



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий

КАФЕДРА ИНСТРУМЕНТАЛЬНОГО И ПРИКЛАДНОГО
ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (ИиППО)

ПРАКТИЧЕСКИЕ РАБОТЫ
ПО ДИСЦИПЛИНЕ «Программирование на языке Джава»

Выполнил студент группы ИКБО-03-21 *Хречко С. В.*

Принял старший преподаватель *Рачков А.В.*

Практические работы работа выполнены «___»_____2022г.

«Зачтено» «___»_____2022г.

Москва 2022

Оглавление

Практическая работа № 1	3
Практическая работа № 2	5
Практическая работа № 3	7
Практическая работа № 4	12
Практическая работа № 5	15
Практическая работа № 18	19
Практическая работа № 19	21

Практическая работа № 1

Цель работы

Цель данной практической работы – освоить на практике работу с классами на Java.

Теоретическое введение

В Java, класс является определением объектов одного и того же вида. Другими словами, класс — это тип данных, создаваемый программистом для решения задач.

Экземпляр класса - реализация конкретного объекта типа класс. Другими словами, экземпляр экземпляра класса. Все экземпляры класса имеют аналогичные свойства, как задано в определении класса.

Переменные (или атрибуты, состояние, поля данных класса): содержит статические атрибуты класса, или описывают свойства класса (сущности предметной области).

Методы (или поведение, функции, работа с данными): описывают динамическое поведение класса.

Выполнения лабораторной работы

Задание:

Реализуйте простейший класс «Книга».

Решение:

Далее приведена реализация простейшего класса «Книга» (Листинг 1).

Листинг 1 – Код программы

```
public class Book {
    private String title;
    private int date;

    public Book(String title, int date) {
        this.title = title;
        this.date = date;
    }
    public Book(String title) {
        this.title = title;
        this.date = 2022;
    }
    public Book(int date) {
        this.title = "untitled";
        this.date = date;
    }
    public Book() {
        this.title = "untitled";
        this.date = 2022;
    }
}
```

```

    }

    public void setTitle(String newTitle) {
        this.title = newTitle;
    }
    public void setDate(int date) {
        this.date = date;
    }
    public String getTitle() {
        return(this.title);
    }
    public int getDate() {
        return(this.date);
    }

    @Override
    public String toString() {
        return title + " " + "dated: " + date;
    }
}

```

Далее приведен пример работы программы (рис. 1).

```

"C:\Program Files\Java\jdk-17.0.4.1\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Communit
untitled dated: 2022

Process finished with exit code 0

```

Рисунок 1 – Тестирование книги.

Выводы по работе:

Был реализован простейший класс «Книга».

Практическая работа № 2

Цель работы

Работа с UML-диаграммами классов.

Теоретическое введение

Язык моделирования Unified Modeling Language (UML) является стандартом де-факто с 1998 года для проектирования и документирования объектно-ориентированных программ. Средствами UML в виде диаграмм можно графически изобразить класс и экземпляр класса.

Графически представляем класс в виде прямоугольника, разделенного на три области – область именования класса, область инкапсуляции данных и область операций (методы).

Имя (или сущность) : определяет класс.

Переменные (или атрибуты, состояние, поля данных класса): содержит статические атрибуты класса, или описывают свойства класса (сущности предметной области).

Методы (или поведение, функции, работа с данными): описывают динамическое поведение класса. Другими словами, класс инкапсулирует статические свойства (данные) и динамические модели поведения (операции, которые работают с данными) в одном месте (“коробке” или прямоугольнике).

Выполнения лабораторной работы

Задание:

По диаграмме класса UML описывающей сущность Автор. Необходимо написать программу, которая состоит из двух классов Author и TestAuthor. Класс Author должен содержать реализацию методов, представленных на диаграмме класса на рисунке 1.

