair>();  
 tables = new ArrayList<Table>();  
 sofas = new ArrayList<Sofa>();  
 chairsBasket = new ArrayList<Chair>();  
 tablesBasket = new ArrayList<Table>();  
 sofasBasket = new ArrayList<Sofa>();  
 }  
  
 public void setCatalog() {  
  
 for (int i = 0; i < 4; i++) {  
 if (i % 2 == 0)  
 chairs.add(new Chair(5500,"straw", 80));  
 else {  
 chairs.add(new Chair(4200,"brown", 120));  
 }  
 }  
  
 for (int i = 0; i < 4; i++) {  
 if (i % 2 == 0)  
 tables.add(new Table(15000,"straw", 220));  
 else {  
 tables.add(new Table(18500,"brown", 190));  
 }  
 }  
  
 for (int i = 0; i < 4; i++) {  
 if (i % 2 == 0)  
 sofas.add(new Sofa(30000, "white", 200));  
 else {  
 sofas.add(new Sofa(25000,"black", 180));  
 }  
 }  
  
 }  
  
 public void getChairsCatalog() {  
 for (int i = 0; i < chairs.size(); i++) {  
 System.*out*.println("№" + (i +1) + " " + chairs.get(i));  
 }  
 }  
 public void getTablesCatalog() {  
 for (int i = 0; i < tables.size(); i++) {  
 System.*out*.println("№" + (i +1) + " " + tables.get(i));  
 }  
 }  
  
 public void getSofasCatalog(){  
 for (int i = 0; i < sofas.size(); i++) {  
 System.*out*.println("№" + (i +1) + " " + sofas.get(i));  
 }  
 }  
  
 public void deleteChair(int pos) {  
 chairs.remove(pos - 1);  
 }  
  
 public void deleteTable(int pos) {  
 tables.remove(pos - 1);  
 }  
  
 public void deleteSofa(int pos){  
 sofas.remove(pos - 1);  
 }  
  
 public void addChairToBasket(int i) {  
 chairsBasket.add(chairs.get(i - 1));  
 }  
  
 public void addTableToBasket(int i) {  
 tablesBasket.add(tables.get(i - 1));  
 }  
  
 public void addSofaToBasket(int i){  
 sofasBasket.add(sofas.get(i - 1));  
 }  
  
 public void getBasket() {  
 int sum = 0;  
 if (chairsBasket.size() == 0 && tablesBasket.size() == 0 && sofasBasket.size() == 0) {  
 System.*out*.println("Корзина пуста");  
 return;  
 }  
  
 System.*out*.println("Кровати:");  
 for (int i = 0; i < sofasBasket.size(); i++) {  
 System.*out*.println("№" + (i + 1) + " " + sofasBasket.get(i));  
 sum += sofasBasket.get(i).getPrice();  
 }  
 if (sofasBasket.size() == 0) System.*out*.println("-\n");  
  
 System.*out*.println("Cтулья:");  
 for (int i = 0; i < chairsBasket.size(); i++) {  
 System.*out*.println("№" + (i + 1) + " " + chairsBasket.get(i));  
 sum += chairsBasket.get(i).getPrice();  
 }  
 if (chairsBasket.size() == 0) System.*out*.println("-\n");  
  
 System.*out*.println("Столы:");  
 for (int i = 0; i < tablesBasket.size(); i++) {  
 System.*out*.println("№" + (i + 1) + " " + tablesBasket.get(i));  
 sum += tablesBasket.get(i).getPrice();  
 }  
 if (tablesBasket.size() == 0) System.*out*.println("-\n");  
  
 System.*out*.println("\n СТОИМОСТЬ + " + sum);  
 }  
  
  
 public void deleteBasket() {  
 chairsBasket.clear();  
 tablesBasket.clear();  
 sofasBasket.clear();  
 }  
  
  
 public void restoreCatalog() {  
 chairs.addAll(chairsBasket);  
 tables.addAll(tablesBasket);  
 sofas.addAll(sofasBasket);  
 }  
}

import Base.Furniture;  
  
public class Sofa extends Furniture {  
 private boolean isBroken;  
  
 public void setBroken(boolean broken) {  
 isBroken = broken;  
 }  
  
 protected Sofa(int price, String color, int length) {  
 super(price, color, length);  
 }  
  
 @Override  
 public void toRepair() {  
 if (isBroken){  
 System.*out*.println("Этот диван нужно починить");  
 } else {  
 System.*out*.println("Этот диван в хорошем состоянии");  
 }  
 }  
  
 @Override  
 public String toString() {  
 return "Sofa{" +  
 "price=" + price +  
 ", color='" + color + '\'' +  
 ", length=" + length +  
 '}';  
 }  
}

import Base.Furniture;  
  
public class Chair extends Furniture {  
 private boolean isBroken;  
  
 public void setBroken(boolean broken) {  
 isBroken = broken;  
 }  
 protected Chair(int price, String color, int length) {  
 super(price, color, length);  
 }  
 @Override  
 public void toRepair() {  
 if (isBroken) {  
 System.*out*.println("Этот стул нужно починить");  
 } else {  
 System.*out*.println("Этот стул в хорошем состоянии");  
 }  
 }  
 @Override  
 public String toString() {  
 return "Chair{" +  
 "price=" + price +  
 ", color='" + color + '\'' +  
 ", length=" + length +  
 '}';  
 }  
}

import Base.Furniture;  
  
public class Table extends Furniture {  
 private boolean isBroken;  
  
 protected Table(int price, String color, int length) {  
 super(price, color, length);  
 }  
  
 public void setBroken(boolean broken) {  
 isBroken = broken;  
 }  
  
 @Override  
 public void toRepair() {  
 if (isBroken){  
 System.*out*.println("Этот стол нужно починить");  
 } else {  
 System.*out*.println("Этот стол в хорошем состоянии");  
 }  
 }  
  
 @Override  
 public String toString() {  
 return "Table{" +  
 "price=" + price +  
 ", color='" + color + '\'' +  
 ", length=" + length +  
 '}';  
 }  
}

package Base;  
  
public abstract class Furniture {  
 protected int price;  
 protected String color;  
 protected int length;  
  
 public int getPrice() {  
 return price;  
 }  
  
 public int rand(int a, int b) {  
 return (int) ( Math.*random*() \* (b - a + 1) + a );  
 }  
  
 protected Furniture(int price, String color, int length) {  
 price = rand( (price/2), price);  
 length = rand( (length - length/3), length);  
  
 this.price = price;  
 this.color = color;  
 this.length = length;  
 }  
  
 public abstract void toRepair();  
  
}

1. Воссоздать классы из UML диаграммы.



import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Хотите создать: 1 - Круг, 2 - Прямоуглольник, 3 - Квадрат");  
 int i = in.nextInt();  
 switch (i) {  
 case (1): {  
 in.nextLine();  
 System.*out*.println("Введите цвет Круга: ");  
 String color = in.nextLine();  
 System.*out*.println("Введите радиус: ");  
 double radius = in.nextDouble();  
 Circle cir = new Circle(radius, color, true);  
 break;  
 }  
 case (2): {  
 in.nextLine();  
 System.*out*.println("Введите цвет Прямоугольника: ");  
 String color = in.nextLine();  
 System.*out*.println("Введите длину: ");  
 double length = in.nextDouble();  
 System.*out*.println("Введите ширину: ");  
 double width = in.nextDouble();  
 Rectangle rectangle = new Rectangle(width,length);  
  
 System.*out*.println(rectangle);  
 System.*out*.println(rectangle.getPerimeter());  
 break;  
 }  
 case (3): {  
 in.nextLine();  
 System.*out*.println("Введите цвет Квадрата: ");  
 String color = in.nextLine();  
 System.*out*.println("Введите длину стороны: ");  
 double side = in.nextDouble();  
 Square square = new Square(10);  
 System.*out*.println(square);  
 System.*out*.println(square.getWidth());  
 break;  
 }  
 }  
 }  
}

public class Rectangle extends Shape {  
 protected double width;  
 protected double length;  
 public Rectangle() {}  
 public Rectangle(double width, double length) {  
 this.width = width;  
 this.length = length;  
 }  
 public Rectangle(double width, double length, String color, boolean filled) {  
 super(color, filled);  
 this.width = width;  
 this.length = length;  
 }  
 public double getWidth() {  
 return width;  
 }  
 public void setWidth(double width) {  
 this.width = width;  
 }  
 public double getLength() {  
 return length;  
 }  
 public void setLength(double length) {  
 this.length = length;  
 }  
 @Override  
 public double getArea() {  
 return this.width\*this.length;  
 }  
 @Override  
 public double getPerimeter() {  
 return 2\*this.width+2\*this.length;  
 }  
 @Override  
 public String toString() {  
 return " Ширина прямоугольника: " + width + "\n" + " Длина прямоугольника: " + length + "\n"  
 + " Площадь прямоугольника " + getArea() + "\n"  
 + " Периметр Прямоугольника " + getPerimeter();  
 }  
}

public class Square extends Rectangle {  
 public Square() {  
 }  
 public Square(double side) {  
 super(side, side);  
 }  
 public Square(String color, boolean filled, double side) {  
 super(side, side, color, filled);  
 }  
 public double getSide() {  
 return width;  
 }  
 public void setSide(double side) {  
 this.width = side;  
 this.length = side;  
 }  
 @Override  
 public double getArea() {  
 return this.width \* this.length;  
 }  
 @Override  
 public double getPerimeter() {  
 return 4\*this.width;  
 }  
 public String toString() {  
 return " Длина стороны квадрата: " + width + "\n"  
 + " Площадь квадрата " + getArea() + "\n"  
 + " Периметр квадрата " + getPerimeter();  
 }  
}

public class Circle extends Shape {  
 protected double radius;  
 public double getRadius() {  
 return radius;  
 }  
 public void setRadius(double radius) {  
 this.radius = radius;  
 }  
 @Override  
 public double getArea() {  
 return radius\*radius\*3.14;  
 }  
 @Override  
 public double getPerimeter() {  
 return 2\*3.14\*radius;  
 }  
 @Override  
 public String toString() {  
 return "Радиус круга: " + radius + "\n"  
 + " Площадь круга " + getArea() + "\n"  
 + " Длина круга " + getPerimeter();  
 }  
 public Circle() {  
 }  
 public Circle(double radius) {  
 setRadius(radius);  
 }  
 public Circle(double radius,String color,boolean filled) {  
 setColor(color);  
 setFilled(filled);  
 Circle cr = new Circle(radius);  
 System.*out*.println("Цвет круга: " + getColor() + "\n"  
 + cr.toString());  
 }  
}

import java.util.Scanner;  
  
public abstract class Shape {  
 protected String color;  
 protected boolean filled;  
 public Shape() {  
  
 }  
 public Shape(String color,boolean filled) {  
 setColor(color);  
 setFilled(filled);  
 System.*out*.println(toString());  
 }  
 public String getColor() {  
 return color;  
 }  
 public void setColor(String color) {  
 this.color = color;  
 }  
 public boolean isFilled() {  
 return filled;  
 }  
 public void setFilled(boolean filled) {  
 this.filled = filled;  
 }  
 public abstract double getArea() ;  
 public abstract double getPerimeter();  
 @Override  
 public String toString() {  
 return "Фигура: " + filled + "\n" + " Цвет: " + color;  
 }  
}

Практическая работа 4

1. Создайте класс Employee, определяющий сотрудника. Необходимые поля –

Фамилия, Имя, фиксированная часть зарплаты(оклад), должность

2. Создайте интерфейс EmployeePosition, определяющий должность. Определите в нем метод String getJobTitle(), возвращающий название занимаемой должности, и double calcSalary(double baseSalary), возвращающий зарплату сотрудника, в соответствии с фиксированным окладом.

3. Создайте классы, определяющие должности сотрудников. Все они должны

реализовывать интерфейс EmployeePosition a. Manager — зарплата складывается из фиксированной части и бонуса в виде 5% от заработанных для компании денег. Количество заработанных денег для компании генерируйте случайным образом от 115 000 до 140 000 рублей.

b. TopManager — зарплата складывается из фиксированной части и бонуса в виде 150% от заработной платы, если доход компании более 10 млн рублей. c. Operator — зарплата складывается только из фиксированной части.

4. Создайте класс компании Company, содержащий сотрудников и реализующей методы:

o найм одного сотрудника — hire(),

o найм списка сотрудников – hireAll(),

o увольнение сотрудника – fire(),

o получение значения дохода компании – getIncome().

Аргументы и возвращаемое значение методов выберите на основании логики работы вашего приложения.

5. Создайте два метода, возвращающие список указанной длины (count). Они должны содержать сотрудников, отсортированных по убыванию и возрастанию заработной платы:

 List&lt;Employee&gt; getTopSalaryStaff(int count),

 List&lt;Employee&gt; getLowestSalaryStaff(int count).

Аргументы и возвращаемое значение метода выберите в соответствии с логикой начисления зарплат.

Для демонстрации и тестирования работы ваших классов:

1. Создайте компанию и наймите в неё: 180 операторов Operator, 80 менеджеров по продажам Manager, 10 топ-менеджеров TopManager.

2. Распечатайте список из 10–15 самых высоких зарплат в компании.

3. Распечатайте список из 30 самых низких зарплат в компании.

4. Увольте 50% сотрудников.

5. Распечатайте список из 10–15 самых высоких зарплат в компании.

6. Распечатайте список из 30 самых низких зарплат в компании.

Примеры вывода списка зарплат

Список из пяти зарплат по убыванию:

 230 000 руб.

 178 000 руб.

 165 870 руб.

 123 000 руб.

 117 900 руб.

Рекомендации

 Можно создавать разные экземпляры компании со своим списком сотрудников и доходом.

 Чтобы получить данные компании внутри класса должности, настройте хранение ссылки на Company и передавайте объект Company с помощью конструктора или сеттера.

