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

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКИМ ЗАДАНИЯМ** | |
| **по дисциплине** | |
| **«**Программирование на языке Джава**»** | |
|  | |
| Выполнил студент группы ИКБО-13-22 | Гиль Н. М. |
| Принял преподаватель | Матчин В. Т. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2023

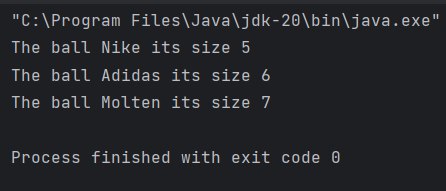
**Практическая работа № 1. Знакомство со средой разработки. Синтаксис и основные управляющие конструкции языка Джава**

**Цель:** введение в разработку программ на языке программирования Джава.

**Код:**

|  |
| --- |
| package Lab1;  public class Ball{  private String brand;  private Integer size;  public Ball(String b, int s){  brand = b;  size = s;  }  public void setBrand(String b){  this.brand = brand;  }  public void setSize(int size){  this.size = size;  }  public String getBrand(String brand){  return brand;  }  public int getSize(){  return size;  }  public String ToString(){  return "The ball " + this.brand + " its size " + this.size;  }  public void ReallyBall(){  System.out.println("The ball " + brand + " its size " + size);  }  }  package MireaLab1;  public class TestBall{  public static void main(String[] args){  Ball b1 = new Ball("Nike",5);  Ball b2 = new Ball("Adidas",6);  Ball b3 = new Ball("Molten",7);  b1.setBrand("NIKE");  b1.ReallyBall();  b2.ReallyBall();  b3.ReallyBall();  }  } |

**Результат:**



**Практическая работа № 2. Объектно-ориентированное программирование в Джава. Классы в Джава**

**Цель** данной практической работы - изучить основные концепции объектно-ориентированного программирования, изучить понятие класса и научиться создавать классы.

Код:

import java.lang.\*;  
public class Author {  
 private String name;  
 private String email;  
 private char gender;  
 public Author(String n, String e, char g){  
 name = n;  
 email = e;  
 gender = g;  
 }  
 public void setEmail(String email){  
 this.email = email;  
 }  
 public String getName(){  
 return name;  
 }  
 public String getEmail(){  
 return email;  
 }  
 public char getGender(){  
 return gender;  
 }  
 public String toString(){  
 return name + "(" + this.getGender() + ") at " + email;  
 }  
}

import java.lang.\*;  
  
public class TestAuthor{  
 public static void main(String[] args){  
 Author a1 = new Author("Gil Nikolay", "ng19022005@gmail.com", 'M');  
 Author a2 = new Author("Anna Ivanova", "fduihdi@gmail.com", 'F');  
 a2.setEmail("fduihdi@gmail.com");  
 System.*out*.println(a1);  
 System.*out*.println(a2.getName());  
 System.*out*.println(a2.getEmail());  
 System.*out*.println(a2.getGender());  
 System.*out*.println(a2);  
 }  
}

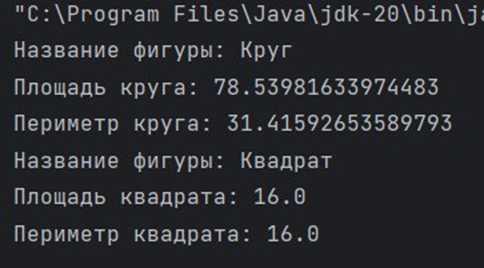
**Практическая работа № 3. Классы Math и Random. Классы оболочки**

**Цель** данной практической работы - изучить работу с классами Math и Random основные концепции объектно-ориентированного программирования, научиться программировать математические вычисления с использованием этих классов, а также познакомиться с классами оболочками и их использованием в Джава программах и научиться форматировать вывод строк.

Код:

|  |
| --- |
| import java.lang.\*;  abstract class Shape {  private String name;  public Shape(String name) {  this.name = name;  }  public String getName() {  return name;  }  // Абстрактные методы  public abstract double area();  public abstract double perimeter();  }  class Circle extends Shape {  private double radius;  public Circle(String name, double radius) {  super(name);  this.radius = radius;  }  @Override  public double area() {  return Math.PI \* Math.pow(radius,2);  }  @Override  public double perimeter() {  return 2 \* Math.PI \* radius;  }  }  class Square extends Shape {  private double sideLength;  public Square(String name, double sideLength) {  super(name);  this.sideLength = sideLength;  }  @Override  public double area() {  return sideLength \* sideLength;  }  @Override  public double perimeter() {  return 4 \* sideLength;  }  }  package MireaLab3;  public class TestShape {  public static void main(String[] args) {  Circle circle = new Circle("Круг", 5.0);  Square square = new Square("Квадрат", 4.0);  System.out.println("Название фигуры: " + circle.getName());  System.out.println("Площадь круга: " + circle.area());  System.out.println("Периметр круга: " + circle.perimeter());  System.out.println("Название фигуры: " + square.getName());  System.out.println("Площадь квадрата: " + square.area());  System.out.println("Периметр квадрата: " + square.perimeter());  }  } |

Результат:



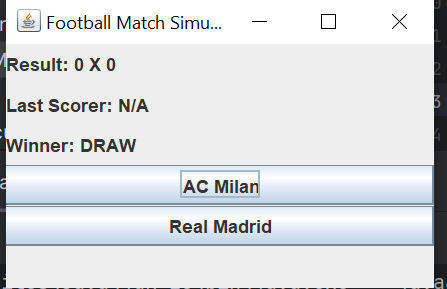
**Практическая работа № 4. Перечисления и их использование в Джава программах**

**Цель** данной практической работы – познакомиться с новым ссылочным типом данных перечислением, научиться разрабатывать перечисления и использовать их в своих программах.

Код:

|  |
| --- |
| import javax.swing.\*;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  public class FootballMatchSimulator extends JFrame {  private int milanScore = 0;  private int madridScore = 0;  private JLabel resultLabel;  private JLabel lastScorerLabel;  private JLabel winnerLabel;  public FootballMatchSimulator() {  setTitle("Football Match Simulator");  setSize(300, 200);  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  setLayout(new GridLayout(6, 3));  JButton milanButton = new JButton("AC Milan");  JButton madridButton = new JButton("Real Madrid");  resultLabel = new JLabel("Result: 0 X 0");  lastScorerLabel = new JLabel("Last Scorer: N/A");  winnerLabel = new JLabel("Winner: DRAW");  milanButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  milanScore++;  updateLabels("AC Milan");  }  });  madridButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  madridScore++;  updateLabels("Real Madrid");  }  });  add(resultLabel);  add(lastScorerLabel);  add(winnerLabel);  add(milanButton);  add(madridButton);  setVisible(true);  }  private void updateLabels(String lastScorer) {  resultLabel.setText("Result: " + milanScore + " X " + madridScore);  lastScorerLabel.setText("Last Scorer: " + lastScorer);  if (milanScore > madridScore) {  winnerLabel.setText("Winner: AC Milan");  } else if (madridScore > milanScore) {  winnerLabel.setText("Winner: Real Madrid");  } else {  winnerLabel.setText("Winner: DRAW");  }  }  public static void main(String[] args) {  SwingUtilities.invokeLater(new Runnable() {  public void run() {  new FootballMatchSimulator();  }  });  }  } |

Результат:



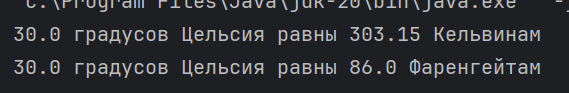
**Практическая работа № 5. Создание программ с графическим интерфейсом пользователя на языке Джава**

**Цель:** цель данной практической работы – научится разрабатывать программы на языке Джава с использованием графического интерфейса пользователя.

Код:

|  |
| --- |
| package MireaLab5;  interface Convertable {  double convertToKelvin(double celsius);  double convertToFahrenheit(double celsius);  }  public class TemperatureConverter implements Convertable {  @Override  public double convertToKelvin(double celsius) {  // Формула для конвертации из Цельсия в Кельвины  return celsius + 273.15;  }  @Override  public double convertToFahrenheit(double celsius) {  // Формула для конвертации из Цельсия в Фаренгейты  return (celsius \* 9/5) + 32;  }  public static void main(String[] args) {  TemperatureConverter converter = new TemperatureConverter();  double celsiusTemperature = 30.0;  double kelvinTemperature = converter.convertToKelvin(celsiusTemperature);  double fahrenheitTemperature = converter.convertToFahrenheit(celsiusTemperature);  System.out.println(celsiusTemperature + " градусов Цельсия равны " + kelvinTemperature + " Кельвинам");  System.out.println(celsiusTemperature + " градусов Цельсия равны " + fahrenheitTemperature + " Фаренгейтам");  }  } |

Результат:



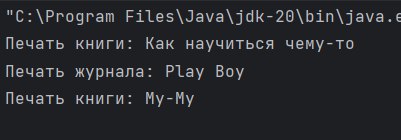
**Практическая работа № 6. Интерфейсы в Java**

**Цель:** цель данной практической работы – научится разрабатывать практике пользовательские интерфейсы, и применять их в программах на языке Джава.

Код:

|  |
| --- |
| package MireaLab6;  // Интерфейс Printable с методом print()  interface Printable {  void print();  }  // Класс Book, реализующий интерфейс Printable  class Book implements Printable {  private String title;  public Book(String title) {  this.title = title;  }  @Override  public void print() {  System.out.println("Печать книги: " + title);  }  }  // Класс Magazine, реализующий интерфейс Printable  class Magazine implements Printable {  private String name;  public Magazine(String name) {  this.name = name;  }  @Override  public void print() {  System.out.println("Печать журнала: " + name);  }  }  public class Main {  public static void main(String[] args) {  // Создаем массив объектов типа Printable  Printable[] printables = new Printable[3];  printables[0] = new Book("Как научиться чему-то");  printables[1] = new Magazine("Play Boy");  printables[2] = new Book("Му-Му");  // Проходим по массиву и вызываем метод print() для каждого объекта  for (Printable printable : printables) {  printable.print();  }  }  } |

Результат:



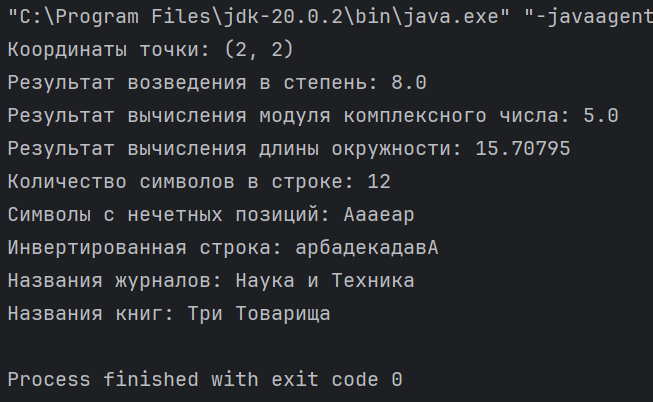
**Практическая работа № 7. Реализация интерфейсов**

**Цель:** цель данной практической работы – научится разрабатывать практике пользовательские интерфейсы, и применять их в программах на языке Джава.

Код:

|  |
| --- |
| package pr7;  public class MovablePoint implements Movable {      int x;      int y;      int speedX;      int speedY;      public MovablePoint(int x, int y, int speedX, int speedY) {          this.x = x;          this.y = y;          this.speedX = speedX;          this.speedY = speedY;      }      public int getSpeedX() {          return speedX;      }      public int getSpeedY() {          return speedY;      }      public void moveUp() {          this.y = speedY;      }      public void moveDown() {          this.y = speedY;      }      public void moveLeft() {          this.x = speedX;      }      public void moveRight() {          this.x = speedX;      }      public String toString() {          return this.x + ", " + this.y;      }  }  package pr7;  public class MovableRectangle implements Movable {      private MovablePoint topLeft = new MovablePoint(0, 1, 0, 0);      private MovablePoint bottomRight = new MovablePoint(1, 0, 0, 0);      public MovableRectangle(int x1, int y1, int x2, int y2, int speedX, int speedY) {          this.topLeft = new MovablePoint(x1, y1, speedX, speedY);          this.bottomRight = new MovablePoint(x2, y2, speedX, speedY);      }      public void moveUp() {          if (SpeedTest(topLeft, bottomRight)) {              this.topLeft.moveUp();              this.bottomRight.moveUp();          }          else {              System.out.println("Error! Points have different speed.");          }      }      public void moveDown() {          if (SpeedTest(topLeft, bottomRight)) {              this.topLeft.moveDown();              this.bottomRight.moveDown();          }          else {              System.out.println("Error! Points have different speed.");          }      }      public void moveLeft() {          if (SpeedTest(topLeft, bottomRight)) {              this.topLeft.moveLeft();              this.bottomRight.moveLeft();          }          else {              System.out.println("Error! Points have different speed.");          }      }      public void moveRight() {          if (SpeedTest(topLeft, bottomRight)) {              this.topLeft.moveRight();              this.bottomRight.moveRight();          }          else {              System.out.println("Error! Points have different speed.");          }      }      public boolean SpeedTest(MovablePoint point1, MovablePoint point2) {          if (point1.getSpeedX() == point2.getSpeedX() && point1.getSpeedY() == point2.getSpeedY()) {              return true;          }          else {              return false;          }      }      public String toString() {          return "Top left poin is at " + topLeft + " and bottom right point is at " + bottomRight;      }  }  package pr7;  interface MathCalculable {      double pow(double base, double exponent);      double mod(double real, double imaginary);      double PI = 3.14159;  }  package pr7;  public class MathTest implements MathCalculable {      @Override      public double pow(double base, double exponent) {          return Math.pow(base, exponent);      }      @Override      public double mod(double real, double imaginary) {          return Math.sqrt(Math.pow(real,2)+Math.pow(imaginary,2));      }      public double lenghtCircle(double diameter){          return PI\*diameter;      }  }  package pr7;  import java.lang.\*;  public interface Movable {      public void moveUp();      public void moveDown();      public void moveLeft();      public void moveRight();  }  package pr7;  public interface Printable {      void print();  }  package pr7;  public class printBooks implements Printable{      private String name;      public printBooks(String name){          this.name = name;      }      @Override      public void print() {          System.out.println("Названия книг: " + name);      }  }  package pr7;  public class ProcessStrings implements Manipulator{      @Override      public int Counter(String s) {          return s.length();      }      @Override      public String getCharacters(String s) {          StringBuilder result = new StringBuilder();          for (int i=0; i<s.length();i++){              if(i%2==0) {                  result.append(s.charAt(i));              }          }          return result.toString();      }      @Override      public String Inverter(String s) {          return new StringBuilder(s).reverse().toString();      }  }  package pr7;  public class printMagazines implements Printable {      private String name;      public printMagazines(String name){          this.name = name;      }      @Override      public void print() {          System.out.println("Названия журналов: " + name);      }  }  package pr7;  public interface Manipulator {      int Counter(String s);      String getCharacters(String s);      String Inverter(String s);  }  package pr7;  public class Main {      public static void main(String[] args) {          MovablePoint point = new MovablePoint(0,0,2,2);          point.moveUp();          point.moveRight();          System.out.println("Координаты точки: (" + point.x + ", " + point.y + ")");          MathTest math = new MathTest();          double result1 = math.pow(2, 3);          double result2 = math.mod(3, 4);          double result3 = math.lenghtCircle(5);          System.out.println("Результат возведения в степень: " + result1);          System.out.println("Результат вычисления модуля комплексного числа: " + result2);          System.out.println("Результат вычисления длины окружности: " + result3);          ProcessStrings processStrings = new ProcessStrings();          String s = "Авадакедабра";          int count = processStrings.Counter(s);          String characters = processStrings.getCharacters(s);          String invertedString = processStrings.Inverter(s);          System.out.println("Количество символов в строке: " + count);          System.out.println("Символы с нечетных позиций: " + characters);          System.out.println("Инвертированная строка: " + invertedString);          Printable[] printables = new Printable[2];          printables[0] = new printMagazines("Наука и Техника");          printables[1] = new printBooks("Три Товарища");          for (Printable printable : printables) {              printable.print();          }      }  } |

Результат:



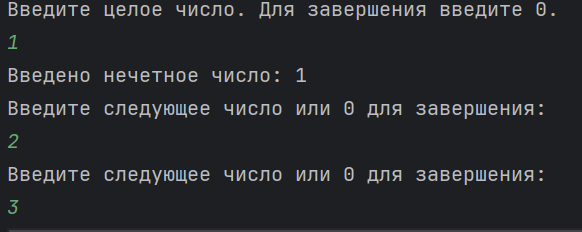
**Практическая работа № 8. Рекурсия Программирование рекурсии в Java. Решение задач на рекурсию**

**Цель:** разработка и программирование рекурсивных алгоритмов на языке Java.

Код:

|  |
| --- |
| import java.util.Scanner;  public class NUMB8 {  public static void main(String[] args) {  System.*out*.println("Введите целое число. Для завершения введите 0.");  *readAndPrintOddNumbers*();  }   public static void readAndPrintOddNumbers() {  Scanner scanner = new Scanner(System.*in*);  int number = scanner.nextInt();   if (number != 0) {  if (number % 2 != 0) {  System.*out*.println("Введено нечетное число: " + number);  }  System.*out*.println("Введите следующее число или 0 для завершения:");  *readAndPrintOddNumbers*(); // Рекурсивный вызов функции для следующего числа  } else {  System.*out*.println("Программа завершена.");  }  } } |

