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 Лещенко В. Р.

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 practice.pr1;
import java.util.Scanner;

public class pr1 {
    public static void main(String[] args)
    {
        task5.main(args);
    }

    public static class task1
    {
        public static void main(String[] args)
        {

            int[] numbers = {1, 2, 3, 4, 5};
            int sum = 0;
            for (int i = 0; i < numbers.length; i++)
            {
                sum += numbers[i];
            }
            double average = (double) sum / numbers.length;
            System.out.println("Сумма элементов массива: " + sum);
            System.out.println("Среднее арифметическое элементов массива: " + average);
        }
    }

    public static class task2
    {
        public static void main(String[] args) {
            Scanner scanner = new Scanner(System.in);
            System.out.println("Введите размер массива: ");
            int size = scanner.nextInt();
            int[] numbers = new int[size];
            System.out.println("Введите элементы массива: ");
            for (int i = 0; i < size; i++) {
                numbers[i] = scanner.nextInt();
            }
            int sum = 0;
            int i = 0;
            do {
                sum += numbers[i];
                i++;
            } while (i < size);
            System.out.println("Сумма элементов массива: " + sum);
            int max = numbers[0];
            int min = numbers[0];
            i = 1;
            while (i < size) {
                if (numbers[i] > max) {
                    max = numbers[i];
                }
                if (numbers[i] < min) {
```

```
                min = numbers[i];
            }
            i++;
        }
        System.out.println("Максимальный элемент в массиве: " + max);
        System.out.println("Минимальный элемент в массиве: " + min);
    }
}

public static class task3
{
    public static void main(String[] args)
    {
        for (int i = 0; i < args.length; i++)
        {
            System.out.println(args[i]);
        }
    }
}

public static class task4
{
    public static void main(String[] args)
    {
        for (int i = 1; i <= 10; i++) {
            System.out.printf("%.2f ", 1.0 / i);
        }
        System.out.println();
    }
}

public static class task5
{
    public static void main(String[] args)
    {
        int n = 5;
        int result = factorial(n);
        System.out.println("Факториал числа " + n + " равен " + result);
    }
}

public static int factorial(int n)
{
    int result = 1;
    for (int i = 1; i <= n; i++)
    {
        result *= i;
    }
    return result;
}
}
```

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:

```
package pr2.pr2_1;

public class Author {
    private String name;
    private String email;
    private char gender;
    public Author(String name, String email, char gender) // Конструктор
    {
        this.name = name;
        this.email = email;
        this.gender = gender;
    }

    // Геттеры и сеттеры для полей name, email и gender

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public char getGender() {
        return gender;
    }

    public void setGender(char gender) {
        this.gender = gender;
    }

    public String toString() {
        return "Author [name=" + name + ", email=" + email + ", gender=" + gender + "]";
    }
```

```
}
```

```
package pr2.pr2_1;

public class SubAuthor extends Author {
    public SubAuthor(String name, String email, char gender) {
        super(name, email, gender);
    }
    // конструктор super - конструктор старшего класса
}
```

```
import pr2.pr2_1.Author;

public class Main {
    public static void main(String[] args)
    {
        Author author = new Author("Вячеслав", "viacheslav@mail.ru", 'M');
        System.out.printf("Hello and welcome!\n");
        String name = author.getName();
        System.out.printf(name);
    }
}
```

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:

```
package pr3.pr3_1;
import java.util.*;

public class task1 {
    public static void main(String[] args) {
        int n = 10;
        int[] nums = new int[n];
        Random random = new Random();
        for (int i = 0; i < n; i++){
            nums[i] = random.nextInt(100);
        }
        System.out.print("Массив сгенерированный через класс Random\n> ");
        for (int i = 0; i < n; i++){
            System.out.print(nums[i] + ",");
        }

        Arrays.sort(nums);
        System.out.print("\n\nМассив отсортирован\n> ");
        for (int i = 0; i < n; i++){
            System.out.print(nums[i] + ",");
        }

        for (int i = 0; i < n; i++){
            nums[i] = (int) (Math.random()*100);
        }

        System.out.print("\n\nМассив сгенерированный через метод .random()\n> ");
        for (int i = 0; i < n; i++){
            System.out.print(nums[i] + ",");
        }
    }
}
```

```
package pr3.pr3_6;

public class WrapperMethodsDemo {
    public static void main(String[] args) {
        // Примеры использования методов классов-оболочек

        // Boolean
        Boolean boolObj = true;
        boolean boolPrimitive = boolObj.booleanValue(); // Преобразование в примитивный тип
```

```
    // Byte
    Byte byteObj = 123;
    byte bytePrimitive = byteObj.byteValue(); // Преобразование в примитивный тип

    // Character
    Character charObj = 'A';
    char charPrimitive = charObj.charValue(); // Преобразование в примитивный тип

    // Double
    Double doubleObj = 3.14;
    double doublePrimitive = doubleObj.doubleValue(); // Преобразование в примитивный тип

    // Float
    Float floatObj = 2.718f;
    float floatPrimitive = floatObj.floatValue(); // Преобразование в примитивный тип

    // Integer
    Integer intObj = 42;
    int intPrimitive = intObj.intValue(); // Преобразование в примитивный тип

    // Long
    Long longObj = 123456789L;
    long longPrimitive = longObj.longValue(); // Преобразование в примитивный тип

    // Short
    Short shortObj = 456;
    short shortPrimitive = shortObj.shortValue(); // Преобразование в примитивный тип

    // Методы для парсинга строк и преобразования чисел в строки
    int parsedInt = Integer.parseInt("123");
    double parsedDouble = Double.parseDouble("3.14");
    String intToString = Integer.toString(42);
    String doubleToString = Double.toString(3.14);

    // Выводим результаты
    System.out.println("Boolean: " + boolPrimitive);
    System.out.println("Byte: " + bytePrimitive);
    System.out.println("Character: " + charPrimitive);
    System.out.println("Double: " + doublePrimitive);
    System.out.println("Float: " + floatPrimitive);
    System.out.println("Integer: " + intPrimitive);
    System.out.println("Long: " + longPrimitive);
    System.out.println("Short: " + shortPrimitive);
    System.out.println("Parsed Int: " + parsedInt);
    System.out.println("Parsed Double: " + parsedDouble);
    System.out.println("Int to String: " + intToString);
    System.out.println("Double to String: " + doubleToString);
  }
}
```

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:

```
package pr4.pr4_1;

public enum Season {
    SPRING("Весна", "Прохладное время года", 15),
    SUMMER("Лето", "Теплое время года", 25),
    AUTUMN("Осень", "Прохладное время года", 18),
    WINTER("Зима", "Холодное время года", 0);

    private String name;
    private String description;
    private int averageTemperature;