 Учтите, в методы получения списков зарплат могут передаваться значения count, превышающие количество сотрудников в компании, или отрицательные.

import java.util.List;  
  
public class Main {  
 public static void main(String[] args) {  
 Company company = new Company();  
 company.generateEmployees(company); // генерирую сотрудников Оп. 180; Мен. 80; ТопМен. 10.  
 company.calcAll(); // считаем зарплаты все сотрудников с базовой ставкой от 25000 до 35000 рандомно  
 System.*out*.println("Доход компании: " + company.getProfitCompany());  
 System.*out*.println();  
 List<Employee> high = company.getTopSalaryStaff(10);  
 List<Employee> low = company.getLowestSalaryStaff(30);  
 System.*out*.println("Топ 10, самые высокие зп\nЗП: \t\t\t Имя,\t\t Фамилия: \t\t\t Должность");  
 for (Employee item : high) {  
 System.*out*.println(item.position.getFinalSalary() + " " + item + " " + item.position.getJobTitle() + "\n");  
 }  
 System.*out*.println("Топ 30, самые низкие зп\nЗП: \t\t\t Имя,\t\t Фамилия: \t\t\t Должность");  
 for (Employee value : low) {  
 System.*out*.println(value.position.getFinalSalary() + " " + value + " " + value.position.getJobTitle() +"\n");  
 }  
 for (int i = 0; i < company.getEmployees().size(); i++) {  
 if (i%2 == 1) {  
 company.fire(i);  
 }  
 }  
 System.*out*.println("После сокращения половины сотрудников");  
 ///////////////  
 high = company.getTopSalaryStaff(10);  
 low = company.getLowestSalaryStaff(30);  
 System.*out*.println("Топ 10, самые высокие зп\nЗП: \t\t\t Имя,\t\t Фамилия: \t\t\t Должность");  
 for (Employee employee : high) {  
 System.*out*.println(employee.position.getFinalSalary() + " " + employee + " " + employee.position.getJobTitle() + "\n");  
 }  
 System.*out*.println("Топ 30, самые низкие зп\nЗП: \t\t\t Имя,\t\t Фамилия: \t\t\t Должность");  
 for (Employee employee : low) {  
 System.*out*.println(employee.position.getFinalSalary() + " " + employee + " " + employee.position.getJobTitle() +"\n");  
 }  
 System.*out*.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ALL");  
 company.print(181); // нарошно ввожу число большее размера списка для обработку этой ситуации  
 company.print(130);  
 }  
}

ёpublic interface EmployeePosition {  
 String getJobTitle(); // название занимающей должности  
 double calcSalary(double baseSalary);  
 double getFinalSalary();  
}

import java.util.Random;  
public class Manager implements EmployeePosition {  
 private double finalSalary;  
 private Company company;  
 private double profitManag;  
 public Manager (Company company){  
 this.company = company;  
 }  
 public double getFinalSalary() {  
 return finalSalary;  
 }  
 @Override  
 public String getJobTitle() {  
 return "Manager";  
 }  
 @Override  
 public double calcSalary(double baseSalary) {  
 Random r = new Random();  
 profitManag = (double) r.nextInt((140000 - 115000) + 1) + 115000;  
 company.refProfitCompany(profitManag);  
 finalSalary = (double) r.nextInt(((int)(baseSalary+profitManag\*0.05) - (int)baseSalary) + 1) + baseSalary;  
 return finalSalary;  
 }  
}

public class Operator implements EmployeePosition {  
 private double finalSalary;  
 private Company company;  
 public Operator (Company company){  
 this.company = company;  
 }  
 public double getFinalSalary() {  
 return finalSalary;  
 }  
 @Override  
 public String getJobTitle() {  
 return "Operator";  
 }  
 @Override  
 public double calcSalary(double baseSalary) {  
 return finalSalary = baseSalary;  
 }  
}

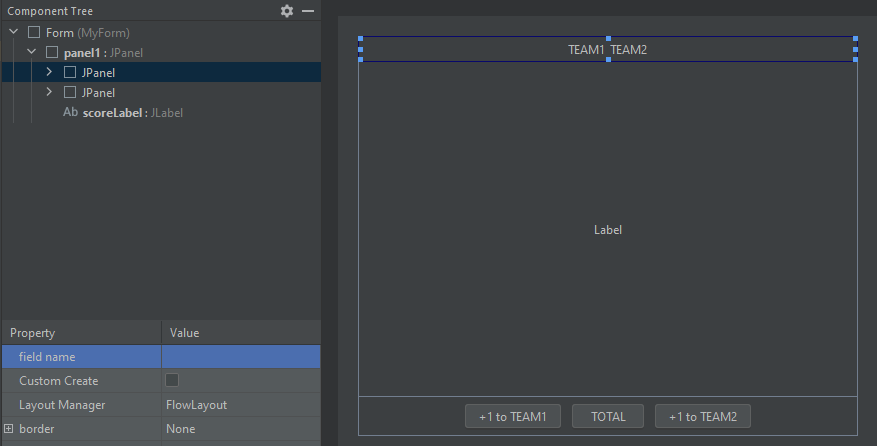
import java.util.Random;  
  
public class TopManager implements EmployeePosition {  
 private double finalSalary;  
 private final Company company;  
 private boolean f;  
 public TopManager(Company company) {  
 this.company = company;  
 }  
 public void setF(){  
 double temp = company.getProfitCompany();  
 if (temp > 10000000)  
 f = true;  
 else f = false;  
 }  
 @Override  
 public double getFinalSalary() {  
 return finalSalary;  
 }  
 @Override  
 public String getJobTitle() {  
 return "Top-Manager";  
 }  
 @Override  
 public double calcSalary(double baseSalary) {  
 Random r = new Random();  
 setF();  
 return company.getProfitCompany() > 10000000 ? baseSalary\*1.5 : baseSalary;  
 }  
}

import java.util.\*;  
  
public class Company {  
 private List<Employee> employees;  
 private double profitCompany;  
 private Random r = new Random();  
 public List<Employee> getEmployees() {  
 return employees;  
 }  
 public void hair(Employee employee) {  
 if (employees == null) {  
 employees = new ArrayList<>();  
 }  
 employees.add(employee);  
 }  
 public void hairAll(List<Employee> employeeList) {  
 if (employees == null) {  
 employees = new ArrayList<>();  
 }  
 employees.addAll(employeeList);  
 }  
 public void fire(int pos) {  
 employees.remove(pos);  
 }  
 List<Employee> getTopSalaryStaff(int count) {  
 ArrayList<Employee> topList= new ArrayList<Employee>();  
 if (count < 1) {  
 System.*out*.println("Сотрудников нет");  
 return topList;  
 }  
 Comparator<Employee> comparator = new Comparator<Employee>() {  
 @Override  
 public int compare(Employee o1, Employee o2) {  
 double O1 = o1.position.getFinalSalary();  
 double O2 = o2.position.getFinalSalary();  
 return Double.*compare*(O2, O1);  
 }  
 };  
 employees.sort(comparator);  
 for (int i = 0; i < count; i++) {  
 topList.add(employees.get(i));  
 }  
 return topList;  
 }  
 List<Employee> getLowestSalaryStaff(int count) {  
 ArrayList<Employee> lowList = new ArrayList<Employee>();  
 if (count < 1) {  
 System.*out*.println("Сотрудников нет");  
 return lowList;  
 }  
 Comparator<Employee> comparator = new Comparator<Employee>() {  
 @Override  
 public int compare(Employee o1, Employee o2) {  
 double O1 = o1.position.getFinalSalary();  
 double O2 = o2.position.getFinalSalary();  
 if (O1 < O2) {  
 return 1;  
 }  
 else if (O1 > O2) {  
 return -1;  
 } else {  
 return 0;  
 }  
 }  
 };  
 employees.sort(comparator);  
 Collections.*reverse*(employees);  
 for (int i = 0; i < count; i++) {  
 lowList.add(employees.get(i));  
 }  
 return lowList;  
 }  
 public void generateEmployees(Company company) {  
 Employee emp;  
 for (int i = 0; i < 80; i++) {  
 if (i%3 == 1) {  
 emp = new Employee("Kate", "Romero", new Manager(company));  
 hair(emp);  
 } else if (i%3 == 2) {  
 emp = new Employee("Fedor", "Rurikov", new Manager(company));  
 hair(emp);  
 } else {  
 emp = new Employee("Kirill", "Sakhnevich", new Manager(company));  
 hair(emp);  
 }  
 }  
 for (int i = 0; i < 180; i++) {  
 if (i%3 == 1) {  
 emp = new Employee("Sasha", "Ivanov", new Operator(company));  
 hair(emp);  
 } else if (i%3 == 2) {  
 emp = new Employee("Ruslan", "Chencov", new Operator(company));  
 hair(emp);  
 } else {  
 emp = new Employee("Roman", "Petrov", new Operator(company));  
 hair(emp);  
 }  
 }  
 for (int i = 0; i < 10; i++) {  
 if (i%3 == 1) {  
 emp = new Employee("Peter", "Darlov", new TopManager(company));  
 hair(emp);  
 } else if (i%3 == 2) {  
 emp = new Employee("John", "Kelvin", new TopManager(company));  
 hair(emp);  
 } else {  
 emp = new Employee("Maria", "Pavlova", new TopManager(company));  
 hair(emp);  
 }  
 }  
 }  
 public void refProfitCompany(double profitEmployee){  
 profitCompany += profitEmployee;  
 }  
 public double getProfitCompany(){  
 return profitCompany;  
 }  
 public void calcAll() {  
 for (Employee employee : employees) {  
 employee.position.calcSalary(r.nextInt((35000 - 25000) + 1) + 25000);  
 }  
 }  
 public void print(int n) {  
 if (n > employees.size())  
 System.*out*.println("Размер введенный для вывода больше, чем размера списка сотрудников");  
 else {  
 for (int i = 0; i < n; i++) {  
 System.*out*.println(employees.get(i) + " " + employees.get(i).position.getFinalSalary() + " " + employees.get(i).position.getJobTitle() + "\n");  
 }  
 }  
 }  
}

public class Employee {  
 protected String name;  
 protected String surname;  
 protected EmployeePosition position;  
  
 public Employee(String name, String surname, EmployeePosition position) {  
 this.name = name;  
 this.surname = surname;  
 this.position = position;  
 }  
  
  
 @Override  
 public String toString() {  
 return "Employee{" +  
 "name='" + name + '\'' +  
 ", surname='" + surname + '\'' +  
 '}';  
 }  
  
}

Практическая работа 5

Gui приложение с использованием библиотеки javax.swing



public class Loader {  
  
 public static void main(String[] args) {  
 Team team1 = new Team("BLACK");  
 Team team2 = new Team("WHITE");  
 new MyForm(team1, team2);  
 }  
  
}

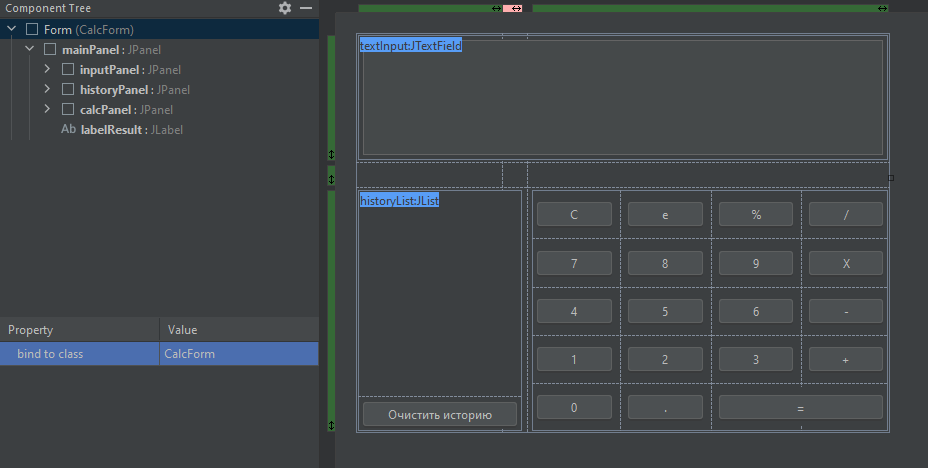
public class Team {  
 private int score;  
 private String name;  
  
 public Team(String name) {  
 this.score = 0;  
 this.name = name;  
 }  
  
 public void setScore(int score) {  
 this.score = score;  
 }  
  
 public void increaseScore() {  
 score++;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public int getScore() {  
 return score;  
 }  
}

import javax.swing.\*;  
import java.awt.event.\*;  
public class MyForm extends JFrame{  
 private JPanel panel1;  
 private JButton team1Button;  
 private JButton team2Button;  
 private JLabel team1Label;  
 private JLabel team2Label;  
 private JButton totalButton;  
 private JLabel scoreLabel;  
 public MyForm(Team team1, Team team2) {  
 setVisible(true);  
 setSize(600, 400);  
 setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);  
 setLocationRelativeTo(null);  
 add(panel1);  
 scoreLabel.setText("0 x 0");  
 team1Label.setText(team1.getName());  
 team2Label.setText(team2.getName());  
 team1Button.addMouseListener(new MouseListener() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 team1.increaseScore();  
 scoreLabel.setText(team1.getScore() + " " + "x" + " " + team2.getScore());  
 }  
 @Override  
 public void mousePressed(MouseEvent e) {  
 }  
 @Override  
 public void mouseReleased(MouseEvent e) {  
 }  
 @Override  
 public void mouseEntered(MouseEvent e) {  
 }  
 @Override  
 public void mouseExited(MouseEvent e) {  
 }  
 });  
 team2Button.addMouseListener(new MouseListener() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 team2.increaseScore();  
 scoreLabel.setText(String.*valueOf*(team1.getScore()) + " " + "x" + " " + String.*valueOf*(team2.getScore()));  
 }  
 @Override  
 public void mousePressed(MouseEvent e) {  
 }  
 @Override  
 public void mouseReleased(MouseEvent e) {  
 }  
 @Override  
 public void mouseEntered(MouseEvent e) {  
 }  
 @Override  
 public void mouseExited(MouseEvent e) {  
 }  
 });  
 totalButton.addMouseListener(new MouseListener() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 if (team1.getScore() > team2.getScore()) {  
 JOptionPane.*showMessageDialog*(MyForm.this, "Team " + team1.getName() + " won!");  
 } else if (team1.getScore() < team2.getScore()) {  
 JOptionPane.*showMessageDialog*(MyForm.this, "Team " + team2.getName() + " won!");  
 }  
 else {  
 JOptionPane.*showMessageDialog*(MyForm.this, "DRAW");  
 }  
 team1.setScore(0);  
 team2.setScore(0);  
 scoreLabel.setText("0 x 0");  
 }  
 @Override  
 public void mousePressed(MouseEvent e) {  
 }  
 @Override  
 public void mouseReleased(MouseEvent e) {  
 }  
 @Override  
 public void mouseEntered(MouseEvent e) {  
 }  
 @Override  
 public void mouseExited(MouseEvent e) {  
 }  
 });  
 }  
}