Результат:



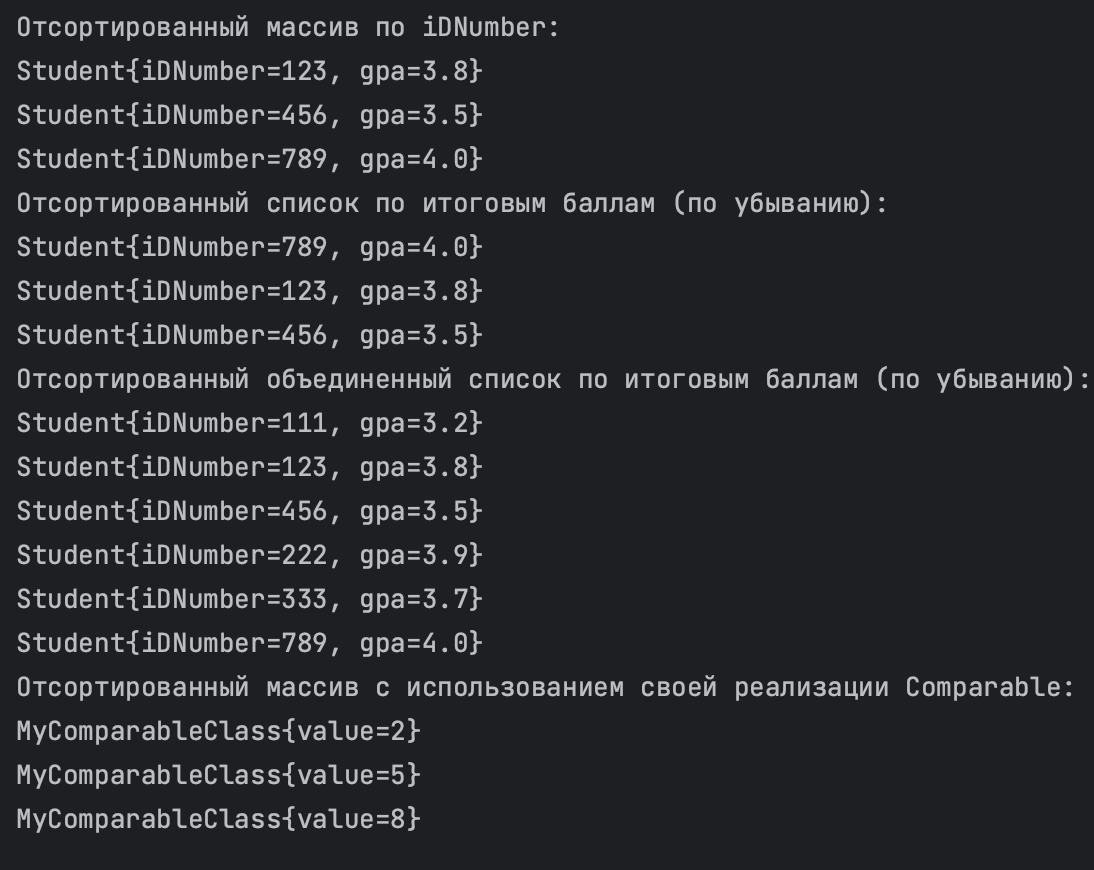
**Практическая работа № 9. Использование полиморфизма при программировании при реализации алгоритмов сортировок и поиска**

**Цель работы**: освоение на практике методов сортировки с использованием приемов программирования на объектно-ориентированном языке Java.

Код:

|  |
| --- |
| import java.util.Arrays; import java.util.Comparator;  // Класс Student class Student implements Comparable<Student> {  private int iDNumber;  private double gpa;   public Student(int iDNumber, double gpa) {  this.iDNumber = iDNumber;  this.gpa = gpa;  }   public int getIDNumber() {  return iDNumber;  }   public double getGPA() {  return gpa;  }   @Override  public String toString() {  return "Student{" +  "iDNumber=" + iDNumber +  ", gpa=" + gpa +  '}';  }   @Override  public int compareTo(Student other) {  return Integer.*compare*(this.iDNumber, other.iDNumber);  } }  // Класс SortingStudentsByGPA class SortingStudentsByGPA implements Comparator<Student> {  @Override  public int compare(Student student1, Student student2) {  if (student1.getGPA() > student2.getGPA()) {  return -1;  } else if (student1.getGPA() < student2.getGPA()) {  return 1;  } else {  return 0;  }  } }  // Программа public class Main {  public static void main(String[] args) {  // Задание 1: Создание массива класса Student и сортировка по iDNumber с использованием сортировки вставками  Student[] students = {  new Student(123, 3.8),  new Student(456, 3.5),  new Student(789, 4.0)  };   for (int i = 1; i < students.length; i++) {  Student key = students[i];  int j = i - 1;  while (j >= 0 && students[j].compareTo(key) > 0) {  students[j + 1] = students[j];  j--;  }  students[j + 1] = key;  }   System.*out*.println("Отсортированный массив по iDNumber:");  for (Student student : students) {  System.*out*.println(student);  }   // Задание 2: Создание списка студентов и сортировка по итоговым баллам в порядке убывания с использованием алгоритма быстрой сортировки  Student[] studentsList = {  new Student(123, 3.8),  new Student(456, 3.5),  new Student(789, 4.0)  };   Arrays.*sort*(studentsList, new SortingStudentsByGPA());   System.*out*.println("Отсортированный список по итоговым баллам (по убыванию):");  for (Student student : studentsList) {  System.*out*.println(student);  }   // Задание 3: Объединение двух списков данных о студентах в один отсортированный список с использованием алгоритма сортировки слиянием  Student[] studentsList1 = {  new Student(123, 3.8),  new Student(456, 3.5),  new Student(789, 4.0)  };  Student[] studentsList2 = {  new Student(111, 3.2),  new Student(222, 3.9),  new Student(333, 3.7)  };   Student[] mergedList = *mergeSort*(studentsList1, studentsList2, new SortingStudentsByGPA());   System.*out*.println("Отсортированный объединенный список по итоговым баллам (по убыванию):");  for (Student student : mergedList) {  System.*out*.println(student);  }   // Задание 4: Реализация своей собственной версии интерфейса Comparable  MyComparableClass[] comparableArray = {  new MyComparableClass(5),  new MyComparableClass(2),  new MyComparableClass(8)  };   Arrays.*sort*(comparableArray);   System.*out*.println("Отсортированный массив с использованием своей реализации Comparable:");  for (MyComparableClass item : comparableArray) {  System.*out*.println(item);  }  }   // Метод для слияния и сортировки двух списков  private static Student[] mergeSort(Student[] list1, Student[] list2, Comparator<Student> comparator) {  int size1 = list1.length;  int size2 = list2.length;  int mergedSize = size1 + size2;  Student[] mergedList = new Student[mergedSize];   int i = 0, j = 0, k = 0;  while (i < size1 && j < size2) {  if (comparator.compare(list1[i], list2[j]) > 0) {  mergedList[k++] = list1[i++];  } else {  mergedList[k++] = list2[j++];  }  }   while (i < size1) {  mergedList[k++] = list1[i++];  }   while (j < size2) {  mergedList[k++] = list2[j++];  }   return mergedList;  } }  // Класс MyComparableClass class MyComparableClass implements Comparable<MyComparableClass> {  private int value;   public MyComparableClass(int value) {  this.value = value;  }   public int getValue() {  return value;  }   @Override  public int compareTo(MyComparableClass other) {  return Integer.*compare*(this.value, other.getValue());  }   @Override  public String toString() {  return "MyComparableClass{" +  "value=" + value +  '}';  } } |

Результат:



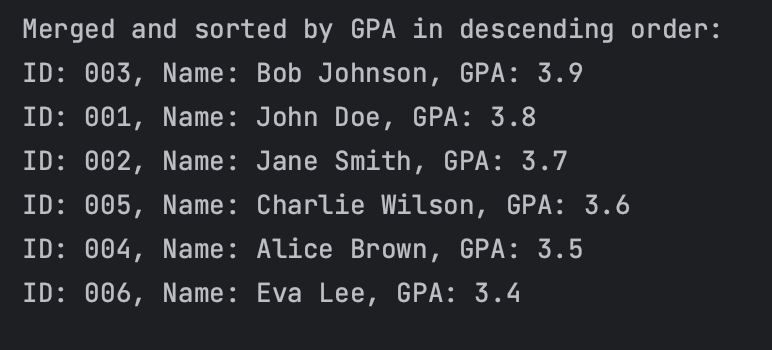
**Практическая работа № 10. Стандартные интерфейсы Джава. Интерфейс Comparator**

**Цель:** цель данной практической работы - закрепить знания в области использования стандартных интерфейсов языка Джава, научиться применять интерфейсы для разработки практических программ на Джаве

Код:

|  |
| --- |
| import java.util.ArrayList; import java.util.Comparator;  public class MergeAndSortStudents {  public static void main(String[] args) {  ArrayList<Student> studentList1 = new ArrayList<>();  studentList1.add(new Student("001", "John", "Doe", 3.8));  studentList1.add(new Student("002", "Jane", "Smith", 3.7));  studentList1.add(new Student("003", "Bob", "Johnson", 3.9));   ArrayList<Student> studentList2 = new ArrayList<>();  studentList2.add(new Student("004", "Alice", "Brown", 3.5));  studentList2.add(new Student("005", "Charlie", "Wilson", 3.6));  studentList2.add(new Student("006", "Eva", "Lee", 3.4));   SortingStudents2ByGPA sorter = new SortingStudents2ByGPA();   // Объединяем два списка в один  ArrayList<Student> mergedStudentList = new ArrayList<>();  mergedStudentList.addAll(studentList1);  mergedStudentList.addAll(studentList2);   // Сортируем объединенный список по GPA  sorter.setArray(mergedStudentList);  sorter.quicksort(Comparator.comparing(Student::getGPA).reversed());   System.out.println("Merged and sorted by GPA in descending order:");  sorter.outArray();  } }  import java.util.ArrayList; import java.util.Collections; import java.util.Comparator;  public class SortingStudents2ByGPA {  private ArrayList<Student> studentList = new ArrayList<>();   public void setArray(ArrayList<Student> students) {  studentList.addAll(students);  }   public void quicksort(Comparator<Student> comparator) {  Collections.*sort*(studentList, comparator);  }   public void mergeSort(Comparator<Student> comparator) {  Collections.*sort*(studentList, comparator);  }   public void outArray() {  for (Student student : studentList) {  System.*out*.println("ID: " + student.getIDNumber() +  ", Name: " + student.getFirstName() + " " + student.getLastName() +  ", GPA: " + student.getGPA());  }  }   public static void main(String[] args) {  SortingStudents2ByGPA sorter = new SortingStudents2ByGPA();   ArrayList<Student> students = new ArrayList<>();  students.add(new Student("001", "John", "Doe", 3.8));  students.add(new Student("002", "Jane", "Smith", 3.7));  students.add(new Student("003", "Bob", "Johnson", 3.9));   sorter.setArray(students);   System.*out*.println("Unsorted:");  sorter.outArray();   // Sorting by GPA in descending order  sorter.quicksort(Comparator.*comparing*(Student::getGPA).reversed());   System.*out*.println("Sorted by GPA in descending order:");  sorter.outArray();  } }  public class Student {  private String iDNumber;  private String firstName;  private String lastName;  private double gpa;   public Student(String iDNumber, String firstName, String lastName, double gpa) {  this.iDNumber = iDNumber;  this.firstName = firstName;  this.lastName = lastName;  this.gpa = gpa;  }   public String getIDNumber() {  return iDNumber;  }   public String getFirstName() {  return firstName;  }   public String getLastName() {  return lastName;  }   public double getGPA() {  return gpa;  } } |

Результат:



**Практическая работа № 11. Работа с датой и временем**

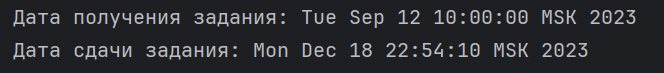
**Цель** данной практической работы – научиться работать с датами и временем, применять методы класса Date и Calenadar, других классов для обработки строк.

Код:

Задание 1:

|  |
| --- |
| import java.util.Date;  public class Task1 {  public static void main(String[] args) {  String developerName = "Ваша фамилия";  Date assignmentDate = new Date(2023-1900, 9-1, 12, 10, 0, 0);  Date submissionDate = new Date();   System.*out*.println("Фамилия разработчика: " + developerName);  System.*out*.println("Дата получения задания: " + assignmentDate);  System.*out*.println("Дата сдачи задания: " + submissionDate);  } } |

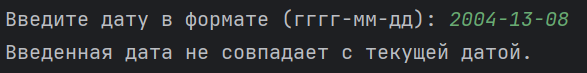
Результат:



Задание 2:

|  |
| --- |
| import java.text.SimpleDateFormat; import java.util.Date; import java.util.Scanner;  public class Task2 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);  System.*out*.print("Введите дату в формате (гггг-мм-дд): ");  String userInput = scanner.nextLine();  scanner.close();   SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");  String currentDate = dateFormat.format(new Date());   if (userInput.equals(currentDate)) {  System.*out*.println("Введенная дата совпадает с текущей датой.");  } else {  System.*out*.println("Введенная дата не совпадает с текущей датой.");  }  } } |

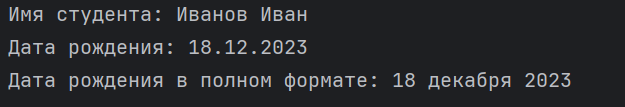
Результат:



Задание 3:

|  |  |
| --- | --- |
| import java.text.SimpleDateFormat; import java.util.Date;  public class Task3 {  private String name;  private Date birthDate;   public Task3(String name, Date birthDate) {  this.name = name;  this.birthDate = birthDate;  }   public String getFormattedBirthDate(String format) {  SimpleDateFormat dateFormat = new SimpleDateFormat(format);  return dateFormat.format(birthDate);  }   @Override  public String toString() {  return "Имя студента: " + name + "\nДата рождения: " + getFormattedBirthDate("dd.MM.yyyy");  }   public static void main(String[] args) {  Date birthDate = new Date(); // Замените на дату рождения  Task3 student = new Task3("Иванов Иван", birthDate);   System.*out*.println(student);  System.*out*.println("Дата рождения в полном формате: " + student.getFormattedBirthDate("dd MMMM yyyy"));  } } |  |

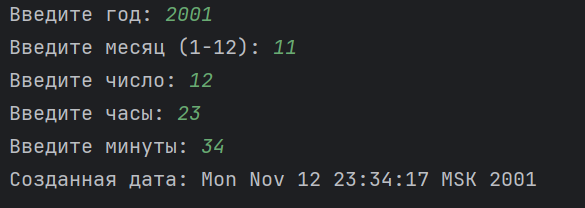
Результат:



Задание 4:

|  |  |
| --- | --- |
| import java.util.Calendar; import java.util.Date; import java.util.Scanner;  public class Task4 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);  System.*out*.print("Введите год: ");  int year = scanner.nextInt();  System.*out*.print("Введите месяц (1-12): ");  int month = scanner.nextInt();  System.*out*.print("Введите число: ");  int day = scanner.nextInt();  System.*out*.print("Введите часы: ");  int hour = scanner.nextInt();  System.*out*.print("Введите минуты: ");  int minute = scanner.nextInt();  scanner.close();   Calendar calendar = Calendar.*getInstance*();  calendar.set(year, month - 1, day, hour, minute);  Date date = calendar.getTime();   System.*out*.println("Созданная дата: " + date);  } } |  |

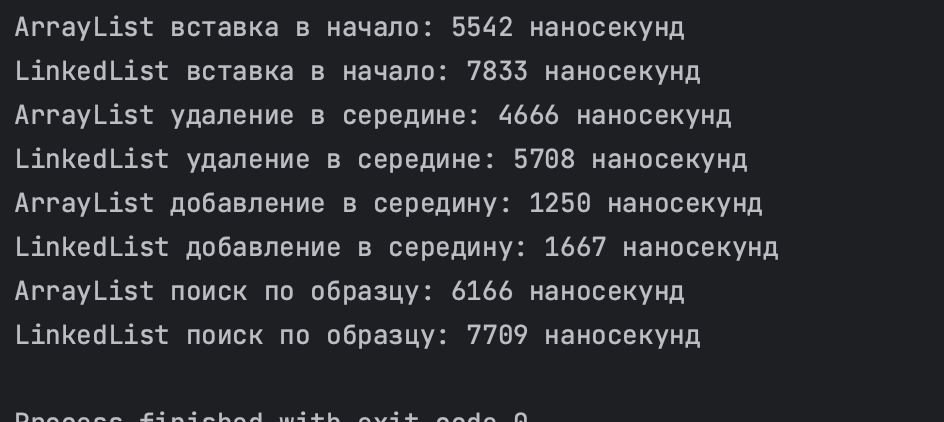
Результат:



Задание 5:

|  |  |
| --- | --- |
| import java.util.ArrayList;  import java.util.LinkedList;  import java.util.Random;  public class Pr5 {  public static void main(String[] args) {  int n = 100;  Random random = new Random();  ArrayList<Integer> arrayList = new ArrayList<>();  LinkedList<Integer> linkedList = new LinkedList<>();  // Заполнение коллекций случайными числами  for (int i = 0; i < n; i++) {  arrayList.add(i);  linkedList.add(i);  }  // Замер времени для операции вставки в начало  long startTime = System.nanoTime();  arrayList.add(0, 999);  long endTime = System.nanoTime();  System.out.println("ArrayList вставка в начало: " + (endTime - startTime) + " наносекунд");  startTime = System.nanoTime();  linkedList.add(0, 999);  endTime = System.nanoTime();  System.out.println("LinkedList вставка в начало: " + (endTime - startTime) + " наносекунд");  // Замер времени для операции удаления в середине  int index = n / 2;  startTime = System.nanoTime();  arrayList.remove(index);  endTime = System.nanoTime();  System.out.println("ArrayList удаление в середине: " + (endTime - startTime) + " наносекунд");  startTime = System.nanoTime();  linkedList.remove(index);  endTime = System.nanoTime();  System.out.println("LinkedList удаление в середине: " + (endTime - startTime) + " наносекунд");  // Замер времени для операции добавления в середину  index = n / 2;  int element = random.nextInt(1000);  startTime = System.nanoTime();  arrayList.add(index, element);  endTime = System.nanoTime();  System.out.println("ArrayList добавление в середину: " + (endTime - startTime) + " наносекунд");  startTime = System.nanoTime();  linkedList.add(index, element);  endTime = System.nanoTime();  System.out.println("LinkedList добавление в середину: " + (endTime - startTime) + " наносекунд");  // Замер времени для операции поиска элемента  int searchElement = arrayList.get(random.nextInt(n)); // Случайный элемент для поиска  startTime = System.nanoTime();  arrayList.contains(searchElement);  endTime = System.nanoTime();  System.out.println("ArrayList поиск по образцу: " + (endTime - startTime) + " наносекунд");  startTime = System.nanoTime();  linkedList.contains(searchElement);  endTime = System.nanoTime();  System.out.println("LinkedList поиск по образцу: " + (endTime - startTime) + " наносекунд");  }  } |  |

Результат:



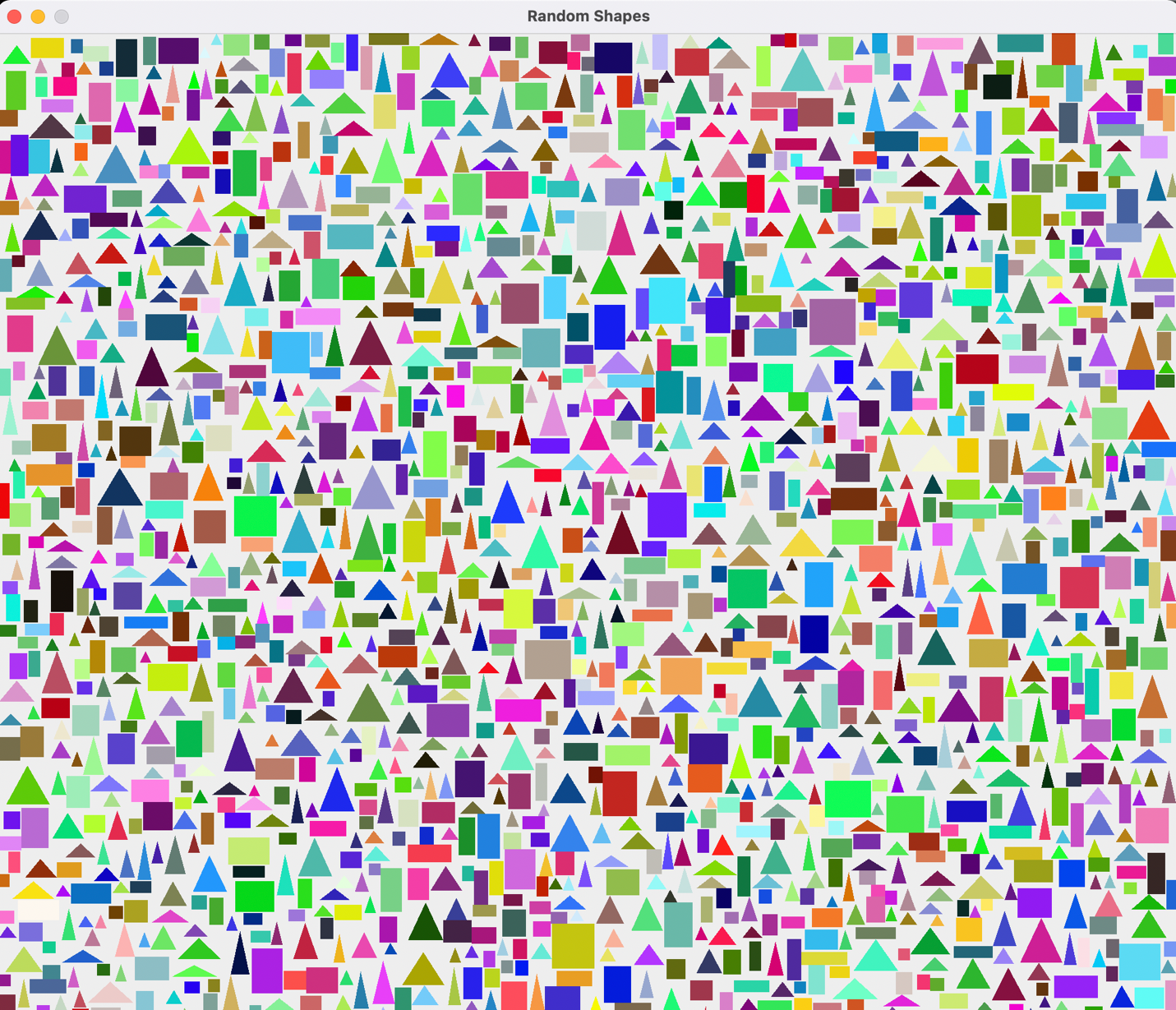
**Практическая работа № 12. Создание программ с графическим интерфейсом пользователя на языке Джава. Компоновка объектов с помощью Layout менеджеров**

**Цель**: цель данной практической работы - научиться создавать графический интерфейс пользователя, освоить на практике работу с различными объектами для создания GUI, менеджерами размещения компонентов.

Код:

|  |  |
| --- | --- |
| import javax.swing.\*;  import java.awt.\*;  import java.util.Random;  // Абстрактный класс для представления базовой геометрической фигуры  abstract class Shape {  protected Color color;  protected int x, y;  protected int width, height;  // Конструктор для инициализации цвета, координат X и Y, а также размеров ширины и высоты фигуры  public Shape(Color color, int x, int y, int width, int height) {  this.color = color;  this.x = x;  this.y = y;  this.width = width;  this.height = height;  }  // Абстрактный метод для отрисовки фигуры  public abstract void draw(Graphics g);  // Абстрактный метод для проверки пересечения с другой фигурой  public abstract boolean intersects(Shape other);  }  // Класс, представляющий квадрат, наследуется от Shape  class Square extends Shape {  // Конструктор для квадрата  public Square(Color color, int x, int y, int width, int height) {  super(color, x, y, width, height);  }  // Метод для отрисовки квадрата  @Override  public void draw(Graphics g) {  g.setColor(color);  g.fillRect(x, y, width, height);  }  // Метод для проверки пересечения с другой фигурой (квадратом или треугольником)  @Override  public boolean intersects(Shape other) {  return x < other.x + other.width && x + width > other.x &&  y < other.y + other.height && y + height > other.y;  }  }  // Класс, представляющий треугольник, наследуется от Shape  class Triangle extends Shape {  // Конструктор для треугольника  public Triangle(Color color, int x, int y, int width, int height) {  super(color, x, y, width, height);  }  // Метод для отрисовки треугольника  @Override  public void draw(Graphics g) {  g.setColor(color);  int[] xPoints = {x, x + width / 2, x + width};  int[] yPoints = {y + height, y, y + height};  g.fillPolygon(xPoints, yPoints, 3);  }  // Метод для проверки пересечения с другой фигурой (квадратом или треугольником)  @Override  public boolean intersects(Shape other) {  return x < other.x + other.width && x + width > other.x &&  y < other.y + other.height && y + height > other.y;  }  }  // Основной класс программы  public class Main {  public static void main(String[] args) {  // Создаем окно с заголовком "Random Shapes"  JFrame frame = new JFrame("Random Shapes");  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.setSize(1000, 1000); // Устанавливаем размер окна  frame.setResizable(false);  frame.add(new DrawingPanel()); // Добавляем панель для рисования на окно  frame.setVisible(true); // Отображаем окно  }  // Вложенный класс для панели, на которой будут рисоваться фигуры  static class DrawingPanel extends JPanel {  private Random rand = new Random();  private boolean[][] occupied;  private int panelWidth = 1000; // Ширина панели  private int panelHeight = 1000; // Высота панели  // Конструктор панели  public DrawingPanel() {  occupied = new boolean[panelWidth][panelHeight]; // Инициализируем массив для отслеживания занятых областей  }  // Метод для отрисовки фигур на панели  @Override  protected void paintComponent(Graphics g) {  super.paintComponent(g);  // Создаем случайные фигуры и рисуем их на панели  for (int i = 0; i < 2000; i++) {  Color color = new Color(rand.nextInt(256), rand.nextInt(256), rand.nextInt(256)); // Генерируем случайный цвет  int x, y, width, height;  boolean intersects;  do {  width = rand.nextInt(30) + 10; // Генерируем случайную ширину (минимальная ширина 10)  height = rand.nextInt(30) + 10; // Генерируем случайную высоту (минимальная высота 10)  x = generateRandomX(width); // Генерируем случайную X-координату в пределах панели  y = generateRandomY(height); // Генерируем случайную Y-координату в пределах панели  intersects = intersectsExistingShapes(x, y, width, height); // Проверяем, пересекается ли новая фигура с существующими  } while (intersects);  Shape shape;  if (rand.nextBoolean()) {  shape = new Square(color, x, y, width, height); // Создаем случайный квадрат  } else {  shape = new Triangle(color, x, y, width, height); // Создаем случайный треугольник  }  shape.draw(g); // Рисуем фигуру на панели  }  }  // Метод для генерации случайной X-координаты в пределах панели  private int generateRandomX(int width) {  return rand.nextInt(panelWidth - width);  }  // Метод для генерации случайной Y-координаты в пределах панели  private int generateRandomY(int height) {  return rand.nextInt(panelHeight - height);  }  // Метод для проверки пересечения новой фигуры с уже существующими  private boolean intersectsExistingShapes(int x, int y, int width, int height) {  for (int i = x; i < x + width; i++) {  for (int j = y; j < y + height; j++) {  if (occupied[i][j]) {  return true;  }  }  }  for (int i = x; i < x + width; i++) {  for (int j = y; j < y + height; j++) {  occupied[i][j] = true;  }  }  return false;  }  }  } |  |

Результат:



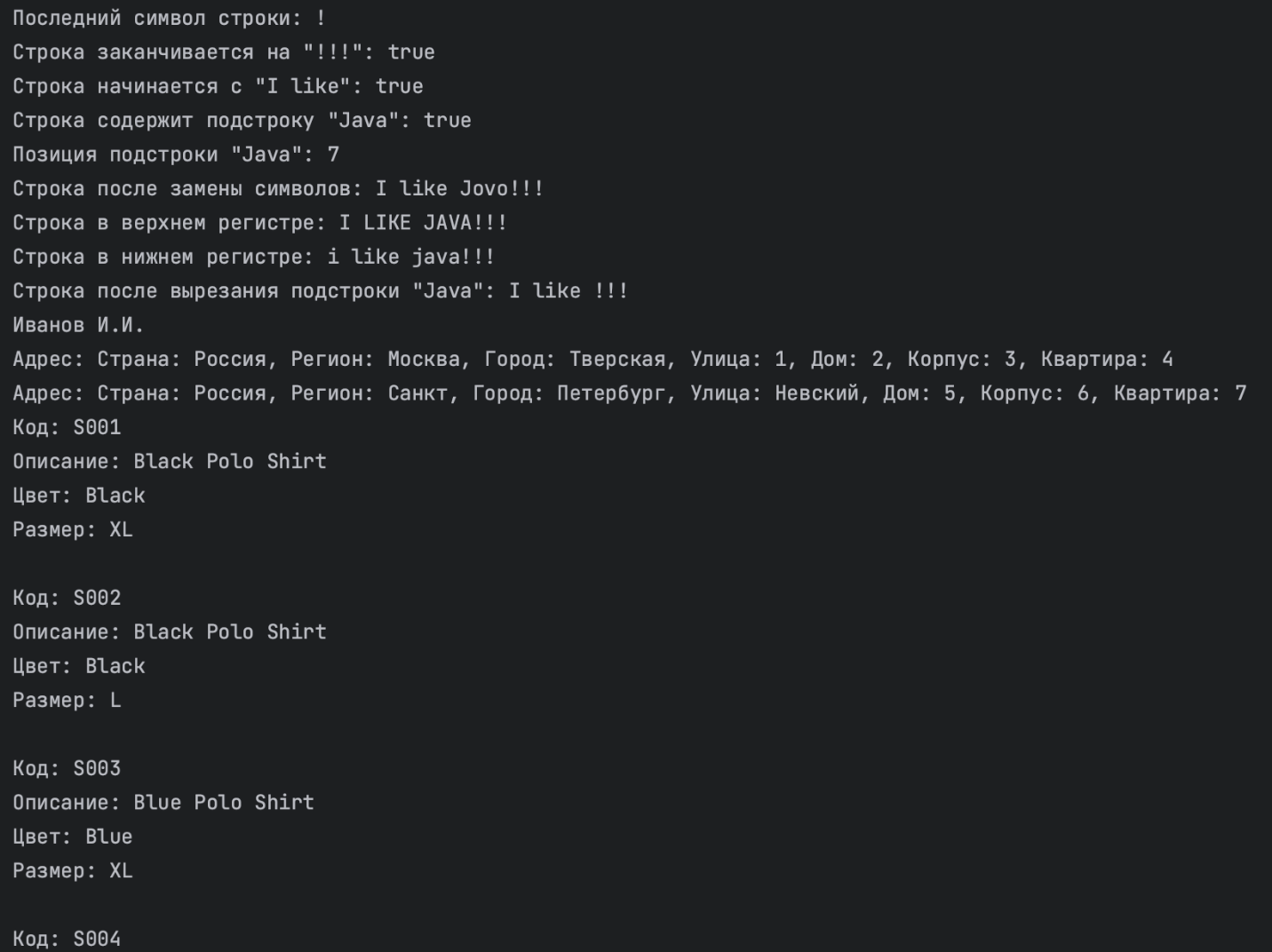
**Практическая работа № 13. Обработка строк в Java**

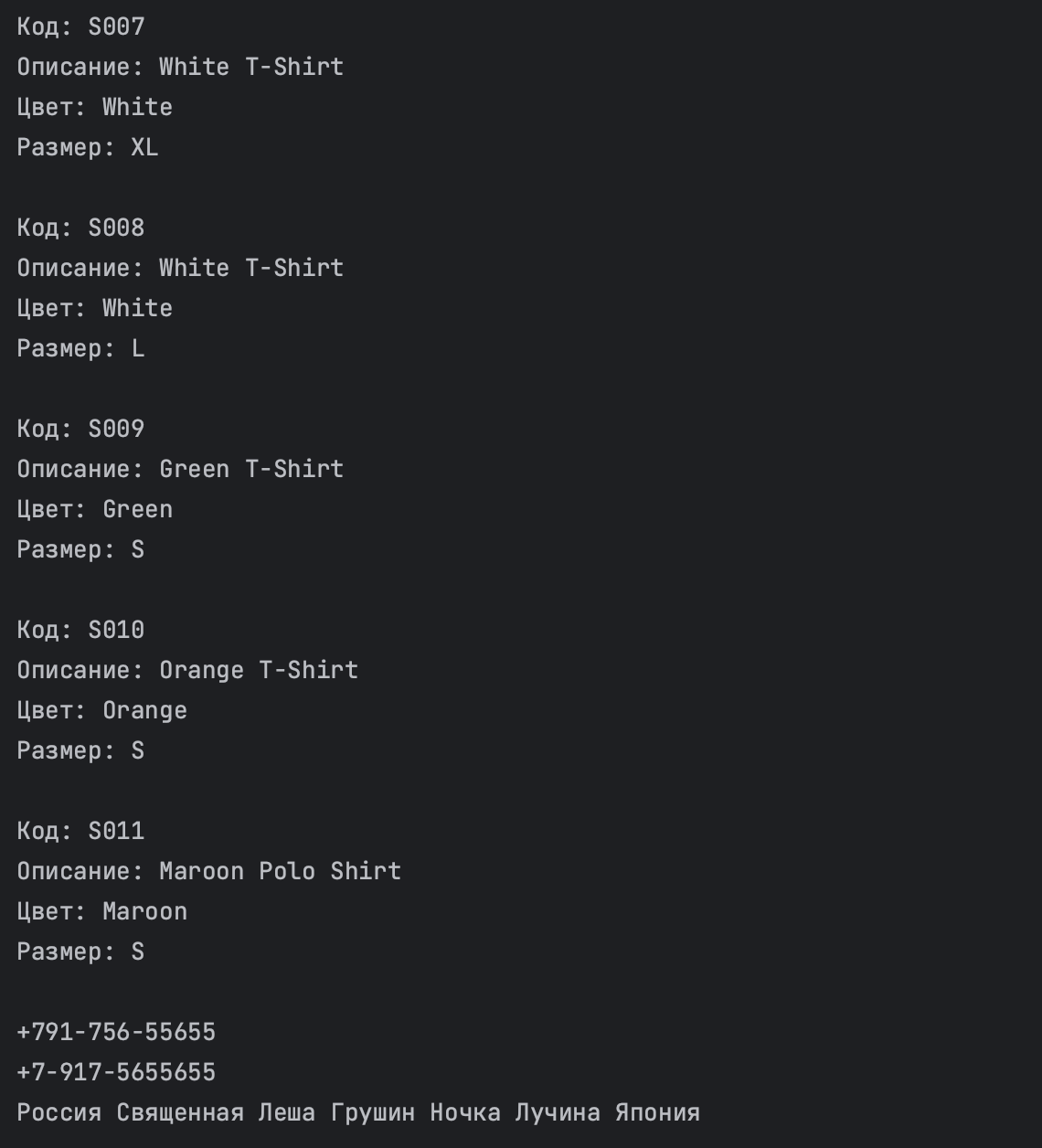
**Цель**: закрепить знания в области обработки строк, научиться применять методы класса String и других классов для обработки строк.