    Season(String name, String description, int averageTemperature) {
        this.name = name;
        this.description = description;
        this.averageTemperature = averageTemperature;
    }

    public String getName() {
        return name;
    }

    public String getDescription() {
        if (this == SUMMER) {
            return "Теплое время года";
        }
        return description;
    }

    public int getAverageTemperature() {
        return averageTemperature;
    }

    public static void printSeasonInfo(Season season) {
        System.out.println("Информация о времени года:");
        System.out.println("Название: " + season.getName());
        System.out.println("Средняя температура: " + season.getAverageTemperature() + "°C");
        System.out.println("Описание: " + season.getDescription());
    }
}

package pr4.pr4_1;

public class Main {
    public static void main(String[] args) {
        Season myFavoriteSeason = Season.SUMMER;
        System.out.println("Мое любимое время года:");
        Season.printSeasonInfo(myFavoriteSeason);
```

```
        System.out.println("\nОписание времен года:");
        for (Season season : Season.values()) {
            System.out.println("Название: " + season.getName());
            System.out.println("Средняя температура: " + season.getAverageTemperature() + "°C");
            System.out.println("Описание: " + season.getDescription() + "\n");
        }
    }
}
```

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

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

Код:

```
package pr5.pr5_1;
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 Label winnerLabel;

    public FootballMatchSimulator() {
        setTitle("Football Match Simulator");
        setSize(400, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new GridLayout(4, 1));

        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 Label("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(milanButton);
        add(madridButton);
        add(resultLabel);
        add(lastScorerLabel);
        add(winnerLabel);
    }

    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 (milanScore < madridScore) {
            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().setVisible(true);
            }
        });
    }
}
```

Практическая работа № 6. Интерфейсы в Java

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

Код:

```
package pr6.pr6_1and2;

// Интерфейс Movable
public interface Movable {
    public void moveUp();
    public void moveDown();
    public void moveLeft();
    public void moveRight();
}

package pr6.pr6_1and2;

// Класс MovableCircle, представляющий движущийся круг
class MovableCircle implements Movable {
    private MovablePoint center;
    private int radius;

    public MovableCircle(int x, int y, int xSpeed, int ySpeed, int radius) {
        center = new MovablePoint(x, y, xSpeed, ySpeed);
        this.radius = radius;
    }

    @Override
    public void moveUp() {
        center.moveUp();
    }

    @Override
    public void moveDown() {
        center.moveDown();
    }

    @Override
    public void moveLeft() {
        center.moveLeft();
    }

    @Override
    public void moveRight() {
        center.moveRight();
    }

    @Override
    public String toString() {
        return "Center: " + center + ", Radius: " + radius;
    }
}

package pr6.pr6_1and2;

// Класс MovablePoint, представляющий точку с координатами и скоростью
class MovablePoint implements Movable {
```

```
    int x, y, xSpeed, ySpeed;

    public MovablePoint(int x, int y, int xSpeed, int ySpeed) {
        this.x = x;
        this.y = y;
        this.xSpeed = xSpeed;
        this.ySpeed = ySpeed;
    }

    @Override
    public void moveUp() {
        y -= ySpeed;
    }

    @Override
    public void moveDown() {
        y += ySpeed;
    }

    @Override
    public void moveLeft() {
        x -= xSpeed;
    }

    @Override
    public void moveRight() {
        x += xSpeed;
    }

    @Override
    public String toString() {
        return "(" + x + ", " + y + ")";
    }
}

package pr6.pr6_1and2;

class MovableRectangle implements Movable {
    private MovablePoint topLeft;
    private MovablePoint bottomRight;

    public MovableRectangle(
            int x1, int y1, int x2, int y2, int xSpeed, int ySpeed) {
        topLeft = new MovablePoint(x1, y1, xSpeed, ySpeed);
        bottomRight = new MovablePoint(x2, y2, xSpeed, ySpeed);

        if (!hasSameSpeed(topLeft, bottomRight)) {
            throw new IllegalArgumentException("Both points must have the same speed.");
        }
    }

    private boolean hasSameSpeed(MovablePoint p1, MovablePoint p2) {
        return p1.xSpeed == p2.xSpeed && p1.ySpeed == p2.ySpeed;
    }

    @Override
    public void moveUp() {
        topLeft.moveUp();
        bottomRight.moveUp();
    }
```

```
    @Override
    public void moveDown() {
        topLeft.moveDown();
        bottomRight.moveDown();
    }

    @Override
    public void moveLeft() {
        topLeft.moveLeft();
        bottomRight.moveLeft();
    }

    @Override
    public void moveRight() {
        topLeft.moveRight();
        bottomRight.moveRight();
    }

    @Override
    public String toString() {
        return "Top Left: " + topLeft + ", Bottom Right: " + bottomRight;
    }
}
```

```
package pr6.pr6_1and2;

public class Main {
    public static void main(String[] args) {
        MovablePoint point1 = new MovablePoint(1, 2, 1, 1);
        MovablePoint point2 = new MovablePoint(4, 5, 1, 1);

        MovableRectangle rectangle = new MovableRectangle(1, 2, 4, 5, 1, 1);

        System.out.println("Initial Point 1: " + point1);
        System.out.println("Initial Point 2: " + point2);
        System.out.println("Initial Rectangle: " + rectangle);

        point1.moveUp();
        point2.moveDown();
        rectangle.moveLeft();

        System.out.println("Point 1 after moving up: " + point1);
        System.out.println("Point 2 after moving down: " + point2);
        System.out.println("Rectangle after moving left: " + rectangle);
    }
}
```

Практическая работа № 7. Реализация интерфейсов

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

Код:

```
package pr7.pr7_4;

public interface MathCalculable {
    double pow(double a, double b);
    double ModuleComp(double a,double b);
    static double PI = 3.1415926535;
}

package pr7.pr7_4;

public class MathFunc implements MathCalculable {
    public double pow(double a, double b) {
        return Math.pow(a,b);
    }
    public double ModuleComp(double a, double b) {
        double apow = Math.pow(a,2);
        double bpow = Math.pow(b,2);
        return Math.sqrt((apow+bpow));
    }
}

package pr7.pr7_4;
import java.util.Scanner;

public class Test_Math {
    public static void main(String[] args) {
        MathFunc math = new MathFunc();
        System.out.print("Circle radius: ");
        Scanner s = new Scanner(System.in);
        double r = s.nextDouble();
        System.out.println("Circle area: " + math.PI * r * r + '\n');