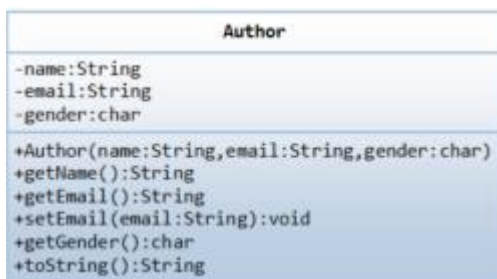


Рисунок 1 - Диаграмма класса Author.

Решение:

Далее приведена реализация класса «Author» (Листинг 1).

Листинг 1 – Код программы

```
public class Author {
    private String name;
```

```

private String email;
private char gender;

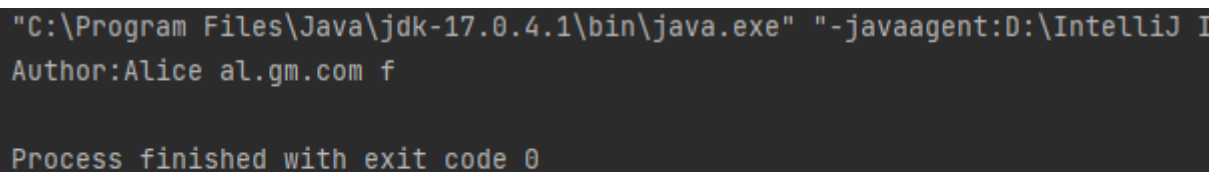
public Author(String name, String email, char gender){
    this.name = name;
    this.email = email;
    this.gender = gender;
}

public String getName(){
    return(this.name);
}
public String getEmail(){
    return(this.email);
}
public void setEmail(String email){
    this.email = email;
}
public char getGender(){
    return(this.gender);
}

@Override
public String toString() {
    return "Author:" + name + ' ' + email + ' ' + gender;
}
}

```

Далее приведен пример работы программы (рис. 2).



```

"C:\Program Files\Java\jdk-17.0.4.1\bin\java.exe" "-javaagent:D:\IntelliJ I
Author:Alice al.gm.com f

Process finished with exit code 0

```

Рисунок 2 – Тестирование автора.

Выводы по работе:

Был реализован класс «Author» по UML диаграмме.

Практическая работа № 3

Освоить на практике работу с абстрактными классами и наследованием на Java.

Цель работы

Теоретическое введение

Класс, содержащий абстрактные методы, называется абстрактным классом. Такие классы при определении помечаются ключевым словом `abstract`.

Абстрактный метод внутри абстрактного класса не имеет тела, только прототип. Он состоит только из объявления и не имеет тела.

Если вы объявляете класс, производный от абстрактного класса, но хотите иметь возможность создания объектов нового типа, вам придётся предоставить определения для всех абстрактных методов базового класса. Если этого не сделать, производный класс тоже останется абстрактным, и компилятор заставит пометить новый класс ключевым словом `abstract`.

Абстрактный класс не может содержать какие-либо объекты, а также абстрактные конструкторы и абстрактные статические методы. Любой подкласс абстрактного класса должен либо реализовать все абстрактные методы суперкласса, либо сам быть объявлен абстрактным.

Выполнения лабораторной работы

Задание:

Создайте абстрактный родительский суперкласс `Shape` и его дочерние классы (подклассы).

Решение:

Далее приведена реализация классов (Листинг 1).