Код:

|  |  |
| --- | --- |
| import java.util.StringTokenizer;  import java.io.BufferedReader;  import java.io.FileReader;  import java.io.IOException;  class Main {  public static void main(String[] args) {  // Задание 1  String str = "I like Java!!!";  printLastCharacter(str);  checkEndsWith(str, "!!!");  checkStartsWith(str, "I like");  checkContains(str, "Java");  findSubstringPosition(str, "Java");  replaceCharacters(str, 'a', 'o');  convertToUpperCase(str);  convertToLowerCase(str);  cutSubstring(str, "Java");  // Задание 2  String firstName = "Иван";  String lastName = "Иванов";  String middleName = "Иванович";  System.out.println(getFullName(firstName, lastName, middleName));  // Задание 3  String addressString = "Россия, Москва, Тверская, 1, 2, 3, 4";  Address address = parseAddress(addressString, ",");  System.out.println(address);  addressString = "Россия; Санкт-Петербург; Невский; 5; 6; 7; 8";  address = parseAddress(addressString, "[,.;-]");  System.out.println(address);  // Задание 4  String[] shirtStrings = {  "S001,Black Polo Shirt,Black,XL",  "S002,Black Polo Shirt,Black,L",  "S003,Blue Polo Shirt,Blue,XL",  "S004,Blue Polo Shirt,Blue,M",  "S005,Tan Polo Shirt,Tan,XL",  "S006,Black T-Shirt,Black,XL",  "S007,White T-Shirt,White,XL",  "S008,White T-Shirt,White,L",  "S009,Green T-Shirt,Green,S",  "S010,Orange T-Shirt,Orange,S",  "S011,Maroon Polo Shirt,Maroon,S"  };  Shirt[] shirts = convertToShirts(shirtStrings);  printShirts(shirts);  // Задание 5  String phoneNumber1 = "+79175655655";  String phoneNumber2 = "89175655655";  String formattedPhoneNumber1 = formatPhoneNumber(phoneNumber1);  String formattedPhoneNumber2 = formatPhoneNumber(phoneNumber2);  System.out.println(formattedPhoneNumber1);  System.out.println(formattedPhoneNumber2);  // Задание 6  String fileName = "/Users/vsevolodgav/IdeaProjects/practice13/src/words.txt";  String result = getLine(fileName);  System.out.println(result);  }  // Задание 1  public static void printLastCharacter(String str) {  char lastChar = str.charAt(str.length() - 1);  System.out.println("Последний символ строки: " + lastChar);  }  public static void checkEndsWith(String str, String suffix) {  boolean endsWithSuffix = str.endsWith(suffix);  System.out.println("Строка заканчивается на \"" + suffix + "\": " + endsWithSuffix);  }  public static void checkStartsWith(String str, String prefix) {  boolean startsWithPrefix = str.startsWith(prefix);  System.out.println("Строка начинается с \"" + prefix + "\": " + startsWithPrefix);  }  public static void checkContains(String str, String substring) {  boolean containsSubstring = str.contains(substring);  System.out.println("Строка содержит подстроку \"" + substring + "\": " + containsSubstring);  }  public static void findSubstringPosition(String str, String substring) {  int position = str.indexOf(substring);  System.out.println("Позиция подстроки \"" + substring + "\": " + position);  }  public static void replaceCharacters(String str, char oldChar, char newChar) {  String replacedStr = str.replace(oldChar, newChar);  System.out.println("Строка после замены символов: " + replacedStr);  }  public static void convertToUpperCase(String str) {  String upperCaseStr = str.toUpperCase();  System.out.println("Строка в верхнем регистре: " + upperCaseStr);  }  public static void convertToLowerCase(String str) {  String lowerCaseStr = str.toLowerCase();  System.out.println("Строка в нижнем регистре: " + lowerCaseStr);  }  public static void cutSubstring  (String str, String substring) {  String cutStr = str.replace(substring, "");  System.out.println("Строка после вырезания подстроки \"" + substring + "\": " + cutStr);  }  // Задание 2  public static String getFullName(String firstName, String lastName, String middleName) {  StringBuilder fullName = new StringBuilder();  if (lastName != null) {  fullName.append(lastName);  if (firstName != null) {  fullName.append(" ").append(firstName.charAt(0)).append(".");  if (middleName != null) {  fullName.append(middleName.charAt(0)).append(".");  }  }  }  return fullName.toString();  }  // Задание 3  public static Address parseAddress(String addressString, String delimiter) {  StringTokenizer tokenizer = new StringTokenizer(addressString, delimiter);  String country = tokenizer.nextToken().trim();  String region = tokenizer.nextToken().trim();  String city = tokenizer.nextToken().trim();  String street = tokenizer.nextToken().trim();  String house = tokenizer.nextToken().trim();  String building = tokenizer.nextToken().trim();  String apartment = tokenizer.nextToken().trim();  return new Address(country, region, city, street, house, building, apartment);  }  // Задание 4  public static Shirt[] convertToShirts(String[] shirtStrings) {  Shirt[] shirts = new Shirt[shirtStrings.length];  for (int i = 0; i < shirtStrings.length; i++) {  String[] shirtData = shirtStrings[i].split(",");  String code = shirtData[0];  String description = shirtData[1];  String color = shirtData[2];  String size = shirtData[3];  shirts[i] = new Shirt(code, description, color, size);  }  return shirts;  }  public static void printShirts(Shirt[] shirts) {  for (Shirt shirt : shirts) {  System.out.println(shirt);  }  }  // Задание 5  public static String formatPhoneNumber(String phoneNumber) {  if (phoneNumber.startsWith("+")) {  String countryCode = phoneNumber.substring(1, 4);  String number = phoneNumber.substring(4);  return "+" + countryCode + "-" + number.substring(0, 3) + "-" + number.substring(3);  } else if (phoneNumber.startsWith("8")) {  String number = phoneNumber.substring(1);  return "+7-" + number.substring(0, 3) + "-" + number.substring(3);  } else {  return phoneNumber;  }  }  // Задание 6  public static String getLine(String fileName) {  StringBuilder result = new StringBuilder();  try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {  String line;  while ((line = reader.readLine()) != null) {  result.append(line).append(" ");  }  } catch (IOException e) {  e.printStackTrace();  }  return result.toString().trim();  }  }  class Address {  private String country;  private String region;  private String city;  private String street;  private String house;  private String building;  private String apartment;  public Address(String country, String region, String city, String street, String house, String building, String apartment) {  this.country = country;  this.region = region;  this.city = city;  this.street = street;  this.house = house;  this.building = building;  this.apartment = apartment;  }  @Override  public String toString() {  return "Адрес: " +  "Страна: " + country +  ", Регион: " + region +  ", Город: " + city +  ", Улица: " + street +  ", Дом: " + house +  ", Корпус: " + building +  ", Квартира: " + apartment;  }  }  class Shirt {  private String code;  private String description;  private String color;  private String size;  public Shirt(String code, String description, String color, String size) {  this.code = code;  this.description = description;  this.color = color;  this.size = size;  }  @Override  public String toString() {  return "Код: " + code +  "\nОписание: " + description +  "\nЦвет: " + color +  "\nРазмер: " + size +  "\n";  }  } |  |

Результат:





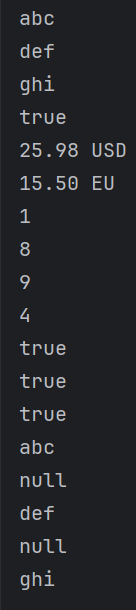
**Практическая работа № 14. Использование регулярных выражений в Джава приложениях**

**Цель**: практической работы – понять особенности использования регулярных выражений в Java, научиться работать с строками и применять регулярные выражения для обработки строк в программах.

Код:

|  |
| --- |
| import java.util.regex.Matcher; import java.util.regex.Pattern; import java.util.HashMap;  public class StringManipulation {   // Задание 1: Манипуляции со строками через регулярные выражения  public void manipulateString() {  String text = "abc123def456ghi";  String[] elements = text.split("\\d+"); // Разбиение по числам   for (String element : elements) {  System.*out*.println(element);  }  }   // Задание 2: Регулярное выражение для проверки строки "abcdefghijklmnopqrstuv18340"  public void checkStringPattern() {  String regex = "^[a-z0-9]{27}$";  String testString = "abcdefghijklmnopqrstuv18340";  boolean matches = testString.matches(regex);  System.*out*.println(matches);  }   // Задание 3: Извлечение цен в разных валютах из текста  public void extractPrices() {  String text = "Prices: 25.98 USD, 30 RUB, 15.50 EU";  String regex = "\\d+\\.\\d+\\s(USD|RUB|EU)";  Pattern pattern = Pattern.*compile*(regex);  Matcher matcher = pattern.matcher(text);   while (matcher.find()) {  System.*out*.println(matcher.group());  }  }   // Задание 4: Поиск цифр без знака "+" в тексте  public void findDigitsWithoutPlus() {  String text = "(1 + 8) – 9 / 4";  String regex = "\\b\\d+(?!\\+)";  Pattern pattern = Pattern.*compile*(regex);  Matcher matcher = pattern.matcher(text);   while (matcher.find()) {  System.*out*.println(matcher.group());  }  }   // Задание 5: Регулярное выражение для даты в формате dd/mm/yyyy  public void checkDateFormat() {  String regex = "^(0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/((19|20)\\d\\d)$";  String testString = "29/02/2000";  boolean matches = testString.matches(regex);  System.*out*.println(matches);  }   // Задание 6: Проверка строки на соответствие RFC 2822 (email адрес)  public void checkRFC2822Email() {  String regex = "^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$";  String testString = "user@example.com";  boolean matches = testString.matches(regex);  System.*out*.println(matches);  }   // Задание 7: Проверка надежности пароля  public void checkPasswordStrength() {  String password = "F032\_Password";  String regex = "^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\\d)[a-zA-Z\\d\_]{8,}$";  boolean matches = password.matches(regex);  System.*out*.println(matches);  }   // Задание 8: Метод filter для убирания лишних элементов из массива  interface Filter {  boolean apply(Object o);  }   public void filterArray(Object[] arr, Filter filter) {  for (int i = 0; i < arr.length; i++) {  if (!filter.apply(arr[i])) {  arr[i] = null;  }  }  }    public static void main(String[] args) {  StringManipulation stringManipulation = new StringManipulation();   // Вызов методов для решения задач  stringManipulation.manipulateString();  stringManipulation.checkStringPattern();  stringManipulation.extractPrices();  stringManipulation.findDigitsWithoutPlus();  stringManipulation.checkDateFormat();  stringManipulation.checkRFC2822Email();  stringManipulation.checkPasswordStrength();   // Пример использования метода filter  Object[] testArray = {"abc", 123, "def", 456, "ghi"};  stringManipulation.filterArray(testArray, o -> o instanceof String);   // Вывод отфильтрованного массива  for (Object element : testArray) {  System.*out*.println(element);  }  } } |

Результат:



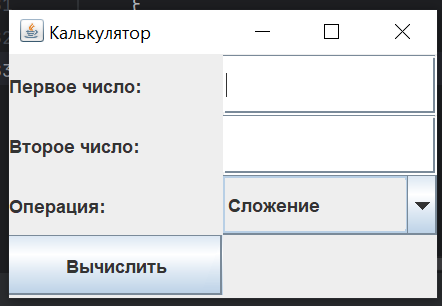
**Практическая работа № 15. Вложенные и внутренние классы. Обработка событий в Джава программах с графическим интерфейсом пользователя**

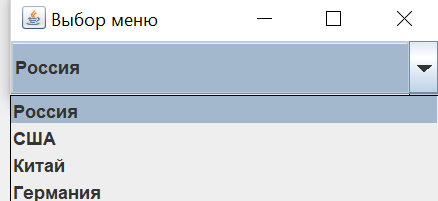
**Цель** данной практической работы - изучить использование анонимных и внутренних классов, научиться разрабатывать интерактивные программы на языке Джава с использованием графического интерфейса пользователя.

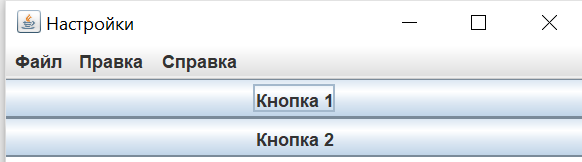
Код:

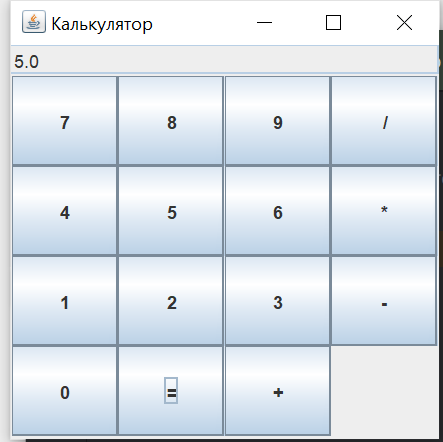
|  |
| --- |
| import javax.swing.\*; import java.awt.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener;  public class Main extends JFrame implements ActionListener {  private JTextField firstNumberField;  private JTextField secondNumberField;  private JComboBox<String> operationComboBox;  private JLabel resultLabel;   public Main() {  setTitle("Калькулятор");  setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  setSize(300, 200);  setLocationRelativeTo(null);   JPanel panel = new JPanel();  panel.setLayout(new GridLayout(4, 2));   JLabel firstNumberLabel = new JLabel("Первое число:");  firstNumberField = new JTextField();  JLabel secondNumberLabel = new JLabel("Второе число:");  secondNumberField = new JTextField();  JLabel operationLabel = new JLabel("Операция:");  String[] operations = {"Сложение", "Вычитание", "Умножение", "Деление"};  operationComboBox = new JComboBox<>(operations);  JButton calculateButton = new JButton("Вычислить");  calculateButton.addActionListener(this);  resultLabel = new JLabel();   panel.add(firstNumberLabel);  panel.add(firstNumberField);  panel.add(secondNumberLabel);  panel.add(secondNumberField);  panel.add(operationLabel);  panel.add(operationComboBox);  panel.add(calculateButton);  panel.add(resultLabel);   add(panel);  setVisible(true);  }   public static void main(String[] args) {  new Main();  }   @Override  public void actionPerformed(ActionEvent e) {  try {  double firstNumber = Double.*parseDouble*(firstNumberField.getText());  double secondNumber = Double.*parseDouble*(secondNumberField.getText());  String operation = (String) operationComboBox.getSelectedItem();  double result = 0;   switch (operation) {  case "Сложение":  result = firstNumber + secondNumber;  break;  case "Вычитание":  result = firstNumber - secondNumber;  break;  case "Умножение":  result = firstNumber \* secondNumber;  break;  case "Деление":  if (secondNumber == 0) {  throw new ArithmeticException("Деление на ноль недопустимо");  }  result = firstNumber / secondNumber;  break;  }   resultLabel.setText("Результат: " + result);  } catch (NumberFormatException ex) {  resultLabel.setText("Ошибка: некорректный формат числа");  } catch (ArithmeticException ex) {  resultLabel.setText("Ошибка: " + ex.getMessage());  }  } }  import javax.swing.\*; import java.awt.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener;  public class Main {  public static void main(String[] args) {  JFrame frame = new JFrame("Выбор меню");  frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);   String[] countries = {"Россия", "США", "Китай", "Германия"};  JComboBox<String> comboBox = new JComboBox<>(countries);   comboBox.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  String selectedCountry = (String) comboBox.getSelectedItem();  JOptionPane.*showMessageDialog*(frame, "Выбрана страна: " + selectedCountry);  }  });   frame.getContentPane().add(comboBox, BorderLayout.*CENTER*);  frame.setSize(300, 200);  frame.setVisible(true);  } }  import javax.swing.\*; import java.awt.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener;  public class Main {  public static void main(String[] args) {  JFrame frame = new JFrame("Настройки");  frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);   JMenuBar menuBar = new JMenuBar();   JMenu fileMenu = new JMenu("Файл");  JMenuItem saveItem = new JMenuItem("Сохранить");  JMenuItem exitItem = new JMenuItem("Выйти");  fileMenu.add(saveItem);  fileMenu.add(exitItem);   JMenu editMenu = new JMenu("Правка");  JMenuItem copyItem = new JMenuItem("Копировать");  JMenuItem cutItem = new JMenuItem("Вырезать");  JMenuItem pasteItem = new JMenuItem("Вставить");  editMenu.add(copyItem);  editMenu.add(cutItem);  editMenu.add(pasteItem);   JMenu helpMenu = new JMenu("Справка");   menuBar.add(fileMenu);  menuBar.add(editMenu);  menuBar.add(helpMenu);   JPanel panel = new JPanel();  panel.setLayout(new GridLayout(2, 1));   JButton button1 = new JButton("Кнопка 1");  JButton button2 = new JButton("Кнопка 2");   JTextArea textArea = new JTextArea();   panel.add(button1);  panel.add(button2);   frame.setJMenuBar(menuBar);  frame.getContentPane().add(panel, BorderLayout.*NORTH*);  frame.getContentPane().add(textArea, BorderLayout.*CENTER*);   saveItem.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  JOptionPane.*showMessageDialog*(frame, "Выбрано меню 'Сохранить'");  }  });   exitItem.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  System.*exit*(0);  }  });   copyItem.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  JOptionPane.*showMessageDialog*(frame, "Выбрано меню 'Копировать'");  }  });   cutItem.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  JOptionPane.*showMessageDialog*(frame, "Выбрано меню 'Вырезать'");  }  });   pasteItem.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  JOptionPane.*showMessageDialog*(frame, "Выбрано меню 'Вставить'");  }  });   frame.setSize(400, 300);  frame.setVisible(true);  } }  import javax.swing.\*; import java.awt.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener;  public class Main {  private static JTextField *textField*;  private static double *num1* = 0;  private static double *num2* = 0;  private static char *operator* = ' ';   public static void main(String[] args) {  JFrame frame = new JFrame("Калькулятор");  frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);   JPanel panel = new JPanel();  panel.setLayout(new GridLayout(4, 4));   *textField* = new JTextField();  *textField*.setEditable(false);   JButton button0 = new JButton("0");  JButton button1 = new JButton("1");  JButton button2 = new JButton("2");  JButton button3 = new JButton("3");  JButton button4 = new JButton("4");  JButton button5 = new JButton("5");  JButton button6 = new JButton("6");  JButton button7 = new JButton("7");  JButton button8 = new JButton("8");  JButton button9 = new JButton("9");  JButton buttonAdd = new JButton("+");  JButton buttonSubtract = new JButton("-");  JButton buttonMultiply = new JButton("\*");  JButton buttonDivide = new JButton("/");  JButton buttonEquals = new JButton("=");   button0.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *textField*.setText(*textField*.getText() + "0");  }  });   button1.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *textField*.setText(*textField*.getText() + "1");  }  });   button2.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *textField*.setText(*textField*.getText() + "2");  }  });   button3.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *textField*.setText(*textField*.getText() + "3");  }  });   button4.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *textField*.setText(*textField*.getText() + "4");  }  });   button5.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *textField*.setText(*textField*.getText() + "5");  }  });   button6.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *textField*.setText(*textField*.getText() + "6");  }  });   button7.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *textField*.setText(*textField*.getText() + "7");  }  });   button8.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *textField*.setText(*textField*.getText() + "8");  }  });   button9.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *textField*.setText(*textField*.getText() + "9");  }  });   buttonAdd.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *num1* = Double.*parseDouble*(*textField*.getText());  *operator* = '+';  *textField*.setText("");  }  });   buttonSubtract.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *num1* = Double.*parseDouble*(*textField*.getText());  *operator* = '-';  *textField*.setText("");  }  });   buttonMultiply.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *num1* = Double.*parseDouble*(*textField*.getText());  *operator* = '\*';  *textField*.setText("");  }  });   buttonDivide.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *num1* = Double.*parseDouble*(*textField*.getText());  *operator* = '/';  *textField*.setText("");  }  });   buttonEquals.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *num2* = Double.*parseDouble*(*textField*.getText());  double result = 0;   switch (*operator*) {  case '+':  result = *num1* + num2;  break;  case '-':  result = num1 - num2;  break;  case '\*':  result = num1 \* num2;  break;  case '/':  if (num2 != 0) {  result = num1 / num2;  } else {  JOptionPane.showMessageDialog(frame, "Ошибка: деление на ноль");  }  break;  }   textField.setText(String.valueOf(result));  }  });   panel.add(button7);  panel.add(button8);  panel.add(button9);  panel.add(buttonDivide);  panel.add(button4);  panel.add(button5);  panel.add(button6);  panel.add(buttonMultiply);  panel.add(button1);  panel.add(button2);  panel.add(button3);  panel.add(buttonSubtract);  panel.add(button0);  panel.add(buttonEquals);  panel.add(buttonAdd);   frame.getContentPane().add(*textField*, BorderLayout.*NORTH*);  frame.getContentPane().add(panel, BorderLayout.*CENTER*);   frame.setSize(300, 300);  frame.setVisible(true);  } } |

Результат:









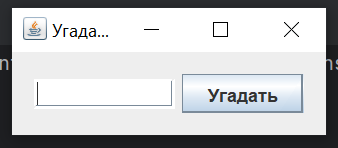
**Практическая работа № 16. Обработка событий мыши и клавиатуры программах на Джава с графическим интерфейсом пользователя**

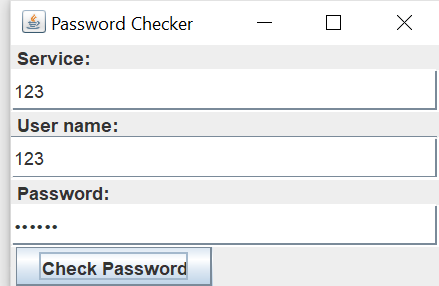
**Цель**: научиться обрабатывать различные события мыши и клавиатуры для разных компонентов

Код:

|  |
| --- |
| import javax.swing.\*; import java.awt.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.awt.event.MouseEvent; import java.awt.event.MouseListener; import java.util.Random;  public class GuessingGame extends JFrame {  private int secretNumber;  private int attempts = 3;   private JTextField textField;  private JLabel infoLabel;   public GuessingGame() {  setTitle("Угадай число");  setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  setLayout(new BorderLayout());   // Генерируем случайное число для угадывания  Random random = new Random();  secretNumber = random.nextInt(21);   JPanel topPanel = new JPanel();  infoLabel = new JLabel("Угадай число от 0 до 20. Попыток осталось: " + attempts);  topPanel.add(infoLabel);  add(topPanel, BorderLayout.*NORTH*);   JPanel centerPanel = new JPanel();  textField = new JTextField(10);  centerPanel.add(textField);   JButton guessButton = new JButton("Угадать");  centerPanel.add(guessButton);  add(centerPanel, BorderLayout.*CENTER*);   guessButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  int guess = Integer.*parseInt*(textField.getText());  if (guess == secretNumber) {  showEndGameMessage("Поздравляем! Вы угадали число " + secretNumber);  } else {  attempts--;  if (attempts > 0) {  String message = "Неверно! Число " + (guess > secretNumber ? "меньше" : "больше") +  " загаданного. Попыток осталось: " + attempts;  infoLabel.setText(message);  } else {  showEndGameMessage("Игра окончена. Загаданное число было: " + secretNumber);  }  }  if (attempts == 0) {  showEndGameMessage("Игра окончена. Загаданное число было: " + secretNumber);  }  textField.setText("");  }  });   // Добавление обработчиков мыши для каждой области окна (для отладки)  centerPanel.addMouseListener(new MouseListener() {  @Override  public void mouseClicked(MouseEvent e) {  }   @Override  public void mousePressed(MouseEvent e) {  }   @Override  public void mouseReleased(MouseEvent e) {  }   @Override  public void mouseEntered(MouseEvent e) {  System.*out*.println("Добро пожаловать в Центральный район");  }   @Override  public void mouseExited(MouseEvent e) {  }  });   // Добавление обработчиков мыши для северной области окна  JPanel northPanel = new JPanel();  northPanel.addMouseListener(new MouseListener() {  @Override  public void mouseClicked(MouseEvent e) {  }   @Override  public void mousePressed(MouseEvent e) {  }   @Override  public void mouseReleased(MouseEvent e) {  }   @Override  public void mouseEntered(MouseEvent e) {  System.*out*.println("Добро пожаловать в Северный район");  }   @Override  public void mouseExited(MouseEvent e) {  }  });  add(northPanel, BorderLayout.*NORTH*);  // Добавление обработчиков мыши для южной области окна  JPanel southPanel = new JPanel();  southPanel.addMouseListener(new MouseListener() {  @Override  public void mouseClicked(MouseEvent e) {  }   @Override  public void mousePressed(MouseEvent e) {  }   @Override  public void mouseReleased(MouseEvent e) {  }  @Override  public void mouseEntered(MouseEvent e) {  System.*out*.println("Добро пожаловать в Южный район");  }   @Override  public void mouseExited(MouseEvent e) {  }  });  add(southPanel, BorderLayout.*SOUTH*);  // Добавление обработчиков мыши для западной области окна  JPanel westPanel = new JPanel();  westPanel.addMouseListener(new MouseListener() {  @Override  public void mouseClicked(MouseEvent e) {  }   @Override  public void mousePressed(MouseEvent e) {  }   @Override  public void mouseReleased(MouseEvent e) {  }   @Override  public void mouseEntered(MouseEvent e) {  System.*out*.println("Добро пожаловать в Западный район");  }   @Override  public void mouseExited(MouseEvent e) {  }  });  add(westPanel, BorderLayout.*WEST*);  // Добавление обработчиков мыши для восточной области окна  JPanel eastPanel = new JPanel();  eastPanel.addMouseListener(new MouseListener() {  @Override  public void mouseClicked(MouseEvent e) {  }   @Override  public void mousePressed(MouseEvent e) {  }   @Override  public void mouseReleased(MouseEvent e) {  }   @Override  public void mouseEntered(MouseEvent e) {  System.*out*.println("Добро пожаловать в Восточный район");  }   @Override  public void mouseExited(MouseEvent e) {  }  });  add(eastPanel, BorderLayout.*EAST*);    // ... (остальные обработчики мыши для других областей)   pack();  setLocationRelativeTo(null);  setVisible(true);  }   private void showEndGameMessage(String message) {  JOptionPane.*showMessageDialog*(this, message, "Конец игры", JOptionPane.*INFORMATION\_MESSAGE*);  System.*exit*(0);  }   public static void main(String[] args) {  SwingUtilities.*invokeLater*(() -> new GuessingGame());  } }  import javax.swing.\*; import java.awt.\*; import java.awt.event.\*;  public class TextEditor extends JFrame {  private JTextArea textArea;  private JMenu colorMenu;  private JMenu fontMenu;   public TextEditor() {  textArea = new JTextArea(10, 20);   JMenuBar menuBar = new JMenuBar();  colorMenu = new JMenu("Цвет");  fontMenu = new JMenu("Шрифт");   JMenuItem blueItem = new JMenuItem("Синий");  JMenuItem redItem = new JMenuItem("Красный");  JMenuItem blackItem = new JMenuItem("Черный");   JMenuItem timesNewRomanItem = new JMenuItem("Times New Roman");  JMenuItem msSansSerifItem = new JMenuItem("MS Sans Serif");  JMenuItem courierNewItem = new JMenuItem("Courier New");   blueItem.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  textArea.setForeground(Color.*BLUE*);  }  });   redItem.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  textArea.setForeground(Color.*RED*);  }  });   blackItem.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  textArea.setForeground(Color.*BLACK*);  }  });   timesNewRomanItem.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  textArea.setFont(new Font("Times New Roman", Font.*PLAIN*, 12));  }  });   msSansSerifItem.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  textArea.setFont(new Font("MS Sans Serif", Font.*PLAIN*, 12));  }  });   courierNewItem.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  textArea.setFont(new Font("Courier New", Font.*PLAIN*, 12));  }  });   colorMenu.add(blueItem);  colorMenu.add(redItem);  colorMenu.add(blackItem);   fontMenu.add(timesNewRomanItem);  fontMenu.add(msSansSerifItem);  fontMenu.add(courierNewItem);   menuBar.add(colorMenu);  menuBar.add(fontMenu);   setJMenuBar(menuBar);  add(new JScrollPane(textArea));   setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  pack();  setVisible(true);  }   public static void main(String[] args) {  SwingUtilities.*invokeLater*(new Runnable() {  public void run() {  new TextEditor();  }  });  } }  import javax.swing.\*; import java.awt.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener;  public class PasswordChecker {   public PasswordChecker() {  JFrame frame = new JFrame("Password Checker");  frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  frame.setSize(300, 200);   JPanel panel = new JPanel();  panel.setLayout(new BoxLayout(panel, BoxLayout.*Y\_AXIS*));   JLabel serviceLabel = new JLabel("Service: ");  JTextField serviceField = new JTextField(15);   JLabel usernameLabel = new JLabel("User name: ");  JTextField usernameField = new JTextField(15);   JLabel passwordLabel = new JLabel("Password: ");  JPasswordField passwordField = new JPasswordField(15);   panel.add(serviceLabel);  panel.add(serviceField);  panel.add(usernameLabel);  panel.add(usernameField);  panel.add(passwordLabel);  panel.add(passwordField);   JButton checkButton = new JButton("Check Password");  checkButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  char[] enteredPassword = passwordField.getPassword();  String enteredPasswordString = new String(enteredPassword);   if (checkPassword(enteredPasswordString)) {  JOptionPane.*showMessageDialog*(frame, "Correct Password");  } else {  JOptionPane.*showMessageDialog*(frame, "Incorrect Password");  }  }  });   panel.add(checkButton);   frame.add(panel);  frame.setVisible(true);  }   private boolean checkPassword(String enteredPassword) {  // Ваш пароль для сравнения  String correctPassword = "Dimirtii\_Tkachenko";   return enteredPassword.equals(correctPassword);  }   public static void main(String[] args) {  SwingUtilities.*invokeLater*(() -> new PasswordChecker());  } } |

Результат:







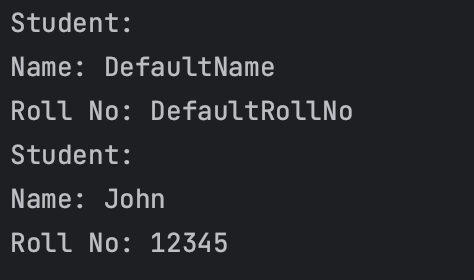
**Практическая работа № 17. Разработка интерактивных программ на языке Джава с использованием паттерна MVC**

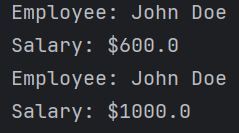
**Цель:** введение в разработку программ c с использованием событийного программирования на языке программирования Джава с использованием паттерна MVC.