        System.out.print("Enter numbers of a complex number: ");
        double a = s.nextDouble();
        double b = s.nextDouble();
        System.out.println("Module: " + math.ModuleComp(a, b) + '\n');

        System.out.print("Number and its pow: ");
        a = s.nextDouble();
        b = s.nextDouble();
        System.out.println(math.pow(a, b));
    }
}
```

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

Цель: разработка и программирование рекурсивных алгоритмов на языке Java.

Код:

```
package pr8.pr8_1;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Введите количество цифр (k): ");
        int k = scanner.nextInt();

        System.out.print("Введите сумму цифр (s): ");
        int s = scanner.nextInt();

        int count = countNumbersWithSum(k, s);
        System.out.println("Количество чисел: " + count);
    }

    // Функция для подсчета количества k-значных чисел с суммой цифр s
    public static int countNumbersWithSum(int k, int s) {
        if (s < 1 || s > 9 * k) {
            // Если сумма невозможна для k-значных чисел, возвращаем 0
            return 0;
        }

        // Используем рекурсивную функцию для подсчета
        return countNumbersWithSumHelper(k, s, 1, "");
    }

    // Вспомогательная рекурсивная функция для подсчета
    private static int countNumbersWithSumHelper(int k, int s, int startDigit, String currentNumber) {
        if (k == 0) {
            // Если осталось нулевое количество цифр, проверяем сумму
            if (s == 0) {
                // Если сумма совпадает, то число подходит
                System.out.println(currentNumber); // Вывод числа для отладки
                return 1;
            } else {
                return 0;
            }
        }

        int count = 0;

        for (int digit = startDigit; digit <= 9; digit++) {
            if (s - digit >= 0) {
                // Рекурсивно вызываем функцию для оставшихся цифр
                count += countNumbersWithSumHelper(k - 1, s - digit, 0, currentNumber + digit);
            }
```

```
        }

        return count;
    }
}
```

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

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

Код:

```
package pr9.pr9_1;

import java.util.Arrays;

class Student implements Comparable<Student> {
    private int iDNumber;
    private String name;

    public Student(int iDNumber, String name) {
        this.iDNumber = iDNumber;
        this.name = name;
    }

    public int getIDNumber() {
        return iDNumber;
    }

    @Override
    public int compareTo(Student otherStudent) {
        // Сравниваем объекты по полю iDNumber
        return Integer.compare(this.iDNumber, otherStudent.iDNumber);
    }

    @Override
    public String toString() {
        return "Student{" +
                "iDNumber=" + iDNumber +
                ", name='" + name + '\'' +
                '}';
    }
}
package pr9.pr9_1;



public class StudentSorter {
    public static void main(String[] args) {
        Student[] students = {
                new Student(102, "Alice"),
                new Student(101, "Bob"),
                new Student(105, "Charlie"),
                new Student(103, "David"),
                new Student(104, "Eve")
        };

        System.out.println("Before sorting:");
        for (Student student : students) {
            System.out.println(student);
```

```
        }

        // Сортировка вставками с использованием compareTo()
        insertionSort(students);

        System.out.println("\nAfter sorting by iDNumber:");
        for (Student student : students) {
            System.out.println(student);
        }
    }

    public static void insertionSort(Comparable[] arr) {
        int n = arr.length;
        for (int i = 1; i < n; i++) {
            Comparable key = arr[i];
            int j = i - 1;

            while (j >= 0 && arr[j].compareTo(key) > 0) {
                arr[j + 1] = arr[j];
                j--;
            }
            arr[j + 1] = key;
        }
    }
}
```

Практическая работа № 10. Стандартные интерфейсы Джава. Интерфейс Comparator

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

Код:

```
package pr10.pr10_1;
// Задание 1: Создаем класс Student
class Student {
    private String firstName;
    private String lastName;
    private String specialty;
    private int course;
    private String group;
    private double gpa;

    public Student(String firstName, String lastName, String specialty, int course, String group, double gpa) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.specialty = specialty;
        this.course = course;
        this.group = group;
        this.gpa = gpa;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getSpecialty() {
        return specialty;
    }

    public void setSpecialty(String specialty) {
        this.specialty = specialty;
    }

    public int getCourse() {
        return course;
    }
```

```
    public void setCourse(int course) {
        this.course = course;
    }

    public String getGroup() {
        return group;
    }

    public void setGroup(String group) {
        this.group = group;
    }

    public double getGpa() {
        return gpa;
    }

    public void setGpa(double gpa) {
        this.gpa = gpa;
    }

    @Override
    public String toString() {
        return "Student{" +
                "firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", specialty='" + specialty + '\'' +
                ", course=" + course +
                ", group='" + group + '\'' +
                ", gpa=" + gpa +
                '}';
    }
}

package pr10.pr10_1;

import java.util.*;

// Задание 2: Создаем класс SortingStudentsByGPA
class SortingStudentsByGPA {
    private List<Student> iDNumber;

    public SortingStudentsByGPA() {
        iDNumber = new ArrayList<>();
    }

    // Метод для заполнения массива студентов
    public void setArray(List<Student> students) {
        iDNumber.addAll(students);
    }

    // Метод для сортировки по среднему баллу (GPA)
    public void sortByGPA() {
        iDNumber.sort(Comparator.comparingDouble(Student::getGpa).reversed());
    }

    // Метод для вывода массива студентов
    public void outArray() {
        for (Student student : iDNumber) {
```

```
            System.out.println(student);
        }
    }

    public List<Student> getIDNumber() {
        return iDNumber;
    }
}

package pr10.pr10_1;
import java.util.*;

public class Main {
    public static void main(String[] args) {
        // Создаем два списка данных о студентах
        List<Student> students1 = new ArrayList<>();
        students1.add(new Student("Иван", "Петров", "Информатика", 3, "Группа А", 4.5));
        students1.add(new Student("Мария", "Иванова", "Физика", 2, "Группа В", 4.1));

        List<Student> students2 = new ArrayList<>();
        students2.add(new Student("Алексей", "Сидоров", "Математика", 4, "Группа А", 3.8));
        students2.add(new Student("Елена", "Козлова", "Химия", 3, "Группа С", 3.1));

        // Задание 2: Создаем объект SortingStudentsByGPA и заполняем его массив студентов
        SortingStudentsByGPA sorter = new SortingStudentsByGPA();
        sorter.setArray(students1);
        sorter.setArray(students2);

        // Задание 2: Сортировка по среднему баллу (GPA)
        sorter.sortByGPA();

        // Задание 2: Вывод отсортированного списка
        System.out.println("Сортировка по GPA:");
        sorter.outArray();
    }
}
```

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:

```
package pr11;

import java.util.Date;

public class task1 {
   public static void main(String[] args)
   {
      int year = 2023;
      int month = 03;
      int day = 3;
      int hour = 20;
      int minute = 15;
      System.out.printf("Фамилия: Борзов Год: %s Месяц: %s День: %s Час: %s Минут: %s\
n",year,month,day,hour,minute);
      Date date = new Date();
      System.out.println("Дата сдачи: " + date);
   }
}
```

Задание 2:

```
package pr11;

import java.util.Scanner;
import java.util.Calendar;

public class Task2 {
   public static void main(String[] args)
   {
      long system = System.currentTimeMillis();
      Calendar calendar = Calendar.getInstance();
      Calendar mytime = Calendar.getInstance();
      Scanner scanner = new Scanner(System.in);

      int year1 = calendar.get(Calendar.YEAR);
      int month1 = calendar.get(Calendar.MONTH);
      int day1 = calendar.get(Calendar.DAY_OF_MONTH);
      int hour1 = calendar.get(Calendar.HOUR_OF_DAY);
      int minute1 = calendar.get(Calendar.MINUTE);
      int second1 = calendar.get(Calendar.SECOND);

      System.out.println("Введите год:");
      int year2 = scanner.nextInt();
      mytime.set(Calendar.YEAR,year2);
      System.out.println("Введите месяц:");
      int month2 = scanner.nextInt();
      mytime.set(Calendar.MONTH,month2);
```

```
        System.out.println("Введите день:");
        int day2 = scanner.nextInt();
        mytime.set(Calendar.DAY_OF_MONTH,day2);
        System.out.println("Введите час:");
        int hour2 = scanner.nextInt();
        mytime.set(Calendar.HOUR_OF_DAY,hour2);
        System.out.println("Введите минуту:");
        int minute2 = scanner.nextInt();
        mytime.set(Calendar.MINUTE,minute2);
        System.out.println("Введите секунду:");
        int second2 = scanner.nextInt();
        mytime.set(Calendar.SECOND,second2);


        if(calendar.getTimeInMillis() != mytime.getTimeInMillis() ){
            System.out.println("Ваше время расходится");
        }
        else
        {
            System.out.println("Ваше время сходится");
        }
    }
}
```

Задание 3:

```
package pr11;
import java.util.Calendar;
import java.util.Scanner;


public class Task3 extends Student {
    public static void main(String[] args) {
        Student student = new Student();
        Scanner scanner = new Scanner(System.in);
        Calendar calendar = Calendar.getInstance();
        System.out.println("0 - обычный формат, 1 - расширенный формат");
        int x = scanner.nextInt();
        int year = calendar.get(Calendar.YEAR);
        int month = calendar.get(Calendar.MONTH) + 1;
        int day = calendar.get(Calendar.DAY_OF_MONTH);
        int hour = calendar.get(Calendar.HOUR_OF_DAY);
        int minute = calendar.get(Calendar.MINUTE);
        int second = calendar.get(Calendar.SECOND);
        String A = "";
        switch (x)
        {
            case 1:
            {
                A+=year+"/"+month+"/"+day+"\n";
            }
            case 0:
            {
                A+=hour+":"+minute+":"+second+"\n";
            }
            default:{}
        }
        System.out.println(A);
    }


}
```

Задание 4:

```
package pr11;


import java.util.Calendar;
import java.util.Scanner;
import java.util.Date;


public class Task4 {
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        Calendar calendar = Calendar.getInstance();
        int year = scanner.nextInt();
        int month = scanner.nextInt();
        int day = scanner.nextInt();
        int hour = scanner.nextInt();
        int minute = scanner.nextInt();
        Date data = new Date(year-1900,month-1,day,hour,minute);
        System.out.println(data);
        System.out.printf("%d %d %d %d %d%n",calendar.get(calendar.YEAR),calendar.get(calendar.MONTH-
1),calendar.get(calendar.DAY_OF_MONTH),calendar.get(calendar.HOUR_OF_DAY),calendar.get(calendar.MIN
UTE));
    }
}
```

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

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;
    }
}
```

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:

```
package pr13;

public class Task2 {
    public static void main(String[] args) {
        String str = "Пример строки";

        // Получаем длину строки
        int length = str.length();

        // Проверяем, что строка не пуста
        if (length > 0) {
            // Индекс последнего символа равен длине строки минус 1
            char lastChar = str.charAt(length - 1);

            // Выводим последний символ на экран
            System.out.println("Последний символ строки: " + lastChar);
        } else {
            // Если строка пуста, выводим сообщение об этом
            System.out.println("Строка пуста, нельзя получить последний символ.");
        }
    }
}
```

```
package pr13;

public class Task5 {
    public static void main(String[] args) {
        String str = "Пример строки, содержащей слово Java";

        // Проверяем, содержит ли строка подстроку "Java"
        boolean containsJava = str.contains("Java");

        if (containsJava) {
            System.out.println("Строка содержит подстроку 'Java'.");
        } else {
            System.out.println("Строка не содержит подстроку 'Java'.");
        }
    }
}
```

```
package pr13;

public class Task6 {
    public static void main(String[] args) {
        String str = "I like Java!!!";

        // Ищем позицию подстроки "Java" в строке
        int position = str.indexOf("Java");
```

```
        if (position != -1) {
            System.out.println("Подстрока 'Java' найдена в позиции: " + position);
        } else {
            System.out.println("Подстрока 'Java' не найдена в строке.");
        }
    }
}
```