Практическая работа 6

Gui приложение с использованием библиотеки javax.swing

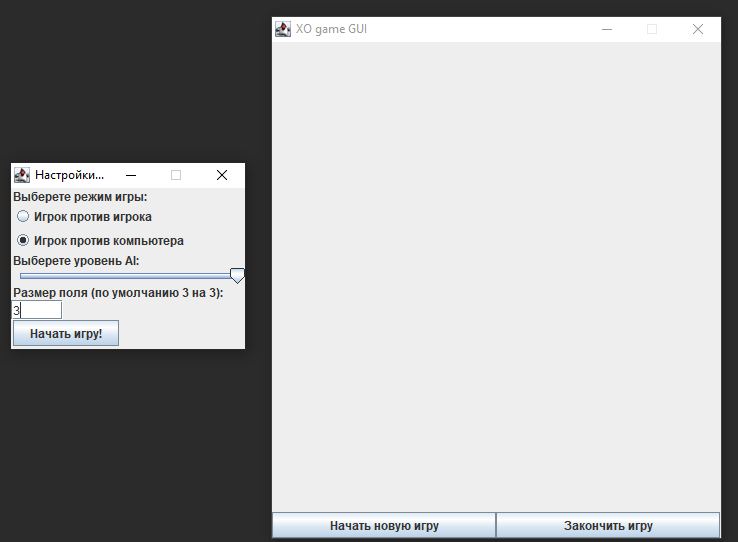
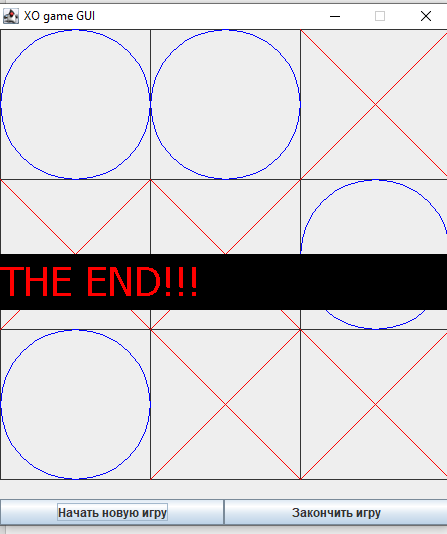
Калькулятор



import javax.swing.\*;  
import javax.swing.event.ListSelectionEvent;  
import javax.swing.event.ListSelectionListener;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
  
public class CalcForm extends JFrame{  
 private JPanel mainPanel;  
 private JList historyList;  
 private JButton buttonReset;  
 private JButton button7;  
 private JButton button4;  
 private JButton button1;  
 private JButton buttonE;  
 private JButton button8;  
 private JButton button5;  
 private JButton button2;  
 private JButton buttonPercent;  
 private JButton button9;  
 private JButton button6;  
 private JButton button3;  
 private JButton buttonDivide;  
 private JButton buttonMultiply;  
 private JButton buttonSubtract;  
 private JButton buttonAdd;  
 private JButton buttonEqually;  
 private JButton button0;  
 private JButton buttonSeparator;  
 private JTextField textInput;  
 private JPanel inputPanel;  
 private JPanel historyPanel;  
 private JButton buttonCleanHistory;  
 private JPanel calcPanel;  
 private JLabel labelResult;  
 private ArrayList<Calc> result;  
 private DefaultListModel listCalcModel;  
 public CalcForm() {  
 super("My calc");  
 this.setContentPane(mainPanel);  
 this.setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);  
 setSize(700, 700);  
 result = new ArrayList<Calc>();  
 listCalcModel = new DefaultListModel();  
 historyList.setModel(listCalcModel);  
  
 buttonCleanHistory.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 result.clear();  
 refreshCalcList();  
 }  
 });  
 historyList.addListSelectionListener(new ListSelectionListener() {  
 @Override  
 public void valueChanged(ListSelectionEvent e) {  
 int historyNumber = historyList.getSelectedIndex();  
 if (historyNumber >= 0){  
 Calc c = result.get(historyNumber);  
 textInput.setText(c.getRecord());  
 labelResult.setText(c.getResult());  
  
 buttonEqually.setEnabled(true);  
 }  
 else {  
 buttonEqually.setEnabled(false);  
 }  
 }  
 });  
 buttonReset.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 int historyNumber = historyList.getSelectedIndex();  
 if (historyNumber >= 0){  
// Calc c = result.get(historyNumber);  
// c.setRecord("");  
 result.remove(historyNumber);  
 refreshCalcList();  
 }  
 }  
 });  
 buttonEqually.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 Calc c = new Calc(textInput.getText());  
 result.add(c);  
 refreshCalcList();  
 textInput.setText(c.getRecord());  
 labelResult.setText(c.getResult());  
  
 }  
 });  
 buttonE.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "e");  
 }  
 });  
 buttonPercent.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "%");  
 }  
 });  
 buttonDivide.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "/");  
 }  
 });  
 buttonMultiply.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "x");  
 }  
 });  
 buttonSubtract.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "-");  
 }  
 });  
 buttonAdd.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "+");  
 }  
 });  
 buttonSeparator.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + ".");  
 }  
 });  
 button0.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "0");  
 }  
 });  
 button1.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "1");  
 }  
 });  
 button2.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "2");  
 }  
 });  
 button3.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "3");  
 }  
 });  
 button4.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "4");  
 }  
 });  
 button5.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "5");  
 }  
 });  
 button6.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "6");  
 }  
 });  
 button7.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "7");  
 }  
 });  
 button8.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "8");  
 }  
 });  
 button9.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 textInput.setText(textInput.getText() + "9");  
 }  
 });  
 }  
 public void refreshCalcList() {  
 listCalcModel.removeAllElements();  
 System.*out*.println("Removing all people from list");  
 for (Calc c : result) {  
 System.*out*.println("Adding record of calculation: " + c.getRecord());  
 listCalcModel.addElement(c.getRecord());  
 }  
 }  
 public void addResult(Calc c){  
 result.add(c);  
 refreshCalcList();  
 }  
 public static void main(String[] args) {  
 CalcForm calcForm = new CalcForm();  
 calcForm.setVisible(true);  
  
 Calc first = new Calc("1+2+33x4/5xe");  
 Calc second = new Calc("1+2x15%");  
 Calc third = new Calc("5+2f9");  
 Calc fourth = new Calc("5+2x8%6");  
 calcForm.addResult(first);  
 calcForm.addResult(second);  
 calcForm.addResult(third);  
 calcForm.addResult(fourth);  
// myForm.setLocationRelativeTo(null);  
 }  
}

import java.util.ArrayList;  
  
public class Calc {  
 private String record;  
 private char[] signs;  
 private boolean flag;  
 private double result;  
 private final double e = 2.7;   
 private String first = "";  
 private String second = "";  
 private String operand = "";  
 Calc(){}  
 Calc(String record){  
 this.record = record;  
 flag = tryCalc(record);  
 }  
 private boolean tryCalc(String record){  
 for (int i = 0; i<(record.length()); i++){  
 if ((record.charAt(i) >= '0' && record.charAt(i) <= '9') || record.charAt(i) == 'e' || record.charAt(i) == '.') {  
 if (record.charAt(i) == 'e'){  
 if (!operand.equals("") && !second.equals("")) {  
 second = String.*valueOf*(Double.*parseDouble*(second) \* e);  
 }  
 else if (!operand.equals(""))  
 second = second + e;  
 else {  
 first = first + e;  
 }  
 }  
 else{  
 if (!operand.equals("")) {  
 second = second + record.charAt(i);  
 } else {  
 first = first + record.charAt(i);  
 }  
 }  
// display.setText(first + operand + second);  
 }  
 else if (record.charAt(i) == 'C') {  
 first = operand = second = "";  
// display.setText(first + operand + second);  
 }  
 else {  
 if (operand.equals("") || second.equals(""))  
 if (first.isEmpty()) {  
 first = String.*valueOf*(record.charAt(i));  
 } else if (record.charAt(i) == '%'){  
 result = (Double.*parseDouble*(first)/100);  
 if ((i+1)<(record.length()))  
 if ((record.charAt(i+1) >= '0' && record.charAt(i+1) <= '9') && (record.charAt(i+1) == 'e'))  
 operand = "x";  
 }  
 else {  
 operand = String.*valueOf*(record.charAt(i));  
 }  
 else {  
 if(!switchOperand()){  
 second = first = operand = "";  
 result = 0;  
 return false;  
 }  
 first = Double.*toString*(result);  
 second = operand = "";  
 i--;  
 }  
 }  
 }  
 if (operand.equals("") || second.equals("")) return true;  
 else {  
 if(!switchOperand()){  
 second = first = operand = "";  
 result = 0;  
 return false;  
 }  
 first = Double.*toString*(result);  
 second = operand = "";  
 }  
 return true;  
 }  
 private boolean switchOperand (){  
 result = switch (operand) {  
 case "+" -> (Double.*parseDouble*(first) + Double.*parseDouble*(second));  
 case "-" -> (Double.*parseDouble*(first) - Double.*parseDouble*(second));  
 case "/" -> (Double.*parseDouble*(first) / Double.*parseDouble*(second));  
 case "x" -> (Double.*parseDouble*(first) \* Double.*parseDouble*(second));  
 case "\*" -> (Double.*parseDouble*(first) \* Double.*parseDouble*(second));  
 default -> 1.79769e+308;  
 };  
 if (result == 1.79769e+308)  
 return false;  
 return true;  
 }  
 public void setRecord(String record){  
 this.record = record;  
 }  
 public boolean getFlag(){  
 return flag;  
 }  
 public String getRecord(){  
 return record;  
 }  
 public String getResult(){  
 if (!flag)  
 return "Ошибка! Неизвестный символ.";  
 else  
 return "= " + result;  
 }  
}

Крестики, нолики.

public class MainClass {  
 public static void main(String[] args) {  
 MainForm gameForm = new MainForm();  
 }  
}

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class MainForm extends JFrame {  
 public MainForm() {  
 setTitle("XO game GUI");  
 setBounds(300, 300, 455, 525);  
 setResizable(false);  
 setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);  
 MainGameField gameField = MainGameField.*getInstance*();  
 // Создаём панель для кнопок  
 JPanel buttonPanel = new JPanel(new GridLayout());  
 add(gameField, BorderLayout.*CENTER*);  
 add(buttonPanel, BorderLayout.*SOUTH*);  
 JButton btnStart = new JButton("Начать новую игру");  
 JButton btnEnd = new JButton("Закончить игру");  
 buttonPanel.add(btnStart);  
 buttonPanel.add(btnEnd);  
 btnEnd.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 System.*exit*(0);  
 }  
 });  
 btnStart.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 System.*out*.println(btnStart.getText());  
 GameSettingsForm gameSettingsForm = new GameSettingsForm();  
 }  
 });  
 setVisible(true);  
 }  
}

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import java.util.Scanner;  
  