Код:

|  |
| --- |
| class Student {  private String rollNo;  private String name;   public String getRollNo() {  return rollNo;  }   public void setRollNo(String rollNo) {  this.rollNo = rollNo;  }   public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  } }  class StudentView {  public void printStudentDetails(String studentName, String studentRollNo) {  System.*out*.println("Student: ");  System.*out*.println("Name: " + studentName);  System.*out*.println("Roll No: " + studentRollNo);  } }  class StudentController {  private Student model;  private StudentView view;   public StudentController(Student model, StudentView view) {  this.model = model;  this.view = view;  }   public void setStudentName(String name) {  model.setName(name);  }   public String getStudentName() {  return model.getName();  }   public void setStudentRollNo(String rollNo) {  model.setRollNo(rollNo);  }   public String getStudentRollNo() {  return model.getRollNo();  }   public void updateView() {  view.printStudentDetails(model.getName(), model.getRollNo());  } }  public class MVCPatternDemo {  public static void main(String[] args) {  // Получение данных из базы данных (заглушка)  Student model = *retrieveStudentFromDatabase*();   // Создание вида и контроллера  StudentView view = new StudentView();  StudentController controller = new StudentController(model, view);   // Обновление вида  controller.updateView();   // Обновление модели и снова обновление вида  controller.setStudentName("John");  controller.setStudentRollNo("12345");  controller.updateView();  }   private static Student retrieveStudentFromDatabase() {  // Заглушка для получения данных из базы данных  Student student = new Student();  student.setName("DefaultName");  student.setRollNo("DefaultRollNo");  return student;  } }  // Класс Employee – слой модели public class Employee {  private String name;  private double hourlyRate;  private int hoursWorked;   public Employee(String name, double hourlyRate, int hoursWorked) {  this.name = name;  this.hourlyRate = hourlyRate;  this.hoursWorked = hoursWorked;  }   public double calculateSalary() {  return hourlyRate \* hoursWorked;  }   // Геттеры и сеттеры  public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   public double getHourlyRate() {  return hourlyRate;  }   public void setHourlyRate(double hourlyRate) {  this.hourlyRate = hourlyRate;  }   public int getHoursWorked() {  return hoursWorked;  }   public void setHoursWorked(int hoursWorked) {  this.hoursWorked = hoursWorked;  } }  // Пример использования public class MVCEmployeeExample {  public static void main(String[] args) {  // Создаем объекты модели, представления и контроллера  Employee model = new Employee("John Doe", 15.0, 40);  EmployeeView view = new EmployeeView();  EmployeeController controller = new EmployeeController(model, view);   // Выводим начальные данные  controller.updateView();   // Обновляем данные и выводим результат  controller.setEmployeeHourlyRate(20.0);  controller.setEmployeeHoursWorked(50);  controller.updateView();  } }  // Класс EmployeeController – уровень контроллера class EmployeeController {  private Employee model;  private EmployeeView view;   public EmployeeController(Employee model, EmployeeView view) {  this.model = model;  this.view = view;  }   public void setEmployeeName(String name) {  model.setName(name);  }   public void setEmployeeHourlyRate(double hourlyRate) {  model.setHourlyRate(hourlyRate);  }   public void setEmployeeHoursWorked(int hoursWorked) {  model.setHoursWorked(hoursWorked);  }   public void updateView() {  view.printEmployeeDetails(model.getName(), model.calculateSalary());  } }  // Класс EmployeeView – слой представления class EmployeeView {  public void printEmployeeDetails(String name, double salary) {  System.*out*.println("Employee: " + name);  System.*out*.println("Salary: $" + salary);  } }  class Employee {  private String name;  private double hourlyRate;  private int hoursWorked;   public Employee(String name, double hourlyRate, int hoursWorked) {  this.name = name;  this.hourlyRate = hourlyRate;  this.hoursWorked = hoursWorked;  }   public double calculateSalary() {  return hourlyRate \* hoursWorked;  }   // Геттеры и сеттеры  public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   public double getHourlyRate() {  return hourlyRate;  }   public void setHourlyRate(double hourlyRate) {  this.hourlyRate = hourlyRate;  }   public int getHoursWorked() {  return hoursWorked;  }   public void setHoursWorked(int hoursWorked) {  this.hoursWorked = hoursWorked;  } }  public class Student {  private String name;  private int age;   public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   public int getAge() {  return age;  }   public void setAge(int age) {  this.age = age;  } }  public class StudentController {  private Student model;  private StudentView view;   public StudentController(Student model, StudentView view) {  this.model = model;  this.view = view;  }   public void setStudentName(String name) {  model.setName(name);  }   public void setStudentAge(int age) {  model.setAge(age);  }   public void updateView() {  view.setStudentDetails(model.getName(), model.getAge());  } }  import javax.swing.\*; public class StudentView {  private JFrame frame;  private JLabel nameLabel;  private JLabel ageLabel;   public StudentView() {  frame = new JFrame("Student View");  nameLabel = new JLabel("Name:");  ageLabel = new JLabel("Age:");   frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  frame.setSize(300, 200);  frame.setLayout(new BoxLayout(frame.getContentPane(), BoxLayout.*Y\_AXIS*));   frame.add(nameLabel);  frame.add(ageLabel);  frame.setVisible(true);  }   public void setStudentDetails(String name, int age) {  nameLabel.setText("Name: " + name);  ageLabel.setText("Age: " + age);  } }  public class MVCPatternDemo {  public static void main(String[] args) {  Student model = new Student();  model.setName("Vsevolod");  model.setAge(25);   StudentView view = new StudentView();   StudentController controller = new StudentController(model, view);   controller.updateView();   // Обновление модели и представления  controller.setStudentName("Ivan");  controller.setStudentAge(30);  controller.updateView();  } } |

Результат:





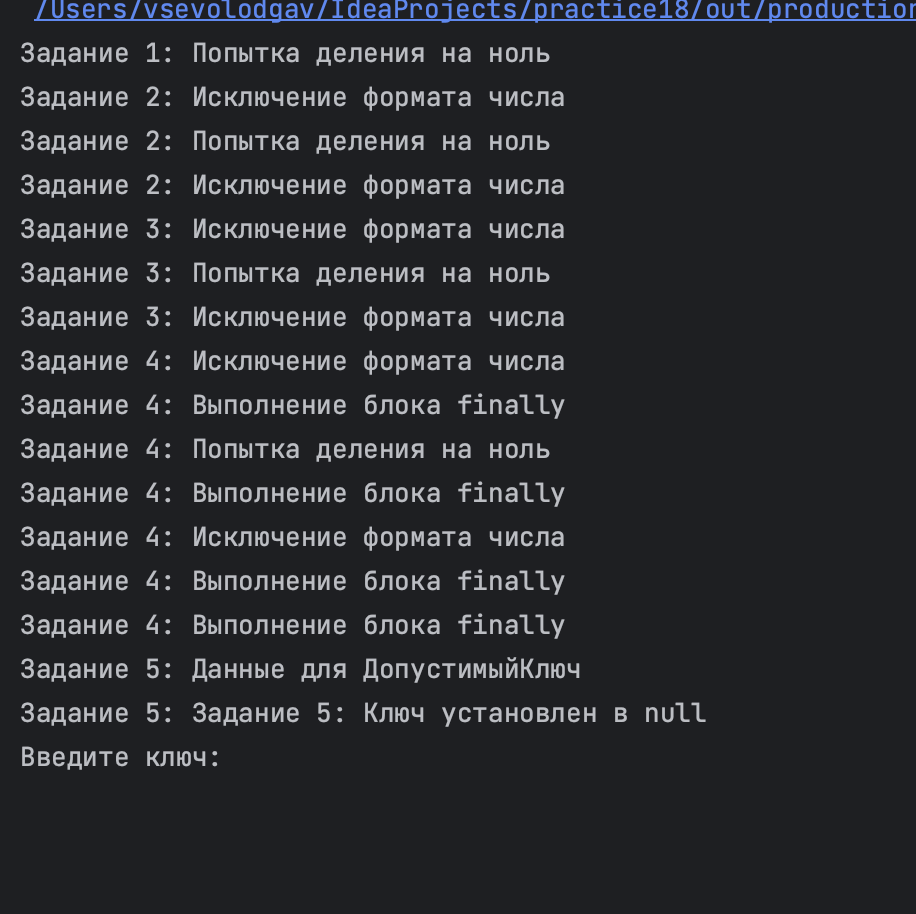
**Практическая работа № 18. Исключения и работа с ними в Джава**

**Цель** данной практической работы являются получение практических навыков разработки программ, изучение синтаксиса языка Java, освоение основных конструкций языка Java (циклы, условия, создание переменных и массивов, создание методов, вызов методов), а также научиться осуществлять стандартный ввод/вывод данных.

Код:

|  |
| --- |
| import java.util.Scanner;  public class ExceptionHandlingDemo {  public static void main(String[] args) {  try {  // Задание №1 - Обработка деления на ноль  Exception1 task1 = new Exception1();  task1.exceptionDemo();  } catch (ArithmeticException e) {  System.*out*.println("Задание 1: " + e.getMessage());  }   try {  // Задание №2 - Обработка пользовательских исключений  Exception2 task2 = new Exception2();  task2.exceptionDemo("Qwerty");  task2.exceptionDemo("0");  task2.exceptionDemo("1.2");  } catch (NumberFormatException e) {  System.*out*.println("Задание 2: Исключение формата числа");  } catch (ArithmeticException e) {  System.*out*.println("Задание 2: Попытка деления на ноль");  }   try {  // Задание №3 - Добавление catch в начало списка catch  Exception3 task3 = new Exception3();  task3.exceptionDemo("Qwerty");  task3.exceptionDemo("0");  task3.exceptionDemo("1.2");  } catch (Exception e) {  System.*out*.println("Задание 3: Общее исключение");  }   try {  // Задание №4 - Добавление блока finally  Exception4 task4 = new Exception4();  task4.exceptionDemo("Qwerty");  task4.exceptionDemo("0");  task4.exceptionDemo("1.2");  } catch (NumberFormatException e) {  System.*out*.println("Задание 4: Исключение формата числа");  } catch (ArithmeticException e) {  System.*out*.println("Задание 4: Попытка деления на ноль");  } finally {  System.*out*.println("Задание 4: Выполнение блока finally");  }   try {  // Задание №5 - Генерация собственных исключений  Exception5 task5 = new Exception5();  task5.printMessage("ДопустимыйКлюч");  task5.printMessage(null);  } catch (Exception e) {  System.*out*.println("Задание 5: " + e.getMessage());  }   try {  // Задание №6 - Прохождение исключения через цепочку вызовов  Exception6 task6 = new Exception6();  task6.getKey();  } catch (Exception e) {  System.*out*.println("Задание 6: " + e.getMessage());  }  } }  class Exception1 {  // Задание №1 - Обработка деления на ноль  public void exceptionDemo() {  try {  System.*out*.println(2 / 0);  } catch (ArithmeticException e) {  System.*out*.println("Задание 1: Попытка деления на ноль");  }  } }  class Exception2 {  // Задание №2 - Обработка пользовательских исключений  public void exceptionDemo(String input) {  try {  int number = Integer.*parseInt*(input);  System.*out*.println(2 / number);  } catch (NumberFormatException e) {  System.*out*.println("Задание 2: Исключение формата числа");  } catch (ArithmeticException e) {  System.*out*.println("Задание 2: Попытка деления на ноль");  }  } }  class Exception3 {  // Задание №3 - Добавление catch в начало списка catch  public void exceptionDemo(String input) {  try {  int number = Integer.*parseInt*(input);  System.*out*.println(2 / number);  } catch (NumberFormatException e) {  System.*out*.println("Задание 3: Исключение формата числа");  } catch (ArithmeticException e) {  System.*out*.println("Задание 3: Попытка деления на ноль");  } catch (Exception e) {  System.*out*.println("Задание 3: Общее исключение");  }  } }  class Exception4 {  // Задание №4 - Добавление блока finally  public void exceptionDemo(String input) {  try {  int number = Integer.*parseInt*(input);  System.*out*.println(2 / number);  } catch (NumberFormatException e) {  System.*out*.println("Задание 4: Исключение формата числа");  } catch (ArithmeticException e) {  System.*out*.println("Задание 4: Попытка деления на ноль");  } finally {  System.*out*.println("Задание 4: Выполнение блока finally");  }  } }  class Exception5 {  // Задание №5 - Генерация собственных исключений  public void printMessage(String key) throws Exception {  String message = getDetails(key);  System.*out*.println(message);  }   private String getDetails(String key) throws Exception {  if (key == null) {  throw new Exception("Задание 5: Ключ установлен в null");  }  return "Задание 5: Данные для " + key;  } }  class Exception6 {  // Задание №6 - Прохождение исключения через цепочку вызовов  public void getKey() {  System.*out*.print("Введите ключ: ");  Scanner myScanner = new Scanner(System.*in*);  String key = myScanner.next();  try {  printDetails(key);  } catch (Exception e) {  System.*out*.println("Задание 6: " + e.getMessage());  }  }   public void printDetails(String key) throws Exception {  String message = getDetails(key);  System.*out*.println(message);  }   private String getDetails(String key) throws Exception {  if (key.equals("")) {  throw new Exception("Задание 6: Ключ установлен как пустая строка");  }  return "Задание 6: Данные для " + key;  } } |

Результат:



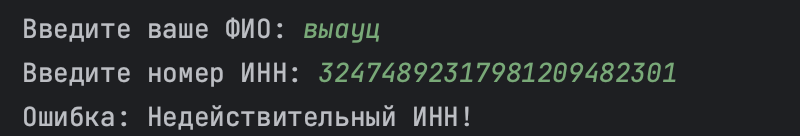
**Практическая работа № 19. Создание пользовательских исключений**

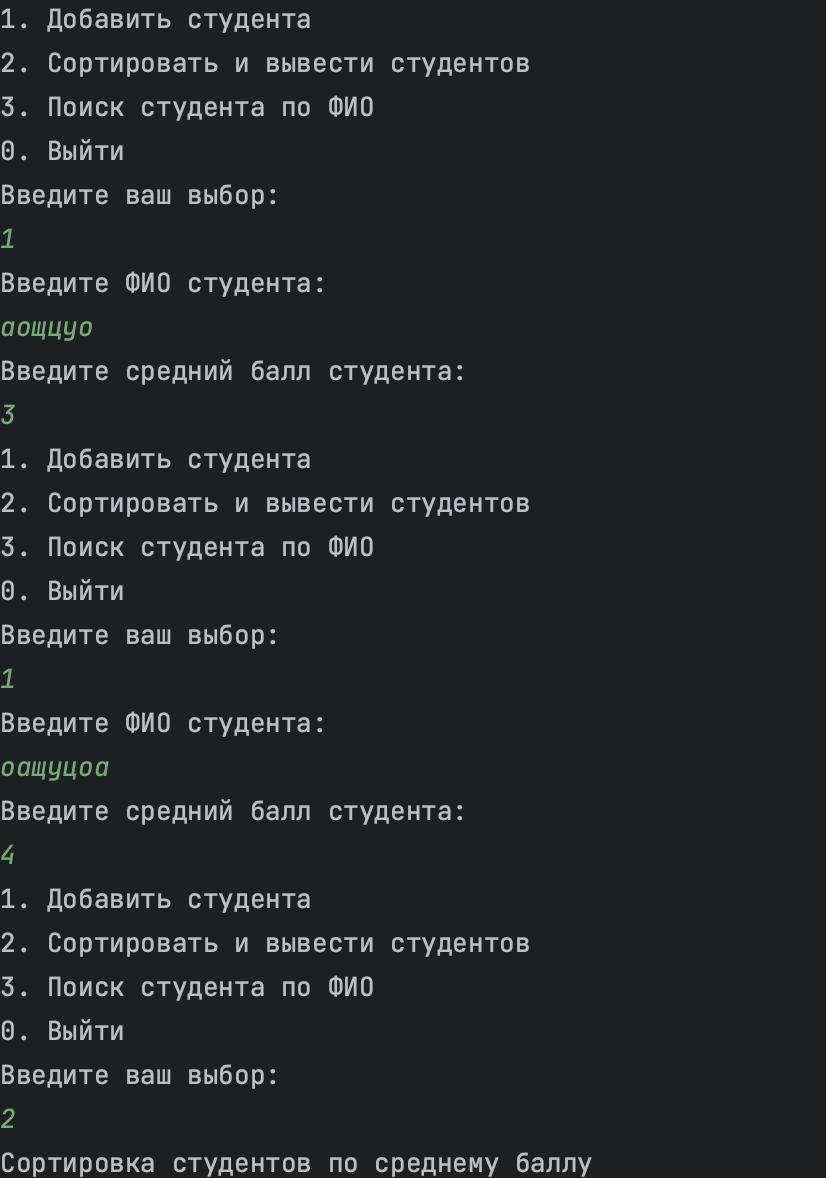
**Цель** данной практической работы – научиться создавать собственные исключения.