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:

```
package pr14;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Task3 {
    public static void main(String[] args) {
        String text = "Пример текста со списками цен: 25.98 USD, 1000 RUB, 5.99 EUR, 44 ERR, 0.004 EU.";

        // Регулярные выражения для поиска цен в разных валютах
        String usdPattern = "\\b(\\d+\\.\\d+) USD\\b";
        String rubPattern = "\\b(\\d+) RUB\\b";
        String eurPattern = "\\b(\\d+\\.\\d+) EUR\\b";

        Pattern usdRegex = Pattern.compile(usdPattern);
        Pattern rubRegex = Pattern.compile(rubPattern);
        Pattern eurRegex = Pattern.compile(eurPattern);

        Matcher usdMatcher = usdRegex.matcher(text);
        Matcher rubMatcher = rubRegex.matcher(text);
        Matcher eurMatcher = eurRegex.matcher(text);

        while (usdMatcher.find()) {
            System.out.println("USD Price: " + usdMatcher.group(1));
        }

        while (rubMatcher.find()) {
            System.out.println("RUB Price: " + rubMatcher.group(1));
        }

        while (eurMatcher.find()) {
            System.out.println("EUR Price: " + eurMatcher.group(1));
        }
    }
}
```

```
package pr14;

import java.util.ArrayList;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Task4 {
    public static void main(String[] args) {
        System.out.println("Enter your string: ");
```

```
        Scanner inp = new Scanner(System.in);
        String string = inp.nextLine();
        Pattern pattern = Pattern.compile("\\w?\\s*\\d+\\s*\\+\\s*\\d+\\s*\\w?");
        Matcher matcher = pattern.matcher(string);
        boolean bool = false;
        if(matcher.find())
        {
            bool = true;
        }
        if(bool == true)
        {
            System.out.println("There is + in your expression");
        }
        else {
            System.out.println("There is no + in your expression");
        }
    }
}
```

```
package pr14;

import java.util.Scanner;
import java.util.StringTokenizer;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Task5 {
    private static boolean dateformat(String string)
    {
        boolean res = false;
        StringTokenizer token = new StringTokenizer(string,"/");
        int[] date = new int[3];

        for(int i = 0;i<3;i++)
        {
            date[i] = Integer.parseInt(token.nextToken());
        }
        if(date[1] == 2)
        {
            if(date[0]<=29 && date[2]%4==0){res = true;}
            else if(date[0]<29)
            {
                res = true;
            }
            else {
                res = false;
            }
        }
        else
            res = true;

        return res;
    }
```

```
   public static void main(String[] args) {
      System.out.println("Enter your string: ");
      Scanner inp = new Scanner(System.in);
      String string = inp.nextLine();
      Pattern pattern = Pattern.compile("(((0[1-9]|[1-2][0-9]|(3[0-1]))/((0[1-9])|(1[0-2]))/((19)|([2-9][0-9]))[0-9][0-
9]\\D*)+)");

      Matcher matcher = pattern.matcher(string);
      boolean bool = matcher.matches();
      if(bool == true && dateformat(string) == true)
      {
         System.out.println("Your expression is right: ");
         System.out.println(string);
      }
      else {
         System.out.println("Your expression isn't right");
      }
   }
}
```

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:

```
package pr15.pr15_1;

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

public class CalculatorGUI extends JFrame {
    private JTextField numField1;
    private JTextField numField2;
    private JTextField resultField;
    private JButton addButton;
    private JButton subtractButton;
    private JButton multiplyButton;
    private JButton divideButton;

    public CalculatorGUI() {
        setTitle("Simple Calculator");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new GridLayout(4, 2));

        numField1 = new JTextField(10);
        numField2 = new JTextField(10);
        resultField = new JTextField(10);
        resultField.setEditable(false);

        addButton = new JButton("Add");
        subtractButton = new JButton("Subtract");
        multiplyButton = new JButton("Multiply");
        divideButton = new JButton("Divide");

        addButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                performOperation("+");
            }
        });

        subtractButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                performOperation("-");
            }
        });

        multiplyButton.addActionListener(new ActionListener() {
```

```
        @Override
        public void actionPerformed(ActionEvent e) {
            performOperation("*");
        }
    });

    divideButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            performOperation("/");
        }
    });

    add(new JLabel("Number 1:"));
    add(numField1);
    add(new JLabel("Number 2:"));
    add(numField2);
    add(new JLabel("Result:"));
    add(resultField);
    add(addButton);
    add(subtractButton);
    add(multiplyButton);
    add(divideButton);

    pack();
    setVisible(true);
}

private void performOperation(String operator) {
    try {
        double num1 = Double.parseDouble(numField1.getText());
        double num2 = Double.parseDouble(numField2.getText());
        double result = 0.0;

        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 {
                    resultField.setText("Division by zero");
                    return;
                }
                break;
        }

        resultField.setText(String.valueOf(result));
    } catch (NumberFormatException e) {
        resultField.setText("Invalid input");
    }
}
public static void main(String[] args) {
```

```
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new CalculatorGUI();
            }
        });
    }
}
```

```
package pr15.pr15_2;

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

public class CountrySelectionApp {
    private JComboBox<String> countryComboBox;

    public CountrySelectionApp() {
        JFrame frame = new JFrame("Выбор страны");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new FlowLayout());

        String[] countries = {"Выберите страну", "Россия", "США", "Канада", "Великобритания", "Германия",
"Франция", "Япония"};

        countryComboBox = new JComboBox<>(countries);
        JButton selectButton = new JButton("Выбрать");

        selectButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String selectedCountry = (String) countryComboBox.getSelectedItem();
                if (selectedCountry.equals("Выберите страну")) {
                    JOptionPane.showMessageDialog(frame, "Пожалуйста, выберите действительную страну.",
"Ошибка", JOptionPane.ERROR_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(frame, "Вы выбрали: " + selectedCountry, "Выбранная страна",
JOptionPane.INFORMATION_MESSAGE);
                }
            }
        });

        frame.add(countryComboBox);
        frame.add(selectButton);

        frame.setSize(250, 100);
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
```

```
            new CountrySelectionApp();
        }
    });
  }
}
```

```
package pr15.pr15_3;

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

public class MenuDemo {
    private JFrame frame;
    private JTextArea textArea;

    public MenuDemo() {
        frame = new JFrame("Программа с меню");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);

        JMenuBar menuBar = new JMenuBar();

        JMenu fileMenu = new JMenu("Файл");
        JMenu editMenu = new JMenu("Правка");
        JMenu helpMenu = new JMenu("Справка");

        JMenuItem saveItem = new JMenuItem("Сохранить");
        JMenuItem exitItem = new JMenuItem("Выйти");
        JMenuItem copyItem = new JMenuItem("Копировать");
        JMenuItem cutItem = new JMenuItem("Вырезать");
        JMenuItem pasteItem = new JMenuItem("Вставить");
        JMenuItem aboutItem = new JMenuItem("О программе");

        saveItem.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Добавьте код для сохранения файла
                JOptionPane.showMessageDialog(frame, "Файл сохранен.");
            }
        });

        exitItem.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });

        copyItem.addActionListener(new ActionListener() {
            @Override
```

```
        public void actionPerformed(ActionEvent e) {
            textArea.copy();
        }
    });

    cutItem.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            textArea.cut();
        }
    });

    pasteItem.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            textArea.paste();
        }
    });

    aboutItem.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            JOptionPane.showMessageDialog(frame, "Пример программы с меню.");
        }
    });

    fileMenu.add(saveItem);
    fileMenu.add(exitItem);
    editMenu.add(copyItem);
    editMenu.add(cutItem);
    editMenu.add(pasteItem);
    helpMenu.add(aboutItem);

    menuBar.add(fileMenu);
    menuBar.add(editMenu);
    menuBar.add(helpMenu);

    frame.setJMenuBar(menuBar);

    textArea = new JTextArea();
    frame.add(new JScrollPane(textArea));

    frame.setVisible(true);
  }
  public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new MenuDemo();
        }
    });
  }
}
```

```
package pr15.pr15_4;

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

public class CalculatorApp {
    private JFrame frame;
    private JTextField display;
    private double currentValue;
    private String currentOperator;

    public CalculatorApp() {
        frame = new JFrame("Калькулятор");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new BorderLayout());

        display = new JTextField(20);
        display.setFont(new Font("Arial", Font.PLAIN, 20));
        display.setEditable(false);
        frame.add(display, BorderLayout.NORTH);

        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new GridLayout(4, 4));

        String[] buttonLabels = {
                "1", "2", "3", "/",
                "4", "5", "6", "*",
                "7", "8", "9", "-",
                "0", ".", "=", "+"
        };

        for (String label : buttonLabels) {
            JButton button = new JButton(label);
            button.setFont(new Font("Arial", Font.PLAIN, 20));
            button.addActionListener(new ButtonClickListener());
            buttonPanel.add(button);
        }

        frame.add(buttonPanel, BorderLayout.CENTER);

        frame.pack();
        frame.setVisible(true);
    }

    private class ButtonClickListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            String command = e.getActionCommand();

            if (Character.isDigit(command.charAt(0)) || command.equals(".")) {
```

```
            display.setText(display.getText() + command);
        } else if (command.equals("=")) {
            performCalculation();
        } else {
            if (currentOperator != null) {
                performCalculation();
            }
            currentValue = Double.parseDouble(display.getText());
            currentOperator = command;
            display.setText("");
        }
    }

    private void performCalculation() {
        if (currentOperator != null) {
            double newValue = Double.parseDouble(display.getText());
            switch (currentOperator) {
                case "+":
                    currentValue += newValue;
                    break;
                case "-":
                    currentValue -= newValue;
                    break;
                case "*":
                    currentValue *= newValue;
                    break;
                case "/":
                    if (newValue != 0) {
                        currentValue /= newValue;
                    } else {
                        display.setText("Error");
                        currentValue = 0;
                        currentOperator = null;
                        return;
                    }
                    break;
            }
            display.setText(Double.toString(currentValue));
            currentOperator = null;
        }
    }
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new CalculatorApp();
        }
    });
}
}
```