public class MainGameField extends JPanel {  
 private static MainGameField *instance* = null;  
 public static final int *FIELD\_SIZE* = 450; // размер поля  
 public final String NOT\_SIGN = "\*"; // начальное значние ячейки поля  
 boolean gameOver = false; // признак окончания игры  
 String gameOverMessage = ""; // сообщение которое появляется при окончании игры  
 static int *linesCount* = 3; // число строк, столбцов  
 int cellSize;  
 int x;  
 int y;  
 boolean nextTurn = false; // Чей ход  
 // 2 игрока  
 Player player1;  
 Player player2;  
 int gameMode = 1; // 1 - против игрока 2 - против компьютера  
 int aiLevel = 0; // уровень AI  
 public String[][] cell;  
 public static synchronized MainGameField getInstance() { // получение экземпляра класса  
 if (*instance* == null)  
 *instance* = new MainGameField();  
 return *instance*;  
 }  
 void startNewGame() { // запуск новой игры  
 gameOver = false;  
 gameOverMessage = "";  
 cellSize = *FIELD\_SIZE* / *linesCount*; // размер одной ячейки  
 cell = new String[*linesCount*][*linesCount*];  
 repaint(); // перерисовка поля  
 for (int i = 0; i < *linesCount*; i++) { // инициализация поля  
 for (int j = 0; j < *linesCount*; j++) {  
 cell[i][j] = NOT\_SIGN;  
 }  
 }  
 setVisible(true);  
 }  
 private MainGameField() {  
 setVisible(false);  
 player1 = new Player("X");  
 player2 = new Player("O");  
 addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 x = e.getX() / cellSize;  
 y = e.getY() / cellSize;  
 System.*out*.println("Mouse clicked on " + e.getX() + " " + e.getY());  
 if (!gameOver) {  
 switch (gameMode) {  
 case 1: {  
 twoPlayersMode();  
 break;  
 }  
 case 2: {  
 modeAgainstAI();  
 break;  
 }  
 }  
 }  
 }  
 });  
 }  
 void twoPlayersMode() {  
 if (player1.isShotReady == 1) {  
 nextTurn = true;  
 player2.isShotReady = 0;  
 System.*out*.println("Player 1 shot!");  
 player1.shot(x,y);  
 }  
 if (player1.win()) {  
 System.*out*.println("Player 1 WIN!!!");  
 gameOver = true;  
 gameOverMessage = "Player 1 WIN!!!";  
 }  
 repaint();  
 if (isFieldFull() && !player1.win() && !player2.win()) {  
 gameOver = true;  
 gameOverMessage = "DRAW!!!";  
 }  
 if (player2.isShotReady == 1) {  
 nextTurn = false;  
 player1.isShotReady = 0;  
 System.*out*.println("Player 2 shot!");  
 player2.shot(x,y);  
 }  
 if (!gameOver) {  
 player2.shot(x, y);  
 }  
 if (player2.win()) {  
 System.*out*.println("Player 2 WIN!!!");  
 gameOver = true;  
 gameOverMessage = "Player 2 WIN!!!";  
 }  
 repaint();  
 if (isFieldFull() && !player2.win() && !player1.win()) {  
 gameOver = true;  
 gameOverMessage = "DRAW!!!";  
 }  
 if (nextTurn) {  
 player1.isShotReady = 0;  
 player2.isShotReady = 1;  
 }  
 else {  
 player1.isShotReady = 1;  
 player2.isShotReady = 0;  
 }  
 }  
 void modeAgainstAI() {  
 Player player = new Player("X");  
 AI ai = new AI("O", aiLevel, player.sign);  
 if (!gameOver) {  
 if (player.shot(x, y)) {  
 if (player.win()) {  
 System.*out*.println("Player WIN!!!");  
 gameOver = true;  
 gameOverMessage = "Player WIN!!!";  
 }  
 if (isFieldFull()) {  
 gameOver = true;  
 gameOverMessage = "THE END!!!";  
 }  
 repaint();  
 if (!gameOver) {  
 ai.shot(x, y);  
 }  
 if (ai.win()) {  
 System.*out*.println("AI WIN!!!");  
 gameOver = true;  
 gameOverMessage = "AI WIN!!!";  
 }  
 repaint();  
 if (isFieldFull() && !ai.win()) {  
 gameOver = true;  
 gameOverMessage = "THE END!!!";  
 }  
 }  
 }  
 }  
 boolean isCellBusy(int x, int y) {  
 if (x < 0 || y < 0 || x > *linesCount* - 1 || y > *linesCount* - 1) {  
 return false;  
 }  
 return cell[x][y] != NOT\_SIGN;  
 }  
 public boolean isFieldFull() {  
 for (int i = 0; i < *linesCount*; i++) {  
 for (int j = 0; j < *linesCount*; j++) {  
 if (cell[i][j] == NOT\_SIGN)  
 return false;  
 }  
 }  
 return true;  
 }  
 // проверка линии на равенство значенией  
 public boolean checkLine(int start\_x, int start\_y, int dx, int dy, String sign) {  
 for (int i = 0; i < *linesCount*; i++) {  
 if (cell[start\_x + i \* dx][start\_y + i \* dy] != sign)  
 return false;  
 }  
 return true;  
 }  
 // проверка победы  
 public boolean checkWin(String sign) {  
 for (int i = 0; i < *linesCount*; i++) {  
 // проверяем строки  
 if (checkLine(i, 0, 0, 1, sign)) return true;  
 // проверяем столбцы  
 if (checkLine(0, i, 1, 0, sign)) return true;  
 }  
 // проверяем диагонали  
 if (checkLine(0, 0, 1, 1, sign)) return true;  
 if (checkLine(0, *linesCount* - 1, 1, -1, sign)) return true;  
 return false;  
 }  
 // метод который занимается отрисовкой всей графикой на форме  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 //рисую линии которые представляют собой сетку  
 for (int i = 0; i <= this.*linesCount*; i++) {  
 g.drawLine(0, i \* this.cellSize, *FIELD\_SIZE*, i \* this.cellSize);  
 g.drawLine(i \* this.cellSize, 0, i \* this.cellSize, *FIELD\_SIZE*);  
 }  
 for (int i = 0; i < *linesCount*; i++) {  
 for (int j = 0; j < *linesCount*; j++) {  
 if (cell[i][j] != NOT\_SIGN) {  
 if (cell[i][j] == "X") {  
 // Рисуем крестик  
 g.setColor(Color.*RED*);  
 g.drawLine((i \* cellSize), (j \* cellSize), (i + 1) \* cellSize, (j + 1) \* cellSize);  
 g.drawLine((i + 1) \* cellSize, (j \* cellSize), (i \* cellSize), (j + 1) \* cellSize);  
 }  
 if (cell[i][j] == "O" || cell[i][j] == "O") {  
 // рисуем нолик  
 g.setColor(Color.*BLUE*);  
 g.drawOval((i \* cellSize), (j \* cellSize), cellSize, cellSize);  
 }  
 }  
 }  
 }  
 if (gameOver) {  
 // Отрисовка сообщения при завершении игры  
 g.setColor(Color.*BLACK*);  
 g.fillRect(0, *FIELD\_SIZE* / 2, *FIELD\_SIZE*, *FIELD\_SIZE* / 8);  
 g.setColor(Color.*RED*);  
 g.setFont(new Font("Tahoma", 10, 40));  
 g.drawString(gameOverMessage, 0, 19 \* *FIELD\_SIZE* / 32);  
 }  
 }  
}

public abstract class AGamer {  
 protected String sign;  
  
 abstract boolean shot(int x, int y);  
 abstract boolean win();  
}

import java.util.Random;  
  
public class AI extends AGamer {  
 MainGameField gameField;  
 String playerSign = "";  
 static int *aiLevel* = 0;  
 public AI(String sign, int aiLevel, String playerSign) {  
 this.sign = sign;  
 this.playerSign = playerSign;  
 this.*aiLevel* = aiLevel;  
 }  
 boolean shot(int x, int y) {  
 gameField = MainGameField.*getInstance*();  
 x = -1;  
 y = -1;  
 boolean ai\_win = false;  
 boolean user\_win = false;  
 // Находим выигрышный ход  
 if (*aiLevel* == 2) {  
 for (int i = 0; i < gameField.*linesCount*; i++) {  
 for (int j = 0; j < gameField.*linesCount*; j++) {  
 if (!gameField.isCellBusy(i, j)) {  
 gameField.cell[i][j] = sign;  
 if (gameField.checkWin(sign)) {  
 x = i;  
 y = j;  
 ai\_win = true;  
 }  
 gameField.cell[i][j] = gameField.NOT\_SIGN;  
 }  
 }  
 }  
 }  
 // Блокировка хода пользователя, если он побеждает на следующем ходу  
 if (*aiLevel* > 0) {  
 if (!ai\_win) {  
 for (int i = 0; i < gameField.*linesCount*; i++) {  
 for (int j = 0; j < gameField.*linesCount*; j++) {  
 if (!gameField.isCellBusy(i, j)) {  
 gameField.cell[i][j] = this.playerSign;  
 if (gameField.checkWin(this.playerSign)) {  
 x = i;  
 y = j;  
 user\_win = true;  
 }  
 gameField.cell[i][j] = gameField.NOT\_SIGN;  
 }  
 }  
 }  
 }  
 }  
 if (!ai\_win && !user\_win) {  
 do {  
 Random rnd = new Random();  
 x = rnd.nextInt(gameField.*linesCount*);  
 y = rnd.nextInt(gameField.*linesCount*);  
 }  
 while (gameField.isCellBusy(x, y));  
 }  
 gameField.cell[x][y] = sign;  
 return true;  
 }  
 boolean win() {  
 gameField = MainGameField.*getInstance*();  
 return gameField.checkWin(this.sign);  
 }  
}

public class Player extends AGamer{  
 MainGameField gameField;  
 int isShotReady = 1;  
 public Player(String sign) {  
 this.sign = sign;  
 }  
 boolean shot(int x, int y) {  
 gameField = MainGameField.*getInstance*();  
 if (!gameField.isCellBusy(x, y)) {  
 gameField.cell[x][y] = sign;  
 return true;  
 }  
 return false;  
 }  
 boolean win() {  
 gameField = MainGameField.*getInstance*();  
 return gameField.checkWin(this.sign);  
 }  
}

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;

public class GameSettingsForm extends JFrame {  
 GameSettingsForm gameSettingsForm = this;  
 public GameSettingsForm() {  
 setTitle("Настройки игры");  
 setBounds(450, 450, 240, 190);  
 setResizable(false);  
 setDefaultCloseOperation(WindowConstants.*DISPOSE\_ON\_CLOSE*);  
 setLayout(new BoxLayout(getContentPane(), BoxLayout.*Y\_AXIS*));  
 JLabel jLabelMode = new JLabel("Выберете режим игры:");  
 add(jLabelMode);  
 JRadioButton radioButtonModeTwoPlayers = new JRadioButton("Игрок против игрока");  
 add(radioButtonModeTwoPlayers);  
 radioButtonModeTwoPlayers.setSelected(true);  
 JRadioButton radioButtonModeAgainstAI = new JRadioButton("Игрок против компьютера");  
 add(radioButtonModeAgainstAI);  
 JLabel jLabelAILevel = new JLabel("Выберете уровень AI:");  
 add(jLabelAILevel);  
 JSlider jSlider = new JSlider(0,2,0);  
 add(jSlider);  
 jSlider.setAlignmentX(*LEFT\_ALIGNMENT*);  
 jSlider.setVisible(false);  
 ButtonGroup buttonGroup = new ButtonGroup();  
 buttonGroup.add(radioButtonModeTwoPlayers);  
 buttonGroup.add(radioButtonModeAgainstAI);  
 JLabel jLabelLinesCount = new JLabel("Размер поля (по умолчанию 3 на 3): ");  
 add(jLabelLinesCount);  
 JTextField jTextFieldLinesCount = new JTextField();  
 jTextFieldLinesCount.setMaximumSize(new Dimension(100, 20));  
 add(jTextFieldLinesCount);  
 JButton jButtonSetSettings = new JButton("Начать игру!");  
 add(jButtonSetSettings);  
 setVisible(true);  
 radioButtonModeAgainstAI.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (radioButtonModeAgainstAI.isSelected()) {  
 jSlider.setVisible(true);  
 }  
 }  
 });  
 radioButtonModeTwoPlayers.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (radioButtonModeTwoPlayers.isSelected()) {  
 jSlider.setVisible(false);  
 }  
 }  
 });  
 jButtonSetSettings.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 MainGameField gameField = MainGameField.*getInstance*();  
 if (jTextFieldLinesCount.getText().isEmpty()) {  
 gameField.*linesCount* = 3;  
 }  
 else {  
 try {  
 gameField.*linesCount* = Integer.*parseInt*(jTextFieldLinesCount.getText());  
 }  
 catch (NumberFormatException ex) {  
 System.*out*.println("Необходимо ввести целое число!");  
 }  
 }  
 gameField.startNewGame();  
 if (radioButtonModeAgainstAI.isSelected()) {  
 gameField.gameMode = 2;  
 }  
 gameField.aiLevel = jSlider.getValue();  
 gameSettingsForm.setVisible(false);  
 }  
 });  
 }  
}

Практическая работа 7

1. String text = “Вася заработал 5000 рублей, Петя - 7563 рубля, а Маша - 30000 рублей”

Напишите код, который считает сумму заработка Васи и Маши.

Использование регулярных выражений в данном задании не допускается.

2. Напишите программу, которая на входе через консоль принимает фамилию, имя и отчество одной строкой (например, «Иванов Сергей Петрович») и выводит фамилию, имя и отчество отдельно в формате:

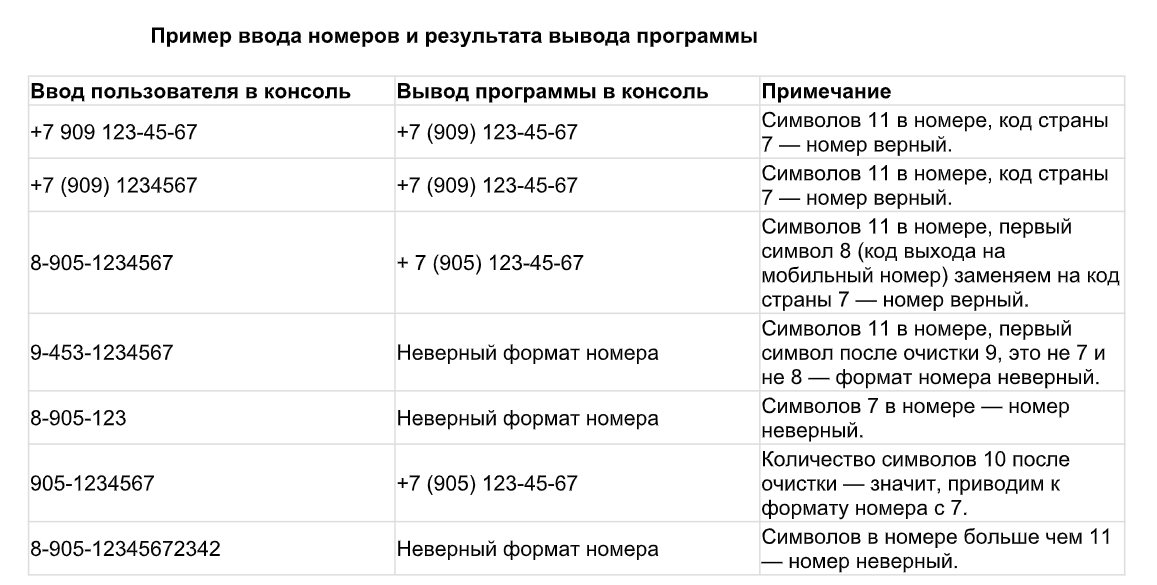
Фамилия: Иванов

Имя: Сергей

Отчество: Петрович

Валидная строка от пользователя, которую мы можем интерпретировать как Ф. И. О., должна содержать три слова, состоящих из символов кириллицы, разделённых пробелом, может содержать дефис. Если у человека отсутствует отчество, то выводить ФИ. Если строка не соответствует формату, то вывести в консоль: Введенная строка не является ФИО

3. Реализуйте удаление лишних символов при вводе номера телефона в консоли и проверку соответствия номера формату мобильных номеров России. Если введённую строку нельзя привести к формату мобильного номера — выводите сообщение о неверном вводе. Телефон может быть введён не только в формате 79091234567, но и с лишними символами.