Листинг 1 – Код программы

```
public abstract class Shape {  
    protected String color;  
    protected boolean filled;  
  
    public Shape() {  
        this.color = "green";  
        this.filled = true;  
    }  
  
    public Shape(String color, boolean filled) {  
        this.color = color;  
        this.filled = filled;  
    }  
  
    public String getColor() {
```

```

        return this.color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public boolean isFilled() {
        return this.filled;
    }

    public void setFilled(boolean filled) {
        this.filled = filled;
    }

    @Override
    public String toString() {
        return "A shapeTask.Shape with color of " + this.color + " and " +
        (this.filled ? "filled" : "not filled");
    }

    public abstract double getArea();

    public abstract double getPerimeter();

    public static void ShapeTest(){
        //commented lines are with errors
        Shape s1 = new Circle(5.5, "RED", false); // Upcast shapeTask.Circle to
shapeTask.Shape
        System.out.println(s1); // which version?
        System.out.println(s1.getArea()); // which version?
        System.out.println(s1.getPerimeter()); // which version?
        System.out.println(s1.getColor());
        System.out.println(s1.isFilled());
        //System.out.println(s1.getRadius()); //no radius in shape
        Circle c1 = (Circle)s1; // Downcast back to shapeTask.Circle
        System.out.println(c1);
        System.out.println(c1.getArea());
        System.out.println(c1.getPerimeter());
        System.out.println(c1.getColor());
        System.out.println(c1.isFilled());
        System.out.println(c1.getRadius());
        //shapeTask.Shape s2 = new shapeTask.Shape(); //shape is abstract
        Shape s3 = new Rectangle(1.0, 2.0, "RED", false); // Upcast
        System.out.println(s3);
        System.out.println(s3.getArea());
        System.out.println(s3.getPerimeter());
        System.out.println(s3.getColor());
        //System.out.println(s3.getLength()); //s3 is shape, no length in shape
        Rectangle r1 = (Rectangle)s3; // downcast
        System.out.println(r1);
        System.out.println(r1.getArea());
        System.out.println(r1.getColor());
        System.out.println(r1.getLength());
        Shape s4 = new Square(6.6); // Upcast
        System.out.println(s4);
        System.out.println(s4.getArea());
        System.out.println(s4.getColor());
        //System.out.println(s4.getSide()); //no side in shape
        // Take note that we downcast shapeTask.Shape s4 to
shapeTask.Rectangle,
        // which is a superclass of shapeTask.Square, instead of
shapeTask.Square
        Rectangle r2 = (Rectangle)s4;
    }

```



```

        System.out.println(r2);
        System.out.println(r2.getArea());
        System.out.println(r2.getColor());
        //System.out.println(r2.getSide()); //no get side in rectangle
        System.out.println(r2.getLength());
        // Downcast shapeTask.Rectangle r2 to shapeTask.Square
        Square sql = (Square)r2;
        System.out.println(sql);
        System.out.println(sql.getArea());
        System.out.println(sql.getColor());
        System.out.println(sql.getSide());
        System.out.println(sql.getLength());
    }
}

class Circle extends Shape {
    protected double radius;

    public Circle() {
        this.radius = 1.0;
    }

    public Circle(double radius) {
        this.radius = radius;
    }

    public Circle(double radius, String color, boolean filled) {
        super(color, filled); // call the parent constructor
        this.radius = radius;
    }

    public double getRadius() {
        return this.radius;
    }

    public void setRadius(double radius) {
        this.radius = radius;
    }

    @Override
    public double getArea() {
        return Math.PI * this.radius * this.radius;
    }

    @Override
    public double getPerimeter() {
        return 2 * Math.PI * this.radius;
    }

    @Override
    public String toString() {
        return "A shapeTask.Circle with radius=" + this.radius + ", which is a subclass of " + super.toString();
    }
}

class Rectangle extends Shape{
    protected double width;
    protected double length;

    public Rectangle() {
        this.width = 1.0;
    }
}

```

```

        this.length = 2.0;
    }

    public Rectangle(double width, double length) {
        this.width = width;
        this.length = length;
    }

    public Rectangle(double width, double length, String color, boolean
filled) {
        super(color, filled);
        this.width = width;
        this.length = length;
    }

    public double getWidth() {
        return this.width;
    }

    public void setWidth(double width) {
        this.width = width;
    }

    public double getLength() {
        return this.length;
    }

    public void setLength(double length) {
        this.length = length;
    }

    @Override
    public double getArea() {
        return this.width * this.length;
    }

    @Override
    public double getPerimeter() {
        return 2 * (this.width + this.length);
    }

    @Override
    public String toString() {
        return "A shapeTask.Rectangle with width=" + this.width + " and
length=" + this.length + ", which is a subclass of " + super.toString();
    }
}

class Square extends Rectangle{
    public Square() {
        super(1.0, 1.0);
    }

    public Square(double side) {
        super(side, side);
    }

    public Square(double side, String color, boolean filled) {
        super(side, side, color, filled);
    }

    public double getSide() {
        return this.width;
    }
}