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:

```
package pr16.pr16_1;
import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class BorderLayoutDemo {
    private JFrame frame;

    public BorderLayoutDemo() {
        frame = new JFrame("Добро пожаловать в Москву!");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new BorderLayout());

        JPanel centerPanel = new JPanel();
        centerPanel.setBackground(Color.WHITE);
        centerPanel.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {
                JOptionPane.showMessageDialog(frame, "Добро пожаловать в ЦАО");
            }
        });
        frame.add(centerPanel, BorderLayout.CENTER);

        JPanel westPanel = new JPanel();
        westPanel.setBackground(Color.LIGHT_GRAY);
        westPanel.setPreferredSize(new Dimension(100, 400)); // Установка размеров для западной панели
        westPanel.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {
                JOptionPane.showMessageDialog(frame, "Добро пожаловать в ЗАО");
            }
        });
        frame.add(westPanel, BorderLayout.WEST);

        JPanel southPanel = new JPanel();
        southPanel.setBackground(Color.CYAN);
        southPanel.setPreferredSize(new Dimension(400, 100)); // Установка размеров для южной панели
        southPanel.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {
                JOptionPane.showMessageDialog(frame, "Добро пожаловать в ЮАО");
            }
        });
        frame.add(southPanel, BorderLayout.SOUTH);

        JPanel northPanel = new JPanel();
        northPanel.setBackground(Color.ORANGE);
        northPanel.setPreferredSize(new Dimension(400, 100)); // Установка размеров для северной панели
```

```
        northPanel.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {
                JOptionPane.showMessageDialog(frame, "Добро пожаловать в САО");
            }
        });
        frame.add(northPanel, BorderLayout.NORTH);

        JPanel eastPanel = new JPanel();
        eastPanel.setBackground(Color.PINK);
        eastPanel.setPreferredSize(new Dimension(100, 400)); // Установка размеров для восточной панели
        eastPanel.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {
                JOptionPane.showMessageDialog(frame, "Добро пожаловать в ВАО");
            }
        });
        frame.add(eastPanel, BorderLayout.EAST);

        frame.setSize(600, 600); // Установка размера окна
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new BorderLayoutDemo();
            }
        });
    }
}
```

```
package pr16.pr16_1;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.lang.Math;

class GuessingGame extends JFrame {
    JButton tryButton = new JButton("Отправить");
    JTextField numberField = new JTextField(2);
    JLabel stateText = new JLabel("Угадайте число (от 0 до 20)");
    int guessNumber = (int) (Math.random()*20);
    int cnt =0;

    public GuessingGame(){
        super("Игра");
        setDefaultCloseOperation( EXIT_ON_CLOSE);
        setSize(300, 300);

        JPanel grid = new JPanel(new GridLayout(3,1,5,0));
        grid.add(stateText);
        grid.add(numberField);
        grid.add(tryButton);
```

```
    JPanel flowButton = new JPanel(new FlowLayout(FlowLayout.CENTER));
    flowButton.add(grid);

    Container window = getContentPane();
    window.add(flowButton);

    tryButton.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        int userNumber = Integer.parseInt(numberField.getText());
        if (userNumber==guessNumber){
          stateText.setText("Вы выиграли");
          numberField.setVisible(false);
          tryButton.setVisible(false);
        }
        else if (userNumber!=guessNumber){
          if (userNumber<guessNumber){
            stateText.setText("Число больше");
          }
          else {
            stateText.setText("Число меньше");
          }
        }
        cnt++;
        if (cnt==3 && userNumber!= guessNumber){
          stateText.setText("Вы проиграли");
          System.out.println(guessNumber);
          numberField.setVisible(false);
          tryButton.setVisible(false);
        }
      }
    });
  }

  public static void main(String[] args) {
    new GuessingGame().setVisible(true);
  }
}
```