public class Loader {  
 public static void main(String[] args) {  
// Scanner sc = new Scanner(System.in);  
// String a = sc.nextLine();  
 String text = "Вася заработал 5000 рублей, " + "Петя - 7563 рубля, Маша - 3000 рублей";  
 int f = text.indexOf(' ');  
 int r;  
 String name = text.substring(0, f);  
 int V = 0;  
 int M = 0;  
 int i = 0;  
 int flag = 0;  
 while (i < f || flag < 2){  
 if (name.equals("Вася")) {  
 r = text.indexOf(' ', f+1);  
 name = text.substring(f + 1, r);  
 f = text.indexOf(' ', f+1);  
 while (!*tryCatch*(name)) {  
 r = text.indexOf(' ', f+1);  
 name = text.substring(f + 1, r);  
 f = text.indexOf(' ', f+1);  
 }  
 V = Integer.*parseInt*(name);  
 flag += 1;  
 } else if (name.equals("Маша")) {  
 r = text.indexOf(' ', f+1);  
 name = text.substring(f + 1, r);  
 f = text.indexOf(' ', f+1);  
 while (!*tryCatch*(name)) {  
 r = text.indexOf(' ', f+1);  
 name = text.substring(f + 1, r);  
 f = text.indexOf(' ', f+1);  
 }  
 M = Integer.*parseInt*(name);  
 flag += 1;  
 }  
 if (i < f || flag < 2){  
 r = text.indexOf(' ', f+1);  
 name = text.substring(f + 1, r);  
 f = text.indexOf(' ', f+1);  
 }  
  
 i += f;  
 }  
 int sum = M + V;  
 System.*out*.println("Суммарная зарплата: "+ sum);  
 }  
 public static boolean tryCatch(String name) {  
 try {  
 Integer.*parseInt*(name);  
 return true;  
 } catch (NumberFormatException e) {  
 return false;  
 }  
 }  
}

import java.util.Scanner;  
  
public class Loader2 {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("Введите ФИО через пробел.");  
 String text = sc.nextLine();  
 String surname = null;  
 String name = null;  
 String patronymic = null;  
 int f;  
 int r;  
 boolean flag = false;  
 int i = 0;  
 while (!flag && i < 3){  
 text.trim();  
 f = text.indexOf(' ');  
 if (f > -1 && f <= text.length()){  
 surname = text.substring(0, f);  
 i++;  
 r = text.indexOf(' ', f+1);  
 if (r > -1 && f <= text.length()){  
 name = text.substring(f+1, r);  
 f = text.indexOf(' ', f+1);  
 i++;  
 r = text.indexOf(' ', f+1);  
 if (r == -1 && f <= text.length()) {  
 patronymic = text.substring(f+1, text.length());  
 flag = true;  
 System.*out*.println("Фамилия: "+ surname + "\nИмя: " + name + "\nОтчество: " + patronymic);  
 }  
 }  
 else if (r == -1 && f <= text.length()){  
 name = text.substring(f+1, text.length());  
 i++;  
 flag = true;  
 System.*out*.println("Фамилия: "+ surname + "\nИмя: " + name);  
 }  
 else {  
 System.*out*.println("Введенная строка не является ФИО");  
 text = sc.next();  
 i = 0;  
 }  
 }  
 else {  
 System.*out*.println("Введенная строка не является ФИО");  
 text = sc.next();  
 i = 0;  
 }  
 }  
 }  
}

import java.util.Scanner;  
public class Loader3 {  
 public static void main(String[] args) {  
 // 89091234567  
 // +7-909-123-45-67  
 String number;  
 StringBuffer numberF = new StringBuffer();  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("Введите новер телефона: ");  
 number = sc.nextLine();  
 boolean f = false;  
 while (!f){  
 number = number.replaceAll("[^0**-**9]", "");  
 if (number.length() == 11 || number.length() == 10) {  
 if (number.charAt(0) == '7' && number.length() == 11) {  
 numberF.append(number);  
 numberF.insert(0, "+");  
 numberF.insert(2, " (");  
 numberF.insert(7, ") ");  
 numberF.insert(12, "-");  
 numberF.insert(15, "-");  
 f = true;  
 System.*out*.print(numberF);  
 } else if (number.charAt(0) == '8' && number.length() == 11) {  
 numberF.append(number);  
 numberF.delete(0,1);  
 numberF.insert(0, "+7");  
 numberF.insert(2, " (");  
 numberF.insert(7, ") ");  
 numberF.insert(12, "-");  
 numberF.insert(15, "-");  
 f = true;  
 System.*out*.print(numberF);  
 }  
 else if (number.charAt(0) != '7' && number.length() == 10) {  
 numberF.append(number);  
 numberF.insert(0, "+7");  
 numberF.insert(2, " (");  
 numberF.insert(7, ") ");  
 numberF.insert(12, "-");  
 numberF.insert(15, "-");  
 f = true;  
 System.*out*.print(numberF);  
 }  
 else {  
 System.*out*.println("Неверный ввод.");  
 System.*out*.println("Попробуйте еще раз:");  
 number = sc.next();  
 numberF.append(number);  
 numberF.delete(0, numberF.length());  
 }  
 } else {  
 System.*out*.println("Неверный ввод.");  
 System.*out*.println("Попробуйте еще раз:");  
 number = sc.next();  
 numberF.delete(0, numberF.length());  
 }  
 }  
 }  
}

Практическая работа 8

Пример разобранный на паре.

import java.util.\*;  
  
public class Loader {  
 public static void main(String[] args) {  
 ArrayList<String> todoList = new ArrayList<>();  
 todoList.add("Первое дело");  
 todoList.add(1, "Второе дело");  
 for (int i = 0; i < todoList.size(); i++) {  
 System.*out*.println(todoList.get(i));  
 }  
 System.*out*.println();  
 for (String item : todoList) {  
 System.*out*.println(item);  
 }  
 HashSet<String> words = new HashSet<>();  
 words.add("Первое");  
 words.add("Второе");  
 words.add("Третье");  
 words.add("Первое");  
// words.contains(); //поиск элемента в массиве  
 for (String item : words) {  
 System.*out*.println(item);  
 }  
 HashMap<String, Integer> good2count = new HashMap<>();  
 Scanner scanner = new Scanner(System.*in*);  
 for (; ; ) {  
 String goodName = scanner.nextLine();  
 if (goodName.equals("List")) {  
 *printMap*(good2count);  
 continue;  
 }  
 int count = 1;  
 if (good2count.containsKey(goodName))  
 count += good2count.get(goodName);  
 good2count.put(goodName, count);  
 }  
 }  
 private static void printMap (Map<String, Integer> map){  
 for (String key : map.keySet()) {  
 System.*out*.println(key + "=>" + map.get(key));  
 }  
 }  
}

1. Разработайте список дел с помощью ArrayList, который управляется командами в консоли. Команды: LIST, ADD, EDIT, DELETE.

Принцип работы команд:

 LIST — выводит дела с их порядковыми номерами;

 ADD — добавляет дело в конец списка или дело на определённое место, сдвигая остальные дела вперёд, если указать номер;

 EDIT — заменяет дело с указанным номером;

 DELETE — удаляет.

Примеры команд

 LIST

 ADD Какое-то дело

 ADD 4 Какое-то дело на четвёртом месте

 EDIT 3 Новое название дела

 DELETE 7

Команды вводятся одной строкой пользователем в консоль.

import java.util.ArrayList;   
import java.util.Scanner;  
  
public class caseList1 {  
 public static void main(String[] args) {  
 ArrayList<String> todoList = new ArrayList<>();  
 Scanner sc = new Scanner(System.*in*);  
 String order;  
 int f;  
 int r;  
 int k = 0;  
 String name = "";  
 boolean flag = false;  
 while (!flag) {  
 order = sc.nextLine();  
 f = order.indexOf(' ');  
 if (f != -1){  
 name = order.substring(0, f);  
 if (name.equals("ADD")){  
 r = order.indexOf(' ', f+1 );  
 if (r == -1){  
 name = order.substring(f+1, order.length()); // без ключа  
 todoList.add(name);  
 }  
 else if (*tryCatch*(order.substring(f+1, r))){ // по ключу  
 k = Integer.*parseInt*(order.substring(f+1, r));  
 name = order.substring(r+1, order.length());  
 if (todoList.size() >= k){  
 todoList.add(k, name);  
 }  
 else {  
 todoList.add(name);  
 }  
 }  
 }  
 else if (name.equals("EDIT")){  
 r = order.indexOf(' ');  
 if (r == -1){  
 System.*out*.println("Неправильный ввод ключа! Попробуйте еще раз.");  
 f = 0;  
 k = 0;  
 name = "";  
 }  
 else if (*tryCatch*(order.substring(f+1, r))){ // изменить по ключу  
 k = Integer.*parseInt*(order.substring(f+1, r));  
 if (k <= todoList.size()){  
 name = order.substring(r+1, order.length());  
 todoList.add(k, name);  
 }  
 else {  
 System.*out*.println("Такого ключа не существует");  
 }  
 }  
 else {  
 System.*out*.println("Неправильный ввод ключа! Попробуйте еще раз.");  
 f = 0;  
 k = 0;  
 name = "";  
 }  
 }  
 else if (name.equals("DELETE")){  
 if (*tryCatch*(order.substring(f+1, order.length()))){ //удалить по ключу  
 k = Integer.*parseInt*(order.substring(f+1, order.length()));  
 if (k <= todoList.size()){  
 todoList.remove(k);  
 }  
 }  
 else { // удалить всё  
 todoList.clear();  
 }  
 }  
 }  
 else{  
 if (order.equals("LIST")){  
 *printList*(todoList);  
 }  
 else{  
 System.*out*.println("Такой команды не существует ");  
 }  
 }  
 }  
 }  
 private static void printList (ArrayList<String> todoList){  
 for (int i = 0; i < todoList.size(); i++) {  
 System.*out*.println(i+1 + " - " + todoList.get(i));  
 }  
 }  
 public static boolean tryCatch(String name) {  
 try {  
 Integer.*parseInt*(name);  
 return true;  
 } catch (NumberFormatException e) {  
 return false;  
 }  
 }  
}

2. Напишите программу используя TreeSet, в которой будет храниться перечень адресов электронной почты. Адреса можно добавлять через консоль командой ADD и печатать весь список командой LIST.

Программа должна проверять корректность вводимых email’ов и печатать сообщение об ошибке при необходимости.

Принцип работы команд

 LIST — выводит список электронных адресов.

 ADD — проверяет и, если формат адреса верный, добавляет в множество.

Примеры команд

 LIST

 ADD hello@skillbox.ru

Команды вводятся одной строкой пользователем в консоль.

import java.util.Scanner;  
import java.util.TreeSet;  
import java.util.regex.Pattern;  
  
public class caseTreeSet {  
// kitill.sakh@gmail.com  
// Kirill.sakh@mail.ru  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);   
 TreeSet<String> mailAddress = new TreeSet<>();  
 String[] s;   
 while (true) {  
 s = sc.nextLine().split("\\s+");  
 switch (s[0]) {  
 case ("ADD") -> {  
 if (s.length > 1) {  
 if (*isMailCorrect*(s[1])) {  
 mailAddress.add(s[1]);  
 }  
 System.*out*.println("Incorrect input");  
 } else {  
 System.*out*.println("Incorrect input");  
 }  
 break;  
 }  
 case ("LIST") -> {  
 *printList*(mailAddress);  
 break;  
 }  
 default -> System.*out*.println("Incorrect command");  
 }  
 }  
 }  
   
 static boolean isMailCorrect(String s) {  
 return Pattern.*matches*("\\w+([\\.-]?\\w+)\*@\\w+([\\.-]?\\w+)\*\\.\\w{2,4}", s);  
 }  
   
 private static void printList (TreeSet<String> todoList){  
 for (String s : todoList) {  
 System.*out*.println(s);  
 }  
 }  
}

3. Напишите программу используя коллекцию Map, которая будет работать как телефонная книга:

 Если пишем новое имя, программа просит ввести номер телефона и запоминает его. Если новый номер телефона — просит ввести имя и также запоминает. Если вводим существующее имя или номер телефона, программа выводит всю информацию о контакте. При вводе команды LIST программа печатает в консоль список всех абонентов в алфавитном порядке с номерами.

Определяйте имя и телефон с помощью регулярных выражений.