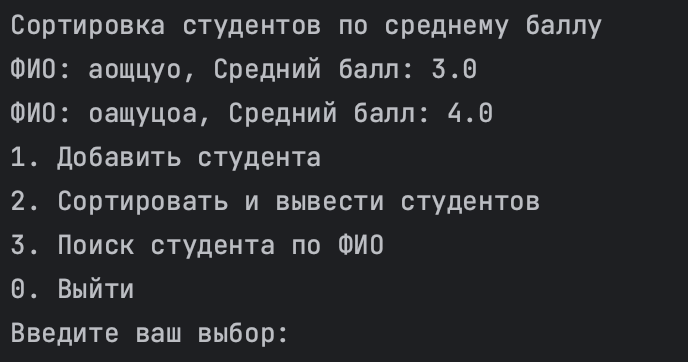
Код:

|  |
| --- |
| import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);   System.*out*.print("Введите ваше ФИО: ");  String fullName = scanner.nextLine();   System.*out*.print("Введите номер ИНН: ");  String inn = scanner.nextLine();   try {  if (*isValidINN*(inn)) {  System.*out*.println("ИНН действителен. Заказ оформлен для: " + fullName);  } else {  throw new Exception("Недействительный ИНН!");  }  } catch (Exception e) {  System.*out*.println("Ошибка: " + e.getMessage());  }  }   public static boolean isValidINN(String inn) {  // считаем, что ИНН действителен, если он состоит из 10 или 12 цифр  return inn.matches("\\d{10}|\\d{12}");  } }  import java.util.ArrayList; import java.util.Comparator; import java.util.List;  public class Student implements Comparable<Student> {   private String fio;  private double averageScore;   public Student(String fio, double averageScore) {  this.fio = fio;  this.averageScore = averageScore;  }   public String getFio() {  return fio;  }   public void setFio(String fio) {  this.fio = fio;  }   public double getAverageScore() {  return averageScore;  }   public void setAverageScore(double averageScore) {  this.averageScore = averageScore;  }   @Override  public int compareTo(Student o) {  return Double.*compare*(this.averageScore, o.averageScore);  } }  import java.util.ArrayList; import java.util.Comparator; import java.util.List;  public class LabClass {   private List<Student> students;   public LabClass() {  students = new ArrayList<>();  }   public void addStudent(Student student) {  students.add(student);  }   public void sort() {  students.sort(Comparator.*comparingDouble*(Student::getAverageScore));  }   public Student findStudentByFio(String fio) throws StudentNotFoundException {  for (Student student : students) {  if (student.getFio().equals(fio)) {  return student;  }  }  throw new StudentNotFoundException("Студент с таким ФИО не найден");  }   public List<Student> getStudents() {  return students;  } }  import java.util.Scanner;  public class LabClassUI {   private LabClass labClass;   public LabClassUI() {  labClass = new LabClass();  }   public void addStudent() {  Scanner scanner = new Scanner(System.*in*);   System.*out*.println("Введите ФИО студента:");  String fio = scanner.nextLine();   double averageScore = 0.0;   boolean validInput = false;  while (!validInput) {  try {  System.*out*.println("Введите средний балл студента:");  averageScore = scanner.nextDouble();  validInput = true;  } catch (java.util.InputMismatchException e) {  System.*out*.println("Ошибка ввода. Введите числовое значение для среднего балла.");  scanner.nextLine(); // Очистка буфера после ошибочного ввода  }  }   scanner.nextLine(); // Очистка буфера   labClass.addStudent(new Student(fio, averageScore));  }   public void sortAndPrint() {  System.*out*.println("Сортировка студентов по среднему баллу");  labClass.sort();   for (Student student : labClass.getStudents()) {  System.*out*.println("ФИО: " + student.getFio() + ", Средний балл: " + student.getAverageScore());  }  }   public void findStudentByFio() {  Scanner scanner = new Scanner(System.*in*);   System.*out*.println("Поиск студента по ФИО");  System.*out*.println("Введите ФИО студента:");  String fio = scanner.nextLine();   try {  Student student = labClass.findStudentByFio(fio);  System.*out*.println("Студент с ФИО " + student.getFio() + " найден");  } catch (StudentNotFoundException e) {  System.*out*.println(e.getMessage());  }  }   public void run() {  Scanner scanner = new Scanner(System.*in*);  int choice;  do {  System.*out*.println("1. Добавить студента");  System.*out*.println("2. Сортировать и вывести студентов");  System.*out*.println("3. Поиск студента по ФИО");  System.*out*.println("0. Выйти");  System.*out*.println("Введите ваш выбор:");  choice = scanner.nextInt();   switch (choice) {  case 1:  addStudent();  break;  case 2:  sortAndPrint();  break;  case 3:  findStudentByFio();  break;  case 0:  System.*out*.println("Выход из программы");  break;  default:  System.*out*.println("Неверный выбор. Попробуйте еще раз.");  }  } while (choice != 0);  scanner.close();  }   public static void main(String[] args) {  LabClassUI ui = new LabClassUI();  ui.run();  } }  public class StudentNotFoundException extends Exception {   public StudentNotFoundException(String message) {  super(message);  } } |

Результат:







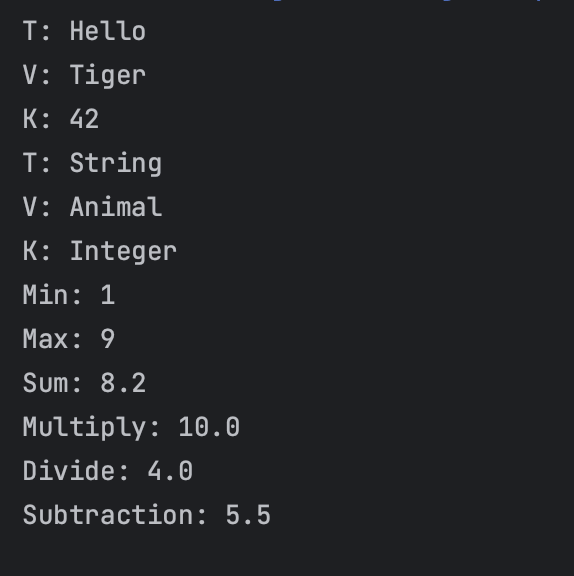
**Практическая работа № 20. Работа с дженериками.**

**Цель** данной практической работы – научиться работать с обобщенными типами в Java и применять их в программах.

Код:

|  |
| --- |
| public class MatrixExample {  public static void main(String[] args) {  // Создаем матрицы  Integer[][] matrix1 = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};  Integer[][] matrix2 = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};   Matrix<Integer> matrixObj1 = new Matrix<>(matrix1);  Matrix<Integer> matrixObj2 = new Matrix<>(matrix2);   // Сложение матриц  Matrix<Integer> sumMatrix = matrixObj1.add(matrixObj2);  System.*out*.println("Sum of matrices:");  sumMatrix.printMatrix();   // Умножение матриц  Matrix<Integer> productMatrix = matrixObj1.multiply(matrixObj2);  System.*out*.println("Product of matrices:");  productMatrix.printMatrix();  } }  // Класс Matrix с реализацией операций class Matrix<T> {  private T[][] matrix;   // Конструктор с матрицей  public Matrix(T[][] matrix) {  this.matrix = matrix;  }   // Метод для сложения матриц  public Matrix<T> add(Matrix<T> other) {  if (matrix.length != other.matrix.length || matrix[0].length != other.matrix[0].length) {  throw new IllegalArgumentException("Matrices must have the same dimensions for addition");  }   T[][] result = (T[][]) new Object[matrix.length][matrix[0].length];   for (int i = 0; i < matrix.length; i++) {  for (int j = 0; j < matrix[0].length; j++) {  // Предполагаем, что тип T поддерживает сложение  // (например, можно использовать только для числовых типов)  result[i][j] = (T) (Number) (((Number) matrix[i][j]).doubleValue() + ((Number) other.matrix[i][j]).doubleValue());  }  }   return new Matrix<>(result);  }   public void printMatrix() {  for (T[] row : matrix) {  for (T element : row) {  System.*out*.print(element + " ");  }  System.*out*.println();  }  }  // Метод для умножения матриц  public Matrix<T> multiply(Matrix<T> other) {  if (matrix[0].length != other.matrix.length) {  throw new IllegalArgumentException("Number of columns in the first matrix must be equal to the number of rows in the second matrix for multiplication");  }   T[][] result = (T[][]) new Object[matrix.length][other.matrix[0].length];   for (int i = 0; i < matrix.length; i++) {  for (int j = 0; j < other.matrix[0].length; j++) {  result[i][j] = multiplyRowColumn(matrix, i, other.matrix, j);  }  }   return new Matrix<>(result);  }   // Вспомогательный метод для умножения строки на столбец  private T multiplyRowColumn(T[][] matrix1, int row, T[][] matrix2, int col) {  T result = (T) (Number) 0;   for (int k = 0; k < matrix1[0].length; k++) {  result = (T) (Number) (((Number) result).doubleValue() + ((Number) matrix1[row][k]).doubleValue() \* ((Number) matrix2[k][col]).doubleValue());  }   return result;  } }  import java.io.Serializable;  // Пример простого класса Animal class Animal implements Serializable {  private String name;   public Animal(String name) {  this.name = name;  }   public String getName() {  return name;  } }  public class Main {  public static void main(String[] args) {  // Пример использования MyGenericClass с классом Animal  MyGenericClass<String, Animal, Integer> genericObj = new MyGenericClass<>("Hello", new Animal("Tiger"), 42);  System.*out*.println("T: " + genericObj.getT());  System.*out*.println("V: " + genericObj.getV().getName());  System.*out*.println("K: " + genericObj.getK());  genericObj.printClassNames();   // Пример использования MinMax  Integer[] intArray = {3, 7, 1, 9, 4};  MinMax<Integer> minMaxObj = new MinMax<>(intArray);  System.*out*.println("Min: " + minMaxObj.findMin());  System.*out*.println("Max: " + minMaxObj.findMax());   // Пример использования Calculator  double resultSum = Calculator.*sum*(5, 3.2);  double resultMultiply = Calculator.*multiply*(2.5, 4);  double resultDivide = Calculator.*divide*(8, 2);  double resultSubtraction = Calculator.*subtraction*(10, 4.5);   System.*out*.println("Sum: " + resultSum);  System.*out*.println("Multiply: " + resultMultiply);  System.*out*.println("Divide: " + resultDivide);  System.*out*.println("Subtraction: " + resultSubtraction);  } }  import java.io.Serializable;  // Обобщенный класс с тремя параметрами (T, V, K) class MyGenericClass<T extends Comparable<T>, V extends Animal & Serializable, K> {  private T t;  private V v;  private K k;   // Конструктор  public MyGenericClass(T t, V v, K k) {  this.t = t;  this.v = v;  this.k = k;  }   // Методы возвращающие значения трех переменных  public T getT() {  return t;  }   public V getV() {  return v;  }   public K getK() {  return k;  }   // Метод выводящий на консоль имена классов для трех переменных класса  public void printClassNames() {  System.*out*.println("T: " + t.getClass().getSimpleName());  System.*out*.println("V: " + v.getClass().getSimpleName());  System.*out*.println("K: " + k.getClass().getSimpleName());  } }  // Обобщенный класс MinMax class MinMax<T extends Comparable<T>> {  private T[] array;   // Конструктор с массивом  public MinMax(T[] array) {  this.array = array;  }   // Метод для нахождения минимального элемента  public T findMin() {  T min = array[0];  for (T element : array) {  if (element.compareTo(min) < 0) {  min = element;  }  }  return min;  }   // Метод для нахождения максимального элемента  public T findMax() {  T max = array[0];  for (T element : array) {  if (element.compareTo(max) > 0) {  max = element;  }  }  return max;  } }  // Класс Калькулятор class Calculator {  // Обобщенные статические методы  public static <T extends Number, U extends Number> double sum(T num1, U num2) {  return num1.doubleValue() + num2.doubleValue();  }   public static <T extends Number, U extends Number> double multiply(T num1, U num2) {  return num1.doubleValue() \* num2.doubleValue();  }   public static <T extends Number, U extends Number> double divide(T num1, U num2) {  if (num2.doubleValue() != 0) {  return num1.doubleValue() / num2.doubleValue();  } else {  throw new IllegalArgumentException("Cannot divide by zero");  }  }   public static <T extends Number, U extends Number> double subtraction(T num1, U num2) {  return num1.doubleValue() - num2.doubleValue();  } } |

Результат:



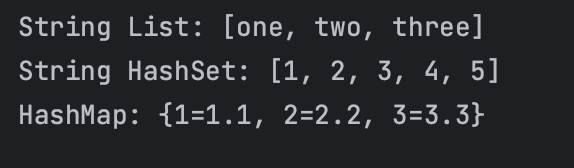
**Практическая работа № 21. Стирание типов в Джава**

**Цель** данной практической работы – научиться работать с обобщенными типами в Java и применять прием стирание типов разработке программ на Джава

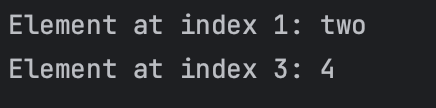
Код:

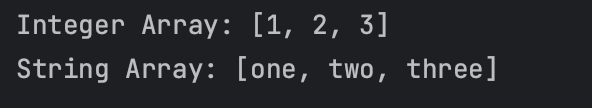
|  |
| --- |
| import java.util.ArrayList; import java.util.HashMap; import java.util.HashSet; import java.util.List; import java.util.Set;  public class Solution<T, K, V> {  private T genericObject;  private K keyObject;  private V valueObject;   public Solution(T genericObject, K keyObject, V valueObject) {  this.genericObject = genericObject;  this.keyObject = keyObject;  this.valueObject = valueObject;  }   public List<T> newArrayList(T... elements) {  return new ArrayList<>(List.*of*(elements));  }   public HashSet<T> newHashSet(T... elements) {  return new HashSet<>(Set.*of*(elements));  }   public HashMap<K, V> newHashMap(List<K> keys, List<V> values) {  HashMap<K, V> hashMap = new HashMap<>();  int size = Math.*min*(keys.size(), values.size());   for (int i = 0; i < size; i++) {  hashMap.put(keys.get(i), values.get(i));  }   return hashMap;  }   public static void main(String[] args) {  Solution<String, Integer, Double> solution = new Solution<>("Generic", 42, 3.14);   List<String> stringList = solution.newArrayList("one", "two", "three");  System.*out*.println("String List: " + stringList);   HashSet<String> stringHashSet = solution.newHashSet("1", "2", "3", "4", "5");  System.*out*.println("String HashSet: " + stringHashSet);   List<Integer> keys = List.*of*(1, 2, 3);  List<Double> values = List.*of*(1.1, 2.2, 3.3);  HashMap<Integer, Double> hashMap = solution.newHashMap(keys, values);  System.*out*.println("HashMap: " + hashMap);  } }  import java.io.File; import java.util.Arrays; import java.util.List;  public class DirectoryLister {  public static List<String> listDirectoryContents(String directoryPath) {  File directory = new File(directoryPath);  String[] files = directory.list();   if (files != null) {  return Arrays.*asList*(files);  } else {  return null;  }  }   public static void main(String[] args) {  String directoryPath = "/path/to/your/directory";  List<String> fileList = *listDirectoryContents*(directoryPath);   if (fileList != null) {  System.*out*.println("First 5 elements: " + fileList.subList(0, Math.*min*(5, fileList.size())));  } else {  System.*out*.println("Directory not found or empty.");  }  } }  public class ArrayAccessor {  public static <T> T getElementAtIndex(T[] array, int index) {  if (index >= 0 && index < array.length) {  return array[index];  }  return null;  }   public static void main(String[] args) {  String[] stringArray = {"one", "two", "three"};  System.*out*.println("Element at index 1: " + *getElementAtIndex*(stringArray, 1));   Integer[] intArray = {1, 2, 3, 4, 5};  System.*out*.println("Element at index 3: " + *getElementAtIndex*(intArray, 3));  } }  import java.util.Arrays;  public class GenericArrayHolder<T> {  private T[] array;   public GenericArrayHolder(T[] array) {  this.array = array;  }   public T[] getArray() {  return array;  }   public static void main(String[] args) {  Integer[] intArray = {1, 2, 3};  GenericArrayHolder<Integer> intHolder = new GenericArrayHolder<>(intArray);  System.*out*.println("Integer Array: " + Arrays.*toString*(intHolder.getArray()));   String[] stringArray = {"one", "two", "three"};  GenericArrayHolder<String> stringHolder = new GenericArrayHolder<>(stringArray);  System.*out*.println("String Array: " + Arrays.*toString*(stringHolder.getArray()));  } }  import java.util.ArrayList; import java.util.Arrays; import java.util.List;  public class ArrayConverter {  public static <T> List<T> convertArrayToList(T[] array) {  return new ArrayList<>(Arrays.*asList*(array));  }   public static void main(String[] args) {  String[] stringArray = {"one", "two", "three"};  List<String> stringList = *convertArrayToList*(stringArray);  System.*out*.println("String List: " + stringList);   Integer[] intArray = {1, 2, 3, 4, 5};  List<Integer> intList = *convertArrayToList*(intArray);  System.*out*.println("Integer List: " + intList);  } } |

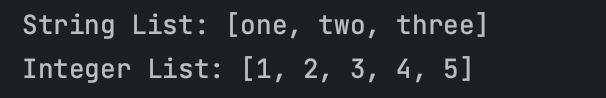
Результат:











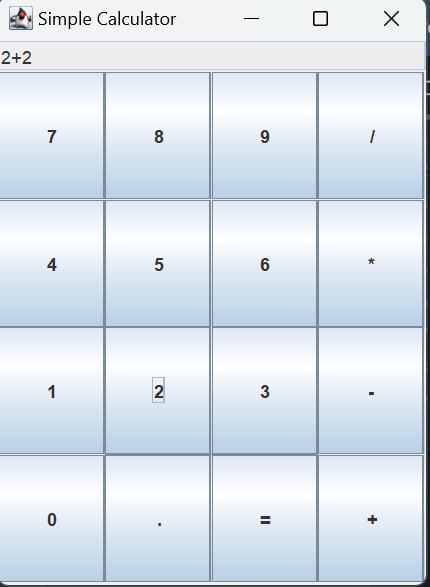
**Практическая работа № 22. Абстрактные типы данных. Стек**

**Цель** данной практической работы – научиться разрабатывать программы с абстрактными типами данных на языке Джава и применять паттерн MVC при разработке программ