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

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

Код:

```
package pr17.pr17_1;

// Создание класса модели.
public 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;
   }
}
```

```
package pr17.pr17_1;

// Создание класса контроллера.
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 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());
    }
}
```

```
package pr17.pr17_1;

// Создание класса представления.
public class StudentView {
    public void printStudentDetails(String studentName, String studentRollNo){
        System.out.println("Студент: ");
        System.out.println("Имя: " + studentName);
        System.out.println("Номер студенческого билета: " + studentRollNo);
    }
}
```

```
package pr17.pr17_1;

public class MVCPatternDemo {
    public static void main(String[] args) {
        // Извлечение студента из базы данных
        Student student = retrieveStudentFromDatabase();

        // Создание модели, представления и контроллера
        Student model = new Student();
        model.setName(student.getName());
        model.setRollNo(student.getRollNo());

        StudentView view = new StudentView();

        StudentController controller = new StudentController(model, view);

        // Обновляем представление
        controller.updateView();
    }

    public static Student retrieveStudentFromDatabase() {
        // Моделирование извлечения студента из базы данных
        Student student = new Student();
        student.setName("John Doe");
        student.setRollNo("12345");
        return student;
    }
}
```

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:

```
package pr18.pr18_1;

public class Exception1 {
    public void exceptionDemo() {
        try {
            System.out.println(2 / 0);
        }
        catch (ArithmeticException e)
        {
            System.out.println("Поделил на ноль?");
        }
    }
}
```

```
package pr18.pr18_1;

public class main {
    public static void main(String[] args) {
        Exception1 exc = new Exception1();
        exc.exceptionDemo();
    }
}
```

```
package pr18.pr18_2;
import java.util.Scanner;

public class Exception2 {
    {
        try {
            Scanner myScanner = new Scanner(System.in);
            System.out.print("Enter an integer ");
            String intString = myScanner.next();
            int i = Integer.parseInt(intString);
            System.out.println(2 / i);
        }
        catch (NumberFormatException e)
        {
            System.out.println("Неверный ввод");
        }
    }
}
```

```java
package pr18.pr18_2;
import java.util.Scanner;

public class Exception4 {
    public void exceptionDemo()
    {
        String intString = "";
        try {
            Scanner myScanner = new Scanner(System.in);
            System.out.print("Enter an integer ");
            intString = myScanner.next();
            int i = Integer.parseInt(intString);
            System.out.println(2 / i);
        }
        catch (NumberFormatException e)
        {
            System.out.println("Неверный ввод");
        }
        finally {
            System.out.printf("Программа завершилась");
        }
        //В Java ключевое слово finally используется для определения блока кода, который должен быть выполнен после блока try-catch,
        // независимо от того, произошло исключение или нет123.
        // Это означает, что блок finally всегда выполняется, даже если в блоке try произошло исключение123.
    }
}
```

Практическая работа № 19. Создание пользовательских исключений

Цель данной практической работы – научиться создавать собственные исключения.

Код:

```
package pr19;

public class InvalidINNException extends Exception {
    public InvalidINNException(String message) {
        super(message);
    }
}
/*Ключевое слово super в Java используется для обращения к членам суперкласса
(родительского класса) из подкласса (дочернего класса). */
```

```
package pr19;

public class OnlinePurchase {
    public void processOrder(String name, String inn) throws InvalidINNException {
        // Проверяем введенный ИНН на действительность
        if (!isValidINN(inn)) {
            throw new InvalidINNException("Недействительный ИНН: " + inn);
        }
        // Другая логика для обработки заказа
        System.out.println("Заказ успешно обработан.");
    }

    private boolean isValidINN(String inn) {
        // Вернуть true, если ИНН действителен, иначе false
        return inn.matches("\\d{10}");
        /*Это регулярное выражение используется для проверки строки на наличие ровно 10 цифр подряд. */
    }

    public static void main(String[] args) {
        OnlinePurchase purchase = new OnlinePurchase();

        try {
            purchase.processOrder("Иван Иванов", "123456890"); // Попробуйте ввести недействительный ИНН
        } catch (InvalidINNException e) {
            System.out.println("Ошибка: " + e.getMessage());
        }
    }
}
```

Практическая работа № 20. Работа с дженериками.

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

Код:

```
package pr20.pr20_1;

public class MyGenericClass<T, V, K> {
    private T t;
    private V v;
    private K k;

    public MyGenericClass(T t, V v, K k) {
        this.t = t;
        this.v = v;
        this.k = k;
    }

    public T getT() {
        return t;
    }

    public void setT(T t) {
        this.t = t;
    }

    public V getV() {
        return v;
    }

    public void setV(V v) {
        this.v = v;
    }

    public K getK() {
        return k;
    }

    public void setK(K k) {
        this.k = k;
    }

    public void printValues() {
        System.out.println("T: " + t);
        System.out.println("V: " + v);
        System.out.println("K: " + k);
    }

    public static void main(String[] args) {
        MyGenericClass<Integer, String, Double> myObject = new MyGenericClass<>(42, "Hello, World!", 3.14);
        myObject.printValues();
    }
}
```

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:

```
package pr21.pr21_1;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        // Создаем массив строк
        String[] stringArray = {"one", "two", "three"};
        // Создаем массив целых чисел
        Integer[] intArray = {1, 2, 3, 4, 5};

        // Конвертируем массив строк в список строк
        List<String> stringList = arrayToList(stringArray);
        // Конвертируем массив целых чисел в список целых чисел
        List<Integer> intList = arrayToList(intArray);

        // Выводим список строк на экран
        System.out.println("Список строк: " + stringList);
        // Выводим список чисел на экран
        System.out.println("Список чисел: " + intList);
    }

    // Метод для конвертации массива в список
    // Этот метод принимает массив элементов и возвращает список того же типа
    //  T является типом данных, который будет определен при вызове метода - generic с прошлой работы
    public static <T> List<T> arrayToList(T[] array) {
        // Создаем новый список, чтобы хранить элементы массива
        // Проходим по каждому элементу в массиве и добавляем его в список
        // Возвращаем список
        return new ArrayList<>(Arrays.asList(array));
    }
}

/*
List и ArrayList в Java тесно связаны, но они представляют разные вещи:

List:

List - это интерфейс, определенный в Java, который представляет упорядоченную коллекцию элементов, в которой могут быть дубликаты.
Он является частью Java Collections Framework и определяет базовые методы для работы с коллекциями, такие как add, get, remove, size, и другие.
List может быть реализован различными классами, такими как ArrayList, LinkedList, и Vector.
Использование интерфейса List позволяет легко заменять одну реализацию на другую без изменения кода.
ArrayList:

ArrayList - это одна из конкретных реализаций интерфейса List.
```

Он представляет собой динамический массив, который автоматически расширяется по мере добавления элементов. Это означает, что вы можете добавлять и удалять элементы из ArrayList без указания размера при создании.
ArrayList предоставляет высокую производительность доступа к элементам по индексу, что делает его хорошим выбором для операций чтения и записи.