Подумайте, что выбрать в качестве ключа и значения для Map, выберите лучший вариант по вашему мнению. Опишите, какие минусы и плюсы видите в вашем выборе.

import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Scanner;  
  
public class caseList {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 HashMap<String, String> todoList = new HashMap<>();  
 String order;  
 String number;  
 int f;  
 int r;  
 int k = 0;  
 String name = "";  
 boolean flag = false;  
 while (!flag){  
 order = sc.nextLine();  
 f = order.indexOf(' ');  
 if (f = -1){ number = order.replaceAll("[^0**-**9]", "");  
 if (order.equals("LIST")){  
 *printMap*(todoList);  
 }  
 else if (number.length() > 2){ //если введено число  
 number = *genNumber*(number); // приводим номер к нормальному виду  
 if (todoList.containsValue(number)){ // если такой номер есть  
 System.*out*.println("Такой контакт уже существует:");  
 *printNode*(todoList, number);  
 }  
 else {  
 System.*out*.println("Введите имя соответствующее номеру: ");  
 order = sc.nextLine();  
 todoList.put(order, number);  
 }  
 }  
 else { // если введено слово  
 if (todoList.containsKey(order)){ // если такое слово есть  
 System.*out*.println("Такой контакт уже существует:");  
 *printNode*(todoList, order);  
 }  
 else if (order.length() > 2){  
 System.*out*.println("Введите контактный номер телефона: ");  
 number = *genNumber*(number = sc.nextLine());  
 todoList.put(order, number);  
 }  
 else {  
 System.*out*.println("Попробуйте снова");  
 }  
 }  
 }  
 }  
 }  
 private static String genNumber(String number) {  
 StringBuffer numberF = new StringBuffer();  
 Scanner sc = new Scanner(System.*in*);  
 boolean f = false;  
 while (!f) {  
 number = number.replaceAll("[^0**-**9]", "");  
 if (number.length() == 11 || number.length() == 10) {  
 if (number.charAt(0) == '7' && number.length() == 11) {  
 numberF.append(number);  
 numberF.insert(0, "+");  
 numberF.insert(2, " (");  
 numberF.insert(7, ") ");  
 numberF.insert(12, "-");  
 numberF.insert(15, "-");  
 f = true;  
 System.*out*.println(numberF);  
 return numberF.toString();  
 } else if (number.charAt(0) == '8' && number.length() == 11) {  
 numberF.append(number);  
 numberF.delete(0, 1);  
 numberF.insert(0, "+7");  
 numberF.insert(2, " (");  
 numberF.insert(7, ") ");  
 numberF.insert(12, "-");  
 numberF.insert(15, "-");  
 f = true;  
 System.*out*.println(numberF);  
 return numberF.toString();  
 } else if (number.charAt(0) != '7' && number.length() == 10) {  
 numberF.append(number);  
 numberF.insert(0, "+7");  
 numberF.insert(2, " (");  
 numberF.insert(7, ") ");  
 numberF.insert(12, "-");  
 numberF.insert(15, "-");  
 f = true;  
 System.*out*.println(numberF);  
 return numberF.toString();  
 } else {  
 System.*out*.println("Неверный ввод.");  
 System.*out*.println("Попробуйте еще раз:");  
 number = sc.next();  
 numberF.append(number);  
 numberF.delete(0, numberF.length());  
 }  
 } else {  
 System.*out*.println("Неверный ввод.");  
 System.*out*.println("Попробуйте еще раз:");  
 number = sc.next();  
 numberF.delete(0, numberF.length());  
 }  
 }  
 return "F";  
 }  
 private static int getKey(String order){  
 int k = 0;  
 for (int i = 0; i < order.length(); i++){  
 k += (int)order.charAt(i);  
 }  
 return k;  
 }  
 private static void printNode(Map<String, String> map, String a){  
 for (String key : map.keySet()) {  
 if (map.get(key).equals(a)){  
 System.*out*.println(key + " => " + map.get(key));  
 }  
 else if(key.equals(a)){  
 System.*out*.println(key + " => " + map.get(key));  
 break;  
 }  
 }  
 }  
 private static void printMap (Map<String, String> map){  
 map.entrySet()  
 .stream()  
 .sorted(Map.Entry.<String, String>*comparingByKey*())  
 .forEach(System.*out*::println);  
// for (String key : map.keySet()) {  
// System.out.println(key + " => " + map.get(key));  
// }  
 }  
 public static boolean tryCatch(String name) {  
 try {  
 Integer.*parseInt*(name);  
 return true;  
 } catch (NumberFormatException e) {  
 return false;  
 }  
 }  
}

Практическая работа 9

1. Напишите генератор «красивых» автомобильных номеров

Используйте правила генерации номеров для получения более 2 млн номеров:

X, Y, Z — различные буквы автомобильного номера (&quot;А&quot;, &quot;В&quot;, &quot;Е&quot;, &quot;К&quot;, &quot;М&quot;, &quot;Н&quot;, &quot;О&quot;, &quot;Р&quot;,

&quot;С&quot;, &quot;Т&quot;, &quot;У&quot;, &quot;Х&quot;), N — цифры, R — регион (от 01 до 199);

XNNNYZR — пример, A111BC197, Y777HC66

2. Используя генератор «красивых» методов проведите поиск номера, введенного из консоли, с помощью методов:

 прямым перебором по ArrayList, (array.contains())

 бинарным поиском по сортированному ArrayList, (Collections.binarySearch())

 поиском в HashSet, (setHash.contains())

 поиском в TreeSet. (setTree.contains())

2. Измерьте и сравните длительность каждого метода поиска.

Формат вывода результатов поиска:

Поиск перебором: номер найден/не найден, поиск занял 34нс

Бинарный поиск: номер найден/не найден, поиск занял 34нс

Поиск в HashSet: номер найден/не найден, поиск занял 34нс

Поиск в TreeSet: номер найден/не найден, поиск занял 34нс

Рекомендации

 Сортировка не входит в учёт времени для бинарного поиска.

 Для детального сравнения методов поиска используйте время в наносекундах: System.nanoTime()

 Перед использованием Collections.binarySearch() не забудьте использовать

Collections.sorted() для сортировки основного ArrayList

import java.util.\*;  
import java.util.stream.IntStream;  
import java.util.stream.Stream;  
  
public class randGen {  
 public static void main(String[] args) {  
 ArrayList<String> todoList = new ArrayList<>();  
 String[] chars = new String[]{"A", "B", "E", "K", "M", "H", "O", "P", "C", "T", "Y", "X", };  
 IntStream.*range*(1, 200).forEach(  
 region -> Arrays.*stream*(chars).forEach(  
 letterA -> Arrays.*stream*(chars).forEach(  
 letterB -> Arrays.*stream*(chars).forEach(  
 letterC -> IntStream.*range*(1, 10).forEach(  
 num -> todoList.add(String.*format*("%s%01d%01d%01d%s%s%03d", letterA, num, num , num, letterB, letterC, region))  
 )  
 )  
 )  
 )  
 );  
 todoList.forEach(System.*out*::println);  
 HashSet<String> todoHash = new HashSet<>();  
 TreeSet<String> todoTree = new TreeSet<>();  
 todoHash.addAll(todoList);  
 todoTree.addAll(todoList);  
 long start;  
 long end;  
 Scanner sc = new Scanner(System.*in*);  
 String[] s;  
 int index;  
 boolean check;  
 while (true) {  
 System.*out*.println("If you wont search the license plate withe: contains: \"S C A555BC199\"; binarySearch: \"S B A555BC199\"; hashSet: \"S H A555BC199\"; treeSet: \"S T A555BC199\";\n if you wont escape program write\"GG\" ");  
 s = sc.nextLine().split("\\s+");  
 check = false;  
 switch (s[0]) {  
 case ("S") -> {  
 if (s.length > 2){  
 if (s[1].equals("C")){  
 start = System.*nanoTime*();  
 check = todoList.contains(s[2]);  
 end = System.*nanoTime*();  
 System.*out*.println(String.*format*("Indicator: %b; Time: %d\n", check, (end - start)));  
 }  
 else if (s[1].equals("B")){  
 Collections.*sort*(todoList);  
 start = System.*nanoTime*();  
 index = Collections.*binarySearch*(todoList, s[2]);  
 end = System.*nanoTime*();  
 System.*out*.println(String.*format*("Indicator: %d; Time: %d", index, (end - start)));  
 }  
 else if (s[1].equals("H")){  
 start = System.*nanoTime*();  
 check = todoHash.contains(s[2]);  
 end = System.*nanoTime*();  
 System.*out*.println(String.*format*("Indicator: %b; Time: %d\n", check, (end - start)));  
 }  
 else if (s[1].equals("T")){  
 start = System.*nanoTime*();  
 check = todoTree.contains(s[2]);  
 end = System.*nanoTime*();  
 System.*out*.println(String.*format*("Indicator: %b; Time: %d\n", check, (end - start)));  
 }  
 else  
 System.*out*.println("Incorrect input");  
 } else {  
 System.*out*.println("Incorrect input");  
 }  
 }  
 case ("GG") -> {  
 return;  
 }  
 default -> System.*out*.println("Incorrect command");  
 }  
 }  
 }  
 public static int generateRandomIntIntRange(int min, int max) {  
 Random r = new Random();  
 return r.nextInt((max - min) + 1) + min;  
 }  
  
}

Практическая работа 10

1. Создать статичный метод, который будет логировать сообщения в отдельный файл

«log.txt» в отдельную папку в вашем проекте. Файл и папка должны создаваться при первом запуске вашего проекта.

Данный метод должен быть использован в последующих заданиях в данной практической работе.

import openandread.FileWork;  
  
import java.io.IOException;  
import java.util.Scanner;  
  
public class Loader {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 String[] s;  
 boolean f = true;  
 FileWork fileWork = new FileWork(f);  
 while (f){  
 System.*out*.println("1 - read; 2 - write \"2 ...\"");  
 s = sc.nextLine().split("\\s+");  
 switch (s[0]) {  
 case ("1") -> {  
 fileWork.printFile();  
 }  
 case ("2") -> {  
 StringBuilder stringBuilder = new StringBuilder();  
 for (int i = 1; i < s.length; i++) {  
 stringBuilder.append(s[i]);  
 }  
 fileWork.writeLine(stringBuilder.toString());  
 }  
 default -> System.*out*.println("Incorrect command");  
 }  
 }  
 }  
}

package openandread;  
import java.io.IOException;  
  
public class FileWork {  
 Write line = null;  
 Read new\_file;  
 String[] aryStrings;  
 String new\_path = "D:/log.txt";  
 public FileWork(boolean f){  
 try {  
 line = new Write(new\_path, f);  
 new\_file = new Read(new\_path);  
 aryStrings = new\_file.OpenAndRead();  
 }  
 catch (IOException a) {  
 System.*out*.println(a.getMessage());  
 }  
 }  
 public void writeLine(String mess) {  
 try {  
 line.fileWrite(mess);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 public void printFile(){  
 for (int i = 0; i < aryStrings.length; i++) {  
 System.*out*.println(aryStrings[i]);  
 }  
 }  
}

package openandread;  
  
import java.io.FileWriter;  
import java.io.PrintWriter;  
import java.io.IOException;  
  
public class Write {  
 private String path\_to\_file;  
 private boolean append = false;  
 public Write (String path) {  
 path\_to\_file = path;  
 }  
 public Write (String path, boolean value) {  
 path\_to\_file = path;  
 append = value;  
 }  
 public void fileWrite (String text) throws IOException {  
 FileWriter write\_to\_file = new FileWriter(path\_to\_file, append);  
 PrintWriter print = new PrintWriter(write\_to\_file);  
 print.printf("%s" + "%n", text);  
 print.close();  
 }  
}

package openandread;  
  
import java.io.IOException;  
import java.io.FileReader;  
import java.io.BufferedReader;  
  
public class Read {  
 private String path\_to\_file;  
 public Read(String path) {  
 path\_to\_file = path;  
 }  
 public String[] OpenAndRead() throws IOException {  
 FileReader x = new FileReader(path\_to\_file);  
 BufferedReader varRead = new BufferedReader(x);  
 int num = numStrings();  
 String[] lines = new String[num];  
 int i;  
 for (i = 0; i < num; i++) {  
 lines[i] = varRead.readLine();  
 }  
 varRead.close();  
 return lines;  
 }  
 int numStrings() throws IOException {  
 FileReader text = new FileReader(path\_to\_file);  
 BufferedReader y = new BufferedReader(text);  
 String one;  
 int num = 0;  
 while ((one = y.readLine()) != null) {  
 num++;  
 }  
 y.close();  
 return num;  
 }  
}

2. Напишите программу. Она должна:

 получать через консоль путь от пользователя до папки;

 суммировать размер файлов папки и вложенных папок;

 выводить полученную сумму файлов в удобочитаемом виде — в байтах, килобайтах, мегабайтах или гигабайтах;

Примеры работы программы

Введите путь до папки: D:\Games

Размер папки D:\Games составляет 56,6 Гб

3. Напишите программу, которая:

 Копирует одну указанную папку в другую. При копировании должны сохраниться файлы и структура папки.

 Папки запрашивайте у пользователя в консоли.

import openandread.FileWork;  
import java.io.\*;  
import java.nio.file.Files;  
import java.text.DecimalFormat;  
import java.util.Scanner;  
  
public class Path {  
 public static FileWork *fileWork* = new FileWork(true);  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 String[] s;  
 boolean f = true;  
 while (f){  
 System.*out*.println("1 - Enter the path to the folder \"1 ...\"; 2 - Enter the path from \"2 ... ...\"; 3 - escape");  
 *fileWork*.writeLine("1 - Enter the path to the folder \"1 ...\"; 2 - Enter the path from \"2 ... ...\"; 3 - escape");  
 s = sc.nextLine().split("\\s+");  
 switch (s[0]) {  
 case ("1") -> {  
 StringBuilder stringBuilder = new StringBuilder();  
 for (int i = 1; i < s.length; i++) {  
 stringBuilder.append(s[i]);  
 }  
 // D:\Java\Practical\_N\_1\Practical\_10\src  
 *whenGetReadableSize\_thenCorrect*(stringBuilder.toString());  
 }  
 case ("2") -> {  
 StringBuilder stringBuilder = new StringBuilder();  
 for (int i = 2; i < s.length; i++) {  
 stringBuilder.append(s[i]);  
 }  
 File src = new File(s[1]);  
 File dest = new File(stringBuilder.toString());  
 try {  
 *copyFolder*(src, dest);  
 } catch (IOException e) {  
 System.*out*.println("File exception");  
 *fileWork*.writeLine("File exception");  
 }  
 }  
 case ("3") -> {  
 f = false;  
 System.*out*.println("The end.");  
 *fileWork*.writeLine("The end.");  
 }  
 default -> System.*out*.println("Incorrect command");  
 }  
 }  
 }  
 private static void whenGetReadableSize\_thenCorrect(String s) {  
 File folder = new File(s);  
 long size = *getFolderSize*(folder);  
 String[]units = new String[]{ "B", "KB", "MB", "GB", "TB" };  
 int unitIndex = (int) (Math.*log10*(size)/3);  
 double unitValue = 1 << (unitIndex \* 10);  
 String readableSize = new DecimalFormat("#,##0.#")  
 .format(size/unitValue) + " "  
 + units[unitIndex];  
 System.*out*.println("Размер папки " + s + " составляет " + readableSize);  
 *fileWork*.writeLine("Размер папки " + s + " составляет " + readableSize);  
 }  
  
 public static void copyFolder(File src, File dest) throws IOException{  
 if(src.isDirectory()){  
 //if directory not exists, create it  
 if(!dest.exists()){  
 dest.mkdir();  
 System.*out*.println("Directory copied from "  
 + src + " to " + dest);  
 *fileWork*.writeLine("Directory copied from "  
 + src + " to " + dest);  
 }  
 //list all the directory contents  
 String files[]= src.list();  
 for (String file : files) {  
 //construct the src and dest file structure  
 File srcFile = new File(src, file);  
 File destFile = new File(dest, file);  
 //recursive copy  
 *copyFolder*(srcFile,destFile);  
 }  
 }else{  
 //if file, then copy it  
 //Use bytes stream to support all file types  
 InputStream in = new FileInputStream(src);  
 OutputStream out = new FileOutputStream(dest);  
 byte[]buffer = new byte[1024];  
 int length;  
 //copy the file content in bytes  
 while ((length = in.read(buffer)) > 0){  
 out.write(buffer, 0, length);  
 }  
 in.close();  
 out.close();  
 System.*out*.println("File copied from " + src + " to " + dest);  
 *fileWork*.writeLine("File copied from " + src + " to " + dest);  
 }  
 }  
 private static long getFolderSize(File folder) {  
 long length = 0;  
 File[]files = folder.listFiles();  
 int count = files.length;  
 for (int i = 0; i < count; i++) {  
 if (files[i].isFile()) {  
 length += files[i].length();  
 }  
 else {  
 length += *getFolderSize*(files[i]);  
 }  
 }  
 return length;  
 }  
}

Практическая работа 11

1. Создать новый проект, который будет читать файл csv банковской выписки movementsList.csv и парсить полученные строки. Путь к файлу выписки храните в константе.