Код:

|  |
| --- |
| import javax.swing.\*;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.util.Stack;  public class CalculatorGUI extends JFrame {  private JTextField display;  public CalculatorGUI() {  setTitle("Simple Calculator");  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  setSize(300, 400);  setLayout(new BorderLayout());  display = new JTextField();  display.setEditable(false);  add(display, BorderLayout.NORTH);  JPanel buttonPanel = new JPanel();  buttonPanel.setLayout(new GridLayout(4, 4));  String[] buttonLabels = {  "7", "8", "9", "/",  "4", "5", "6", "\*",  "1", "2", "3", "-",  "0", ".", "=", "+"  };  for (String label : buttonLabels) {  JButton button = new JButton(label);  button.addActionListener(new ButtonClickListener());  buttonPanel.add(button);  }  add(buttonPanel, BorderLayout.CENTER);  setVisible(true);  }  private class ButtonClickListener implements ActionListener {  @Override  public void actionPerformed(ActionEvent e) {  JButton source = (JButton) e.getSource();  String buttonText = source.getText();  if (buttonText.equals("=")) {  evaluateExpression();  } else {  display.setText(display.getText() + buttonText);  }  }  private void evaluateExpression() {  String expression = display.getText();  try {  double result = evaluatePostfix(convertToPostfix(expression));  display.setText(Double.toString(result));  } catch (Exception ex) {  display.setText("Error");  }  }  private String convertToPostfix(String infix) {  StringBuilder postfix = new StringBuilder();  Stack<Character> stack = new Stack<>();  for (char ch : infix.toCharArray()) {  if (Character.isDigit(ch) ch == '.') {  postfix.append(ch);  } else if (ch == '(') {  stack.push(ch);  } else if (ch == ')') {  while (!stack.isEmpty() && stack.peek() != '(') {  postfix.append(stack.pop());  }  stack.pop(); // Pop '('  } else {  while (!stack.isEmpty() && getPrecedence(ch) <= getPrecedence(stack.peek())) {  postfix.append(stack.pop());  }  stack.push(ch);  }  }  while (!stack.isEmpty()) {  postfix.append(stack.pop());  }  return postfix.toString();  }  private int getPrecedence(char operator) {  if (operator == '+' operator == '-') {  return 1;  } else if (operator == '\*' operator == '/') {  return 2;  }  return 0; // For '('  }  private double evaluatePostfix(String postfix) {  Stack<Double> stack = new Stack<>();  for (char ch : postfix.toCharArray()) {  if (Character.isDigit(ch) ch == '.') {  stack.push(Double.parseDouble(Character.toString(ch)));  } else {  double operand2 = stack.pop();  double operand1 = stack.pop();  switch (ch) {  case '+':  stack.push(operand1 + operand2);  break;  case '-':  stack.push(operand1 - operand2);  break;  case '\*':  stack.push(operand1 \* operand2);  break;  case '/':  stack.push(operand1 / operand2);  break;  }  }  }  return stack.pop();  }  }  public static void main(String[] args) {  SwingUtilities.invokeLater(() -> new CalculatorGUI());  }  } |

Результат:



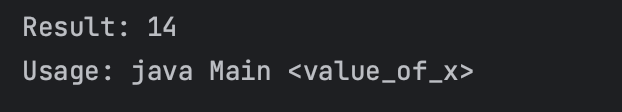
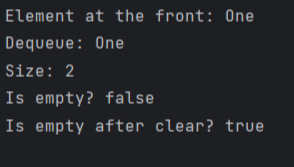
**Практическая работа № 23. Абстрактные типы данных. Очередь**

**Цель:** цель данной практической работы – – научиться разрабатывать программы с абстрактными типами данных на языке Джава

Код:

|  |
| --- |
| import java.util.Arrays;  abstract class ArrayQueueADT implements Queue {  protected Object[] elements = new Object[10];  protected int size = 0;  protected int front = 0;  protected int rear = 0;   // Абстрактные методы, которые должны быть реализованы в подклассах  public abstract void enqueue(Object element);  public abstract Object element();  public abstract Object dequeue();   // Реализация остальных методов как общих для всех подклассов  public int size() {  return size;  }   public boolean isEmpty() {  return size == 0;  }   public void clear() {  Arrays.*fill*(elements, null);  size = 0;  front = 0;  rear = 0;  }   // Приватный метод для увеличения емкости массива при необходимости  protected void ensureCapacity() {  if (size == elements.length) {  elements = Arrays.*copyOf*(elements, 2 \* size);  }  } }  // Реализация очереди в виде класса class ArrayQueue extends ArrayQueueADT {  // Переопределение абстрактных методов  public void enqueue(Object element) {  ensureCapacity();  elements[rear] = element;  rear = (rear + 1) % elements.length;  size++;  }   public Object element() {  if (isEmpty()) {  throw new IllegalStateException("Queue is empty");  }  return elements[front];  }   public Object dequeue() {  if (isEmpty()) {  throw new IllegalStateException("Queue is empty");  }  Object removedElement = elements[front];  elements[front] = null;  front = (front + 1) % elements.length;  size--;  return removedElement;  } }  import java.util.Arrays;  public class ArrayQueueModule implements Queue {  private static Object[] *elements* = new Object[10];  private static int *size* = 0;  private static int *front* = 0;  private static int *rear* = 0;   // Проверка инварианта  // Инвариант: size >= 0 && size <= elements.length  // Инвариант: front >= 0 && front < elements.length  // Инвариант: rear >= 0 && rear < elements.length  // Инвариант: size == 0 => front == rear   // Постусловие: size = size' + 1, front = front', rear = (rear' + 1) % elements.length  public void enqueue(Object element) {  ensureCapacity();  *elements*[*rear*] = element;  *rear* = (*rear* + 1) % *elements*.length;  *size*++;  }   // Постусловие: size > 0  public Object element() {  if (isEmpty()) {  throw new IllegalStateException("Queue is empty");  }  return *elements*[*front*];  }   // Постусловие: size = size' - 1, front = (front' + 1) % elements.length  public Object dequeue() {  if (isEmpty()) {  throw new IllegalStateException("Queue is empty");  }  Object removedElement = *elements*[*front*];  *elements*[*front*] = null;  *front* = (*front* + 1) % *elements*.length;  *size*--;  return removedElement;  }   // Постусловие: результат - текущий размер очереди  public int size() {  return *size*;  }   // Постусловие: результат - true, если очередь пуста, иначе - false  public boolean isEmpty() {  return *size* == 0;  }   // Постусловие: size = 0, front = 0, rear = 0  public void clear() {  Arrays.*fill*(*elements*, null);  *size* = 0;  *front* = 0;  *rear* = 0;  }   // Приватный метод для увеличения емкости массива при необходимости  private void ensureCapacity() {  if (*size* == *elements*.length) {  *elements* = Arrays.*copyOf*(*elements*, 2 \* *size*);  }  } }  public class Main {  public static void main(String[] args) {  *testArrayQueueModule*();  *testArrayQueue*();  *testArrayQueueADT*();  }   private static void testArrayQueueModule() {  ArrayQueueModule queueModule = new ArrayQueueModule();   // Тест 1: Добавление элементов в очередь  queueModule.enqueue(1);  queueModule.enqueue(2);  if (queueModule.size() != 2) {  System.*out*.println("Test 1 failed: Incorrect size");  }   // Тест 2: Извлечение элемента из очереди  if ((int) queueModule.dequeue() != 1 || queueModule.size() != 1) {  System.*out*.println("Test 2 failed: Incorrect dequeue or size");  }   // Тест 3: Получение элемента из очереди  if ((int) queueModule.element() != 2) {  System.*out*.println("Test 3 failed: Incorrect element");  }   // Тест 4: Очистка очереди  queueModule.clear();  if (!queueModule.isEmpty()) {  System.*out*.println("Test 4 failed: Queue not cleared");  }  }   private static void testArrayQueue() {  ArrayQueue queue = new ArrayQueue();   // Тест 5: Добавление элементов в очередь  queue.enqueue(3);  queue.enqueue(4);  if (queue.size() != 2) {  System.*out*.println("Test 5 failed: Incorrect size");  }   // Тест 6: Извлечение элемента из очереди  if ((int) queue.dequeue() != 3 || queue.size() != 1) {  System.*out*.println("Test 6 failed: Incorrect dequeue or size");  }   // Тест 7: Получение элемента из очереди  if ((int) queue.element() != 4) {  System.*out*.println("Test 7 failed: Incorrect element");  }   // Тест 8: Очистка очереди  queue.clear();  if (!queue.isEmpty()) {  System.*out*.println("Test 8 failed: Queue not cleared");  }  }   private static void testArrayQueueADT() {  ArrayQueueADT queueADT = new ArrayQueue();   // Тест 9: Добавление элементов в очередь через ADT  queueADT.enqueue(7);  queueADT.enqueue(8);  if (queueADT.size() != 2) {  System.*out*.println("Test 9 failed: Incorrect size");  }   // Тест 10: Извлечение элемента из очереди через ADT  if ((int) queueADT.dequeue() != 7 || queueADT.size() != 1) {  System.*out*.println("Test 10 failed: Incorrect dequeue or size");  }   // Тест 11: Получение элемента из очереди через ADT  if ((int) queueADT.element() != 8) {  System.*out*.println("Test 11 failed: Incorrect element");  }   // Тест 12: Очистка очереди через ADT  queueADT.clear();  if (!queueADT.isEmpty()) {  System.*out*.println("Test 12 failed: Queue not cleared");  }   // Тест 13: Дополнительный тест на пустую очередь через ADT  if (!queueADT.isEmpty() || queueADT.size() != 0) {  System.*out*.println("Test 13 failed: Incorrect empty state");  }   // Попытка извлечения из пустой очереди через ADT  try {  queueADT.dequeue();  System.*out*.println("Test 14 failed: Expected IllegalStateException, but no exception was thrown");  } catch (IllegalStateException e) {  // Ожидаемый результат  }   // Попытка получения элемента из пустой очереди через ADT  try {  queueADT.element();  System.*out*.println("Test 15 failed: Expected IllegalStateException, but no exception was thrown");  } catch (IllegalStateException e) {  // Ожидаемый результат  }  } }  public interface Queue {  void enqueue(Object element);   // Получить первый элемент в очереди  Object element();   // Удалить и вернуть первый элемент в очереди  Object dequeue();   // Получить текущий размер очереди  int size();   // Проверить, является ли очередь пустой  boolean isEmpty();   // Очистить очередь  void clear(); }  public class LinkedQueue<E> implements Queue<E> {  private Node<E> front; // Указатель на начало очереди  private Node<E> rear; // Указатель на конец очереди  private int size; // Текущий размер очереди   // Конструктор  public LinkedQueue() {  this.front = null;  this.rear = null;  this.size = 0;  }   // Реализация методов интерфейса Queue  @Override  public void enqueue(E element) {  Node<E> newNode = new Node<>(element, null);  if (isEmpty()) {  front = newNode;  } else {  rear.setNext(newNode);  }  rear = newNode;  size++;  }   @Override  public E dequeue() {  if (isEmpty()) {  throw new IllegalStateException("Queue is empty");  }  E removedElement = front.getElement();  front = front.getNext();  size--;  if (isEmpty()) {  rear = null;  }  return removedElement;  }   @Override  public E element() {  if (isEmpty()) {  throw new IllegalStateException("Queue is empty");  }  return front.getElement();  }   @Override  public boolean isEmpty() {  return size == 0;  }   @Override  public int size() {  return size;  }   @Override  public void clear() {  front = null;  rear = null;  size = 0;  }   // Вспомогательный класс Node  private static class Node<E> {  private E element;  private Node<E> next;   public Node(E element, Node<E> next) {  this.element = element;  this.next = next;  }   public E getElement() {  return element;  }   public Node<E> getNext() {  return next;  }   public void setNext(Node<E> next) {  this.next = next;  }  } }  public abstract class AbstractQueue<E> implements Queue<E> {  protected int size; // Текущий размер очереди   // Конструктор  public AbstractQueue() {  this.size = 0;  }   // Реализация методов интерфейса Queue  @Override  public boolean isEmpty() {  return size == 0;  }   @Override  public int size() {  return size;  }   @Override  public void clear() {  doClear();  }   // Абстрактные методы, которые должны быть реализованы в подклассах  protected abstract void doEnqueue(E element);   protected abstract E doDequeue();   protected abstract E doElement();   protected abstract void doClear(); }  public interface Queue<E> {  // Добавить элемент в конец очереди  void enqueue(E element);   // Извлечь и удалить элемент из начала очереди  E dequeue();   // Получить элемент из начала очереди без удаления  E element();   // Проверить, является ли очередь пустой  boolean isEmpty();   // Получить текущий размер очереди  int size();   // Очистить очередь (удалить все элементы)  void clear(); }  public class Main {  public static void main(String[] args) {  // Создаем выражение x^2 - 2x + 1  Expression expression = new Subtract(  new Add(  new Multiply(new Variable("x"), new Variable("x")),  new Multiply(new Const(-2), new Variable("x"))  ),  new Const(1)  );   // Вычисляем значение выражения для x = 5  int result = expression.evaluate(5);  System.*out*.println("Result: " + result);   if (args.length != 1) {  System.*out*.println("Usage: java Main <value\_of\_x>");  return;  }   // Получаем значение x из командной строки  int x = Integer.*parseInt*(args[0]);   // Создаем выражение x^2 - 2x + 1  Expression expressio = new Subtract(  new Add(  new Multiply(new Variable("x"), new Variable("x")),  new Multiply(new Const(-2), new Variable("x"))  ),  new Const(1)  );   // Вычисляем значение выражения для заданного x  int result1 = expressio.evaluate(x);  System.*out*.println("Result: " + result1);  } }  public class Add extends BinaryOperation {  public Add(Expression left, Expression right) {  super(left, right);  }   @Override  public int evaluate(int x) {  return left.evaluate(x) + right.evaluate(x);  } }  public abstract class BinaryOperation implements Expression {  protected final Expression left;  protected final Expression right;   public BinaryOperation(Expression left, Expression right) {  this.left = left;  this.right = right;  } }  public class Const implements Expression {  private final int value;   public Const(int value) {  this.value = value;  }   @Override  public int evaluate(int x) {  return value;  } }  public class Devide extends BinaryOperation {  public Devide(Expression left, Expression right) {  super(left, right);  }   @Override  public int evaluate(int x) {  int divisor = right.evaluate(x);  if (divisor == 0) {  throw new ArithmeticException("Division by zero");  }  return left.evaluate(x) / divisor;  } }  public interface Expression {  int evaluate(int x); }  public class Multiply extends BinaryOperation {  public Multiply(Expression left, Expression right) {  super(left, right);  }   @Override  public int evaluate(int x) {  return left.evaluate(x) \* right.evaluate(x);  } }  public class Subtract extends BinaryOperation {  public Subtract(Expression left, Expression right) {  super(left, right);  }   @Override  public int evaluate(int x) {  return left.evaluate(x) - right.evaluate(x);  } }  public class Variable implements Expression {  private final String name;   public Variable(String name) {  this.name = name;  }   @Override  public int evaluate(int x) {  return x;  } } |

Результат:



**Практическая работа № 24. Паттерны проектирования. порождающие паттерны: абстрактная фабрика, фабричный метод**

**Цель:** научиться применять порождающие паттерны при разработке программ на Java. В данной практической работе рекомендуется использовать следующие паттерны: Абстрактная фабрика и фабричный метод.

Код:

|  |
| --- |
| public class ConcreteFactory implements ComplexAbstractFactory {  @Override  public Complex createComplex() {  return new ConcreteComplex(0, 0);  }   @Override  public Complex createComplex(int real, int image) {  return new ConcreteComplex(real, image);  } }  public class ConcreteComplex implements Complex {  private int real;  private int image;   public ConcreteComplex(int real, int image) {  this.real = real;  this.image = image;  }   @Override  public int getReal() {  return real;  }   @Override  public int getImage() {  return image;  } }  public interface ComplexAbstractFactory {  // Создание комплексного числа  Complex createComplex();  // Создание комплексного числа с указанными действительной и мнимой частями  Complex createComplex(int real, int image); }  public interface Complex {  // Методы для получения действительной и мнимой части комплексного числа  int getReal();  int getImage(); }  public class MagicChair implements Chair {  public void doMagic() {  System.*out*.println("Magic is happening!");  } }  public class FunctionalChair implements Chair {  public int sum(int a, int b) {  return a + b;  } }  public class Client {  private Chair chair;   public void sit() {  // Возможно, здесь будет использоваться chair для каких-то действий  System.*out*.println("Sitting on a chair.");  }   public void setChair(Chair chair) {  this.chair = chair;  } }  public interface Chair {  // Возможно, здесь будут добавлены методы общего интерфейса для всех стульев }  public class ChairFactory implements AbstractChairFactory {  @Override  public VictorianChair createVictorianChair() {  return new VictorianChair(0); // Здесь 0 - это пример возраста  }   @Override  public MagicChair createMagicChair() {  return new MagicChair();  }  @Override  public FunctionalChair createFunctionalChair() {  return new FunctionalChair();  } }  public class VictorianChair implements Chair {  private int age;   public VictorianChair(int age) {  this.age = age;  }  public int getAge() {  return age;  } }  public class Main {  public static void main(String[] args) {  System.*out*.println("Hello world!");  } }  public class CreateTextDocument implements ICreateDocument {  @Override  public IDocument createNew() {  System.*out*.println("Creating a new text document.");  return new TextDocument();  }   @Override  public IDocument createOpen() {  System.*out*.println("Opening a text document.");  return new TextDocument();  } }  public class EditorApp {  private IDocument document;  private ICreateDocument documentFactory;   public EditorApp(ICreateDocument documentFactory) {  this.documentFactory = documentFactory;  this.document = null;  }   public void createNewDocument() {  document = documentFactory.createNew();  document.open();  }   public void openDocument() {  document = documentFactory.createOpen();  document.open();  }   public void saveDocument() {  if (document != null) {  document.save();  } else {  System.*out*.println("No document open to save.");  }  }  public static void main(String[] args) {  // Пример использования каркаса для текстового редактора  EditorApp textEditor = new EditorApp(new CreateTextDocument());   textEditor.createNewDocument();  textEditor.saveDocument();   textEditor.openDocument();  textEditor.saveDocument();  } }  public interface ICreateDocument {  IDocument createNew();  IDocument createOpen(); }  public interface IDocument {  void open();  void save(); }  public class TextDocument implements IDocument {  @Override  public void open() {  System.*out*.println("Text document opened.");  }  @Override  public void save() {  System.*out*.println("Text document saved.");  } } |