*/

```
package pr21.pr21_2_and_3;

// Создание массива по заданному типу данных

public class GenericArray<T> {
    private T[] array;

    public GenericArray(int size) {
        // Инициализируем массив с заданным размером
        array = (T[]) new Object[size];
    }

    public void set(int index, T value) {
        // Устанавливаем значение по указанному индексу
        array[index] = value;
    }

    public T get(int index) {
        // Получаем значение по индексу
        return array[index];
    }

    public int length() {
        // Возвращаем длину массива
        return array.length;
    }

    public static void main(String[] args) {
        // Создаем объект GenericArray, хранящий целые числа
        GenericArray<Integer> intArray = new GenericArray<>(5);

        // Устанавливаем и получаем значения
        intArray.set(0, 1);
        intArray.set(1, 2);
        intArray.set(2, 3);

        System.out.println("Value at index 1: " + intArray.get(1));

        // Создаем объект GenericArray, хранящий строки
        GenericArray<String> stringArray = new GenericArray<>(3);

        // Устанавливаем и получаем значения
        stringArray.set(0, "Hello");
        stringArray.set(1, "World");

        System.out.println("Value at index 0: " + stringArray.get(0));
```

```
    }
}
```

```
package pr21.pr21_2_and_3;

public class MixedTypeArray {
    private Object[] array;

    public MixedTypeArray(int size) {
        array = new Object[size];
    }

    public void set(int index, Object value) {
        array[index] = value;
    }

    public Object get(int index) {
        return array[index];
    }

    public int length() {
        return array.length;
    }

    public static void main(String[] args) {
        MixedTypeArray mixedArray = new MixedTypeArray(5);

        mixedArray.set(0, 42);
        mixedArray.set(1, "Hello");
        mixedArray.set(2, 3.14);

        for (int i = 0; i < mixedArray.length(); i++) {
            System.out.println("Value at index " + i + ": " + mixedArray.get(i));
        }
    }
}
```

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:

```
package pr22;

import java.util.Stack;

public class RPNCalculator {
    // Метод для вычисления RPN-выражения

    public static double evaluateRPN(String expression) {
        Stack<Double> stack = new Stack<>();
        String[] tokens = expression.split(" ");

        for (String token : tokens) {
            if (isNumber(token)) {
                // Если токен является числом, помещаем его в стек
                stack.push(Double.parseDouble(token));
            } else if (isOperator(token)) {
                // Если токен является оператором, выполняем операцию над двумя верхними элементами
стека
                if (stack.size() < 2) {
                    throw new IllegalArgumentException("Неверное выражение");
                }
                double operand2 = stack.pop();
                double operand1 = stack.pop();
                double result = performOperation(operand1, operand2, token);
                stack.push(result);
            } else {
                // Если токен не является ни числом, ни оператором, выбрасываем исключение
                throw new IllegalArgumentException("Невозможный символ: " + token);
            }
        }

        if (stack.size() != 1) {
            throw new IllegalArgumentException("Неверное выражение");
        }

        return stack.pop();
    }

    // Метод для проверки, является ли строка числом
    private static boolean isNumber(String token) {
        try {
            Double.parseDouble(token);
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }

    // Метод для проверки, является ли строка оператором (+, -, *, /)
    private static boolean isOperator(String token) {
        return token.matches("[+\\-*/]");
```

```
    }

    // Метод для выполнения операции между двумя операндами в зависимости от оператора
    private static double performOperation(double operand1, double operand2, String operator) {
        switch (operator) {
            case "+":
                return operand1 + operand2;
            case "-":
                return operand1 - operand2;
            case "*":
                return operand1 * operand2;
            case "/":
                if (operand2 == 0) {
                    throw new ArithmeticException("Деление на ноль!");
                }
                return operand1 / operand2;
            default:
                throw new IllegalArgumentException("Неверный оператор!: " + operator);
        }
    }

    public static void main(String[] args) {
        String rpnExpression = "2 3 4 5 6 * + - /";
        double result = evaluateRPN(rpnExpression);
        System.out.println("Result: " + result);
    }
}

/*
"2 3 4 5 6 * + - /"
результат калькулятора на телефоне -0,064516129032258(064516129032258)
*/
```

Практическая работа № 23. Абстрактные типы данных. Очередь

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

Код:

```
package pr23.pr23_1;
public class ArrayQueue {
    private int front, rear, size;
    private int capacity;
    private int array[];

    public ArrayQueue(int capacity) {
        this.capacity = capacity;
        front = this.size = 0;
        rear = capacity - 1;
        array = new int[this.capacity];
    }

    // Очередь пуста, когда размер равен 0
    public boolean isEmpty() {
        return (this.size == 0);
    }

    // Очередь полна, когда размер равен максимальной вместимости
    public boolean isFull() {
        return (this.size == this.capacity);
    }

    public int size() {
        return this.size;
    }

    // Метод для добавления элемента в очередь.
    // Изменяет rear и size
    public void enqueue(int item) {
        if (isFull())
            return;
        this.rear = (this.rear + 1) % this.capacity;
        this.array[this.rear] = item;
        this.size = this.size + 1;
    }

    // Метод для удаления элемента из очереди.
    // Изменяет front и size
    public int dequeue() {
        if (isEmpty())
            return Integer.MIN_VALUE;

        int item = this.array[this.front];
        this.front = (this.front + 1) % this.capacity;
        this.size = this.size - 1;
        return item;
    }

    // Метод для получения front элемента очереди.
    public int element() {
```

```
        if (isEmpty())
            return Integer.MIN_VALUE;

        return this.array[this.front];
    }

    // Метод для получения rear элемента очереди.
    public int rear() {
        if (isEmpty())
            return Integer.MIN_VALUE;

        return this.array[this.rear];
    }

    public void clear() {
        for (int i = 0; i < size; i++) {
            array[(front + i) % capacity] = 0;
        }
        front = 0;
        size = 0;
        rear = capacity - 1;
    }

}
```

```
package pr23.pr23_1;

public class Test {
    public static void main(String[] args) {
        ArrayQueue queue = new ArrayQueue(4);

        queue.enqueue(10);
        queue.enqueue(20);
        queue.enqueue(30);
        queue.enqueue(40);

        System.out.println(queue.dequeue() + " удалён из очереди\n");

        System.out.println("Первый элемент " + queue.element());

        System.out.println("Последний элемент " + queue.rear());
        System.out.println("Размер очереди " + queue.size());
        System.out.println("Очередь пустая?  " + queue.isEmpty());
        System.out.println("Очередь полная?  " + queue.isFull());
        queue.enqueue(30);
        System.out.println("Очередь полная?  " + queue.isFull());
        queue.clear();
        System.out.println("Очередь пустая?  " + queue.isEmpty());
    }
}
```

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.");
    }
}
```