Код должен выводить сводную информацию по этой выписке: общий приход, общий расход и разбивку расходов.

|  |
| --- |
| **Примеры работы программы**  Сумма расходов: 398 563.39 руб.  Сумма доходов: 289 890.06 руб.    Суммы расходов по организациям:  RUSMOSKVA56  SHLOVE REPUBLIC        1 081.53 руб.  RUSMOSCOW42 SHCL ETOILE                     126.34 руб.  RUSPUSHKINO105ZOOMAGAZIN 4             217.65 руб. |

import openandread.FileWork;  
  
import java.io.IOException;  
import java.util.Scanner;  
  
public class Loader {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 String[] s;  
 boolean f = true;  
 FileWork fileWork = new FileWork(f);  
 fileWork.setMap();  
  
 while (f){  
 System.*out*.println("1 - print; 2 - exit");  
 s = sc.nextLine().split("\\s+");  
 switch (s[0]) {  
 case ("1") -> {  
 fileWork.printMap();  
 }  
 case ("2") ->{  
 return;  
 }  
 default -> System.*out*.println("Incorrect command");  
 }  
 }  
  
 }  
}

package Company;  
  
public class Company {  
 private String company;  
 private int income = 0;  
 private int expenses = 0;  
 public Company(){}  
 Company(String company, int income, int expenses) {  
 this.company = company;  
 this.income = income;  
 this.expenses = expenses;  
 }  
 public String getCompany() {  
 return company;  
 }  
 public int getIncome() {  
 return income;  
 }  
 public void setCompany(String company) {  
 this.company = company;  
 }  
 public void addIncome(int income) {  
 this.income += income;  
 }  
 public void addExpenses(int expenses) {  
 this.expenses += expenses;  
 }  
 @Override  
 public String toString() {  
 System.*out*.println("Имя компнии:" + company);  
 System.*out*.println("Сумма дооходов: " + income + "; Сумма расходов: " + expenses);  
 return "";  
 }  
}

package openandread;  
import Company.Company;  
import java.io.IOException;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
public class FileWork {  
 int incomeAll = 0;  
 int expenseAll = 0;  
 Write fileWrite = null;  
 Read fileRead;  
 String[] aryStrings;  
 HashMap<String, Company> map = new HashMap<>();;  
 final String new\_path = "movementList.csv";  
 public FileWork(boolean f){  
 try {  
 fileWrite = new Write(new\_path, f);  
 fileRead = new Read(new\_path);  
 aryStrings = fileRead.OpenAndRead();  
 }  
 catch (IOException a) {  
 System.*out*.println(a.getMessage());  
 }  
 }  
 public void setMap(){  
 String[] s;  
 Company company = null;  
 for (String line: aryStrings) {  
 compareToCompany(line);  
 }  
 }  
 public void compareToCompany(String line){  
 Company company = new Company();  
 String[] column = line.split(",");  
 String describe = null;  
 if (column[5].contains("\\"))  
 describe = column[5].substring(column[5].lastIndexOf("\\") + 1, column[5].indexOf(" "));  
 else if (column[5].contains("/"))  
 describe = column[5].substring(column[5].lastIndexOf("/") + 1, column[5].indexOf(" "));  
 column[6] = column[6].replaceAll("\\D", "");  
 column[7] = column[7].replaceAll("\\D", "");  
 System.*out*.println("Приход = " + column[6] + " , расход = " + column[7] + "\nОписание - " + describe + "\n");  
 if (column[6].matches("\\d+") && column[7].matches("\\d+")) {  
 int income = Integer.*parseInt*(column[6]);  
 int expense = Integer.*parseInt*(column[7]);  
 if (income > 0) {  
 if (map.containsKey(describe)) {  
 map.get(describe).addIncome(income);  
 }  
 else {  
 company.setCompany(describe);  
 company.addIncome(income);  
 map.put(describe, company);  
 }  
 incomeAll += income;  
 } else {  
 if (map.containsKey(describe)) {  
 map.get(describe).addExpenses(expense);  
 }  
 else {  
 company.setCompany(describe);  
 company.addExpenses(expense);  
 map.put(describe, company);  
 }  
 expenseAll += expense;  
 }  
 }  
 }  
 public void printMap(){  
 System.*out*.println("Сумма дооходов:" + incomeAll + "\nСумма расходов:" + expenseAll);  
 for(Map.Entry<String, Company> entry: map.entrySet()) {  
 System.*out*.println(entry.getValue());  
 }  
 }  
 public void writeLine(String mess) {  
 try {  
 fileWrite.fileWrite(mess);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 public void printFile(){  
 for (int i = 0; i < aryStrings.length; i++) {  
 System.*out*.println(aryStrings[i]);  
 }  
 }  
}

package openandread;  
  
import java.io.IOException;  
import java.io.FileReader;  
import java.io.BufferedReader;  
public class Read {  
 private String path\_to\_file;  
 public Read(String path) {  
 path\_to\_file = path;  
 }  
 public String[] OpenAndRead() throws IOException {  
 FileReader x = new FileReader(path\_to\_file);  
 BufferedReader varRead = new BufferedReader(x);  
 int num = numStrings();  
 String[] lines = new String[num];  
 int i;  
 for (i = 0; i < num; i++) {  
 lines[i] = varRead.readLine();  
 }  
 varRead.close();  
 return lines;  
 }  
 int numStrings() throws IOException {  
 FileReader text = new FileReader(path\_to\_file);  
 BufferedReader y = new BufferedReader(text);  
 String one;  
 int num = 0;  
 while ((one = y.readLine()) != null) {  
 num++;  
 }  
 y.close();  
 return num;  
 }  
}

package openandread;  
import java.io.FileWriter;  
import java.io.PrintWriter;  
import java.io.IOException;  
  
public class Write {  
 private String path\_to\_file;  
 private boolean append = false;  
 public Write (String path) {  
 path\_to\_file = path;  
 }  
 public Write (String path, boolean value) {  
 path\_to\_file = path;  
 append = value;  
 }  
 public void fileWrite (String text) throws IOException {  
 FileWriter write\_to\_file = new FileWriter(path\_to\_file, append);  
 PrintWriter print = new PrintWriter(write\_to\_file);  
 print.printf("%s" + "%n", text);  
 print.close();  
 }  
}

1. Напишите программу, которая:

* получает с помощью библиотеки jsoup HTML-код страницы https://www.mirea.ru;
* находит в HTML-коде теги **img** и получает абсолютные ссылки на изображения из атрибута **src**;
* скачивает изображения в папку **images** проекта, при этом сохраняя оригинальные названия файлов;
* выводит в консоль список c названиями скачанных файлов.

**Рекомендации**

* cсылка на библиотеку jsoup: https://jsoup.org/
* варианты подключения библиотеки jsoup в проект [на странице скачивания библиотеки](https://jsoup.org/download)
* Для получения ссылок воспользуйтесь префиксом **abs** при получении атрибута **src** у тега **img**. [Пример из официальной документации](https://jsoup.org/cookbook/extracting-data/working-with-urls).

import org.jsoup.Jsoup;  
import org.jsoup.nodes.Document;  
import org.jsoup.select.Elements;  
import java.io.File;  
import java.io.FileOutputStream;  
import java.nio.file.Files;  
import java.nio.file.Path;

public class HTML {  
 public static void main(String[] args) {  
 *getImages*();  
 }  
 public static void getImages() {  
 try {  
 Document html = Jsoup.*connect*("https://www.mirea.ru/").get();  
 System.*out*.println(html.title() + "\n");  
 Elements news = html.getElementsByTag("img");  
 File imgFolder = new File("images");  
 if(!Files.*isDirectory*(Path.*of*("images")))  
 Files.*createDirectory*(Path.*of*("images"));  
 for (var item : news)  
 {  
 if (!item.attr("abs:src").equals("")) {  
 //System.out.println(item.attr("abs:src"));  
 byte[] bytes = Jsoup.*connect*(item.attr("abs:src")).ignoreContentType(true).execute().bodyAsBytes();  
 //System.out.println(bytes);  
 if (!item.attr("src").substring(item.attr("src").lastIndexOf("/") + 1).contains(";")) {  
 FileOutputStream out = new FileOutputStream(new File("images/" + item.attr("src")  
 .substring(item.attr("src")  
 .lastIndexOf("/") + 1)));  
 out.write(bytes);  
 out.close();  
 }  
 }  
 }  
 for (File file: imgFolder.listFiles()) {  
 System.*out*.println(file.toString().substring(file.toString().indexOf("\\") + 1));  
 }  
 } catch (Exception ex) {  
 ex.printStackTrace();  
 }  
 }  
}

Практическая работа 12

**Цель задания**

Научиться получать данные из HTML-страницы, создавать и читать JSON-файлы.

**Что нужно сделать**

Напишите программу, которая:

**1.** Получает HTML-код страницы «Список станций Московского метрополитена» <https://www.moscowmap.ru/metro.html#lines> с помощью библиотеки jsoup.

**2.** Парсит полученную страницу и получает из неё:

* Линии московского метро (получаете имя линии, номер линии).
* Станции московского метро (получаете имя станции, номер линии).

**3.** Создаёт и записывает на диск JSON-файл со списком станций по линиям и списком линий по формату:

{

    "stations" : {

        "1" : [

           "",

           ""

        ],

        "2" : [

           "",

           ""

        ]

    },

    "lines" : [

        {

            "number" : 1,

            "name" : ""

        },

        {

            "number" : 2,

            "name" : ""

        }

    ]

}

**4.**Читает файл и выводит в консоль количество станций на каждой линии.

**Рекомендации**

* По умолчанию Jsoup читает 2 Мб данных с запрашиваемой страницы. Чтобы снять это ограничение, необходимо добавить вызов метода maxBodySize(0), устанавливающий максимальное значение получаемых данных:

Document doc = Jsoup.connect(URL).**maxBodySize(0)**.get();

значение 0 означает, что нет ограничений на принимаемый объём данных.

* При изучении кода страницы удобно использовать консоль разработчика в браузере. Для этого нажмите F12, перейдите во вкладку Elements и найдите тег <div id="metrodata">. В нём содержатся таблицы с линиями, станциями и пересадками. Обращайте внимание на классы, напишите селекторы на основе найденных классов. Посмотрите, как получать элементы по селекторам в [документации JSoup](https://jsoup.org/cookbook/extracting-data/selector-syntax).
* Для более быстрого поиска нужных селекторов используйте онлайн сервис jsoup <https://try.jsoup.org/>
* Ссылка на библиотеку Gson для работы с Json форматом: <https://github.com/google/gson#download>

import com.google.gson.Gson;  
import com.google.gson.GsonBuilder;  
import org.jsoup.Jsoup;  
import org.jsoup.nodes.Document;  
import org.jsoup.nodes.Element;  
import org.jsoup.select.Elements;  
import java.io.File;  
import java.io.FileOutputStream;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Paths;  
import java.util.\*;

public class Loader {  
 static ArrayList<Line> *lines* = new ArrayList<>();  
 static TreeMap<String, String[]> *stationsMap* = new TreeMap<>();  
 static Document *html*;  
 public static void main(String[] args) {  
 *html* = *createHTML*();  
 Elements elLines = *html*.select("span[data-line]");  
 Elements elStations = *html*.select("div[data-line]");   
 for (var item : elLines) {  
 Line line = new Line(item.attr("data-line"), item.text());  
 *lines*.add(line);  
 }  
 for (var line : *lines*) {  
 for (var item : elStations) {  
 if (line.getNumber().equals(item.attr("data-line"))) {  
 String[] stationsOfLine = item.getElementsByTag("p").text().replaceAll("[^а**-**яА**-**ЯёЁъ.\\- ]","").split("\\.");  
 if (stationsOfLine.length - 1 >= 0) System.*arraycopy*(stationsOfLine, 1, stationsOfLine, 0, stationsOfLine.length - 1);  
 String[] stationsOfLineWDeletedSpaces = new String[stationsOfLine.length];  
 int i = 0;  
 for (String station : stationsOfLine) {  
 station = station.substring(1);  
 if (station.charAt((station.length()) - 1) == ' ') {  
 station = station.substring(0, station.length() - 1);  
 }  
 stationsOfLineWDeletedSpaces[i++] = station;  
 }  
 *stationsMap*.put(item.attr("data-line"), stationsOfLineWDeletedSpaces);  
 //System.out.println(" " + item.getElementsByTag("p").text().replaceAll("[^а-яА-ЯёЁъ.\\- ]","")); // value that need to split to array  
 }  
 }  
 }  
 Metro metroObject = new Metro(*stationsMap*,*lines*);  
 Gson json = new GsonBuilder().setPrettyPrinting().create();  
 *writeJsonToFile*(json, metroObject);  
 Metro parsedMetro = json.fromJson(*parseFile*(),Metro.class);  
  
 ArrayList<Line> parsedList = parsedMetro.getLines();  
 TreeMap<String, String[]> parsedMap = parsedMetro.getStations();  
 for (var line : parsedList) {  
 System.*out*.println(line.getName() + ". Количество станций - " + parsedMap.get(line.getNumber()).length);  
 }  
 //System.out.println(json.toJson(metroObject));  
 }  
 private static void writeJsonToFile(Gson json, Metro metroObject) {  
 File dir = new File("data");  
 if (!dir.exists())  
 dir.mkdir();  
 FileWriter jsonFile;  
 try {  
 jsonFile = new FileWriter("data/metro.json");  
 jsonFile.write(json.toJson(metroObject));  
 jsonFile.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 private static Document createHTML() {  
 Document html = null;  
 try {  
 html = Jsoup.*connect*("https://www.moscowmap.ru/metro.html#lines").maxBodySize(0).get();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 // System.out.println(html.title());  
 return html;  
 }  
 public static String parseFile() {  
 StringBuilder builder = new StringBuilder();  
 try {  
 List<String> files = Files.*readAllLines*(Paths.*get*("data/metro.json"));  
 files.forEach(line -> builder.append(line).append("\n"));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return builder.toString();  
 }  
}

public class Line {  
 private String number;  
 private String name;  
 public Line(String number, String name) {  
 this.number = number;  
 this.name = name;  
 }  
 public Line(){}  
 public String getNumber() {  
 return number;  
 }  
 public String getName() {  
 return name;  
 }  
}

import java.util.ArrayList;  
import java.util.TreeMap;  
  
public class Metro{  
 private TreeMap<String, String[]> stations;  
 private ArrayList<Line> lines;  
 public Metro(TreeMap<String, String[]> stations, ArrayList<Line> lines) {  
 this.stations = stations;  
 this.lines = lines;  
 }  
 public TreeMap<String, String[]> getStations() {  
 return stations;  
 }  
 public ArrayList<Line> getLines() {  
 return lines;  
 }  
}

Практическая работа 13

**Цель задания**

Освоить лямбда-выражения.

**Что нужно сделать**

Сделайте сортировку сотрудников одновременно по заработной плате и алфавиту.

Дмитриев Василий 78000 12.07.2016  
Алексей Ивлиев 125000 11.11.2018  
Анна Сетяева 140000 10.05.2012  
Дмитрий Кочергин 140000 31.01.2017  
Михаил Быковский 78000 10.02.2018  
Артём Фигурнов 78000 22.04.2018  
Марина Евстафьева 95000 17.03.2017  
Павел Смородинов 88000 19.09.2018  
Жанна Ковалёва 78000 20.02.2018  
Степан Богданов 140000 17.04.2016  
Виктор Рязанов 95000 20.03.2017  
Савелий Фёдоров 115000 07.05.2019  
Андрей Леонов 88000 16.12.2017  
Татьяна Рыхальская 95000 10.10.2018  
Анастасия Бернадо 78000 05.06.2015  
Евгений Алфёров 105000 12.08.2016

import java.nio.file.Files;  
import java.nio.file.Paths;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.\*;  
import java.util.stream.Stream;  
  
public class Main  
{  
 private static String *staffFile* = "data/staff.txt";  
 private static String *dateFormat* = "dd.MM.yyyy";  
 public static void main(String[] args)  
 {  
 ArrayList<Employee> staff = *loadStaffFromFile*();  
 staff.forEach(System.*out*::println);  
 //staff.sort(((o1, o2) -> {if (o1.getName().equals(o2.getName())) {return Integer.compare(o1.getSalary(), o2.getSalary());} else{return o1.getName().compareTo(o2.getName());}}));  
 staff.sort(  
 Comparator.*comparing*(Employee::getName).thenComparing(Employee::getSalary)  
 );  
 System.*out*.println("\n");  
 staff.forEach(System.*out*::println);  
 }  
 private static ArrayList<Employee> loadStaffFromFile()  
 {  
 ArrayList<Employee> staff = new ArrayList<>();  
 try  
 {  
 List<String> lines = Files.*readAllLines*(Paths.*get*(*staffFile*));  
 for(String line : lines)  
 {  
 String[] fragments = line.split("\t");  
 if(fragments.length != 3) {  
 System.*out*.println("Wrong line: " + line);  
 continue;  
 }  
 staff.add(new Employee(  
 fragments[0],  
 Integer.*parseInt*(fragments[1]),  
 (new SimpleDateFormat(*dateFormat*)).parse(fragments[2])  
 ));  
 }  
 }  
 catch (Exception ex) {  
 ex.printStackTrace();  
 }  
 return staff;  
 }  
}

import java.text.SimpleDateFormat;  
import java.util.Date;  
  
public class Employee  
{  
 private String name;  
 private Integer salary;  
 private Date workStart;  
 public Employee(String name, Integer salary, Date workStart)  
 {  
 this.name = name;  
 this.salary = salary;  
 this.workStart = workStart;  
 }  
 public String getName() {  
 return name;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 public Integer getSalary() {  
 return salary;  
 }  
 public void setSalary(int salary) {  
 this.salary = salary;  
 }  
 public Date getWorkStart() {  
 return workStart;  
 }  
 public void setWorkStart(Date workStart) {  
 this.workStart = workStart;  
 }  
 public String toString()  
 {  
 return name + " - " + salary + " - " +  
 (new SimpleDateFormat("dd.MM.yyyy")).format(workStart);  
 }  
}

Практическая работа 14

**Что нужно сделать**

Напишите программу уменьшения изображений на основе кода, который был рассмотрен на занятии. Она должна запускать число потоков, равное количеству ядер процессора вашего компьютера.

Ссылка на исходник проекта: <https://github.com/MrWoWander/JavaPractica-2020/tree/main/src/p13>

**Рекомендации**:

* Для получения количества ядер используйте метод Runtime.getRuntime().availableProcessors();
* Вместе с данным Word файлом приложены набор картинок, которые вы можете использовать в работе (необязательно)

import java.io.File;  
import java.nio.file.Files;  
import java.nio.file.Paths;  
  
public class Loader {  
 public static void main(String[] args)  
 {  
 String srcFolder = "images";  
 String dstFolder = "dst";  
 File srcDir = new File(srcFolder);  
 long start;  
 File[] files = srcDir.listFiles();  
 try  
 {  
 if (!Files.*exists*(Paths.*get*(dstFolder)))  
 {  
 Files.*createDirectories*(Paths.*get*(dstFolder));  
 }  
 int i = 0;  
 int p = Runtime.*getRuntime*().availableProcessors();  
 int temp = files.length/p;  
 while (i < p){  
 File[] files1 = new File[temp];  
 System.*arraycopy*(files, temp\*i, files1, 0, files1.length);  
 start = System.*currentTimeMillis*();  
 Thread resize = new Thread (new ImageResize(files1, start, dstFolder));  
 resize.start();  
 i++;  
 }  
 }  
 catch (Exception ex) {  
 ex.printStackTrace();  
 }  
 }  
}

import javax.imageio.ImageIO;  
import java.awt.image.BufferedImage;  
import java.io.File;  
import java.nio.file.Files;  
import java.nio.file.Paths;  
  
public class ImageResize implements Runnable{  
 File[] files;  
 long start;  
 String dstFolder;  
 public ImageResize(File[] files, long start, String dstFolder) {  
 this.files = files;  
 this.start = start;  
 this.dstFolder = dstFolder;  
 }  
 @Override  
 public void run() {  
 try  
 {  
 if (!Files.*exists*(Paths.*get*(dstFolder)))  
 {  
 Files.*createDirectories*(Paths.*get*(dstFolder));  
 }  
 for(File file : files)  
 {  
 BufferedImage image = ImageIO.*read*(file);  
 if(image == null) {  
 continue;  
 }  
 int newWidth = image.getWidth() / 2;  
 int newHeight = (int) Math.*round*(  
 image.getHeight() / (image.getWidth() / (double) newWidth)  
 );  
 BufferedImage newImage = new BufferedImage(  
 newWidth, newHeight, BufferedImage.*TYPE\_INT\_RGB* );  
 int widthStep = image.getWidth() / newWidth;  
 int heightStep = image.getHeight() / newHeight;  
 for (int x = 0; x < newWidth; x++)  
 {  
 for (int y = 0; y < newHeight; y++) {  
 int rgb = image.getRGB(x \* widthStep, y \* heightStep);  
 newImage.setRGB(x, y, rgb);  
 }  
 }  
 File newFile = new File(dstFolder + "/" + file.getName());  
 ImageIO.*write*(newImage, "jpg", newFile);  
 }  
 }  
 catch (Exception ex) {  
 ex.printStackTrace();  
 }  
 System.*out*.println("Duration: " + (System.*currentTimeMillis*() - start));  
 }  
}

Практическая работа 15

В банке (класс Bank) есть счета (класс Account) с двумя полями — money и accNumber.

Все счета хранятся внутри банка. Множество клиентов банка могут одновременно переводить деньги между счетами и запрашивать баланс по своему счёту. Всё происходит в highly concurrent (многопоточной) среде.

При этом транзакции на суммы >50000 отправляются на проверку в службу безопасности. Можно считать, что таких транзакций не более 5% от всех. За проверку отвечает отдельный и уже реализованный метод Bank.isFraud().

Служба безопасности не может обрабатывать более одной транзакции одновременно. Проверка занимает 1000 мс.

Если служба безопасности обнаружила мошенничество, необходимо заблокировать оба счёта, то есть запретить любые изменения остатков в дальнейшем.

**Что нужно сделать**

1. Создайте метод transfer() класса Bank, который переводит деньги с одного счёта на другой. Если сумма транзакции >50000 — транзакция отправляется на проверку службе безопасности: вызывается метод isFraud(). Если возвращается true, то счета блокируются (как – на ваше усмотрение).
2. Создайте метод getBalance класса Bank, который возвращает остаток на счёте по переданной строке номера аккаунта.

**Рекомендации**

* Для решения задачи вы можете дорабатывать классы Account и Bank как угодно.
* Проверяйте сумму на банковских счетах до запуска транзакций и после завершения — сумма в банке не должна измениться.
* Удостоверьтесь, что ваша программа работает в многопоточном режиме

import java.util.ArrayList;  
import java.util.Random;  
  
public class Loader {  
 public static void main(String[] args) {  
 Bank bank = new Bank();  
 ArrayList<Thread> threads = new ArrayList<>();  
 for (int j = 0;j < 1000; j++) {  
 Account acc = new Account(new Random().nextInt(100000), j);  
 bank.addAccount(acc);  
 }  
 long sumBefore = 0;  
 for (int i = 0; i < 1000; i++) {  
 sumBefore+=bank.getBalance(i);  
 }  
 for (int i = 0; i < 8; i++) {  
 threads.add(new Thread(() -> {  
 for (int j = 0; j < 500; j++) {  
 Account from = bank.accounts.get(new Random().nextInt(999));  
 Account to = bank.accounts.get(new Random().nextInt(999));  
 try {  
 //System.out.println("Acc " + from.getAccNumber() + " trying transfer money to " + to.getAccNumber());  
 bank.transfer(from.getAccNumber(), to.getAccNumber(), new Random().nextInt(65000));  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }));  
 threads.get(i).start();  
 }  
 for (Thread thread: threads) {  
 try {  
 if (thread != null) {  
 thread.join();  
 }  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 long sumAfter = 0;  
 for (int i = 0; i < 1000; i++) {  
 sumAfter += bank.getBalance(i);  
 }  
 if (sumBefore == sumAfter) System.*out*.println("Суммы сошлись.");  
 else System.*out*.println("Суммы не сошлись!");  
 }  
}

import java.util.HashMap;  
import java.util.Map;  
import java.util.Random;  
import java.util.concurrent.atomic.AtomicLong;  
  
public class Bank {  
 private static AtomicLong *sum* = new AtomicLong();  
 HashMap<Integer, Account> accounts = new HashMap<>();  
 private final Random random = new Random();  
 public int transfer(int accFrom, int accTo, int money) throws InterruptedException{  
 if (accounts.get(accFrom).getStatus() || accounts.get(accTo).getStatus()){  
 System.*out*.println("Одна из карт уже заблокирована.");  
 return 1;  
 }  
 *sum*.lazySet(checkSum(accounts));  
 if (money > 50000){  
 try {  
 if (isFraud(money)){  
 accounts.get(accFrom).blockAccNumber();  
 accounts.get(accTo).blockAccNumber();  
 return 1;  
 }  
 else {  
 if (accounts.get(accFrom).getMoney() - money >= 0){  
 accounts.get(accFrom).subtract(money);  
 accounts.get(accTo).addMoney(money);  
 }  
 else {  
 System.*out*.println("Недоастаточно средств.");  
 return 2;  
 }  
 }  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 else{  
 accounts.get(accFrom).subtract(money);  
 accounts.get(accTo).addMoney(money);  
 }  
 if (*sum*.equals(checkSum(accounts))){  
 System.*out*.println("Сумма не сошлась!");  
 return 3;  
 }  
 return 0;  
 }  
 private synchronized int checkSum (Map<Integer, Account> map){  
 int sum = 0;  
 for (Integer key : map.keySet()) {  
 sum += map.get(key).money;  
 }  
 return sum;  
 }  
  
 public void addAccount(Account account) {  
 accounts.put(account.getAccNumber(), account);  
 }  
  
//  
// public synchronized boolean isFraud(String fromAccountNum, String toAccountNum, long amount)  
// throws InterruptedException  
// {  
// Thread.sleep(1000);  
// return random.nextBoolean();  
// }  
  
 public synchronized boolean isFraud(int money) throws InterruptedException{  
 Thread.*sleep*(1);  
 if (money%50 == 0){  
 return true;  
 }  
 else return false;  
 }  
  
 public int getBalance(int acc){  
 return accounts.get(acc).getMoney();  
 }  
}

public class Account {  
 int money = 0;  
 int accNumber = 0;  
 boolean status = false;  
 public Account(int money, int accNumber) {  
 this.money = money;  
 this.accNumber = accNumber;  
 }  
 public int addMoney(int money){  
 return this.money += money;  
 }  
 public int subtract(int money){  
 return this.money -= money;  
 }  
 public int getMoney() {  
 return money;  
 }  
 public int getAccNumber() {  
 return accNumber;  
 }  
 public boolean getStatus() {  
 return status;  
 }  
 public void blockAccNumber(){  
 status = true;  
 }  
}