```

```

    }

    public void setSide(double side) {
        this.width = side;
        this.length = side;
    }

    @Override
    public void setWidth(double side) {
        this.setSide(side);
    }

    @Override
    public void setLength(double side) {
        this.setSide(side);
    }

    @Override
    public String toString() {
        return "A shapeTask.Square with side=" + this.width + ", which is a subclass of " + super.toString();
    }
}

```

Далее приведен пример работы программы (рис. 1).

```

"C:\Program Files\Java\jdk-17.0.4.1\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edition 2022.2.2\lib\idea_rt.jar=52543:D:\IntelliJ IDEA Community Edition 2022.2.2\bin" -Dfile.encoding=UTF-8
A shapeTask.Circle with radius=5.5, which is a subclass of A shapeTask.Shape with color of RED and not filled
95.03317777109123
34.55751918948772
RED
false
A shapeTask.Circle with radius=5.5, which is a subclass of A shapeTask.Shape with color of RED and not filled
95.03317777109123
34.55751918948772
RED
false
5.5
A shapeTask.Rectangle with width=1.0 and length=2.0, which is a subclass of A shapeTask.Shape with color of RED and not filled
2.0
0.0
RED
A shapeTask.Rectangle with width=1.0 and length=2.0, which is a subclass of A shapeTask.Shape with color of RED and not filled
2.0
RED
2.0
A shapeTask.Square with side=0.0, which is a subclass of A shapeTask.Rectangle with width=0.0 and length=0.0, which is a subclass of A shapeTask.Shape with color of green and filled
43.559999999999995
green
A shapeTask.Square with side=0.0, which is a subclass of A shapeTask.Rectangle with width=0.0 and length=0.0, which is a subclass of A shapeTask.Shape with color of green and filled
43.559999999999995
green
0.0
A shapeTask.Square with side=0.0, which is a subclass of A shapeTask.Rectangle with width=0.0 and length=0.0, which is a subclass of A shapeTask.Shape with color of green and filled
43.559999999999995
green
0.0
0.0
Process finished with exit code 0

```

Рисунок 1 – Тестирование супер класса и дочерних классов.

Выводы по работе:

Были реализованы класс Shape и его дочерние классы.

Практическая работа № 4

Цель работы

Введение в событийное программирование на языке Java.

Теоретическое введение

Text Fields - текстовое поле или поля для ввода текста (можно ввести только одну строку). Примерами текстовых полей являются поля для ввода логина и пароля, например, используемые, при входе в электронную почту.

Компонент TextArea похож на TextField, но в него можно вводить более одной строки. В качестве примера TextArea можно рассмотреть текст, который мы набираем в теле сообщения электронной почты.

Разделяет компонент на пять областей (WEST, EAST, NORTH, SOUTH and Center). Другие компоненты могут быть добавлены в любой из этих компонентов пятерками.

Мы можем реализовывать слушателей мыши и также слушателей клавиатуры на компонентах GUI.

Выполнения лабораторной работы

Задание:

Напишите интерактивную программу с использованием GUI имитирует таблицу результатов матчей между командами Милан и Мадрид. Создайте JFrame приложение у которого есть следующие компоненты GUI:

- одна кнопка JButton labeled “AC Milan”
- другая JButton подписана “Real Madrid”
- надпись JLabel содержит текст “Result: 0 X 0”
- надпись JLabel содержит текст “Last Scorer: N/A”
- надпись JLabel содержит текст “Winner: DRAW”;

Всякий раз, когда пользователь нажимает на кнопку AC Milan, результат будет увеличиваться для Милана, сначала 1 X 0, затем 2 X 0 и так далее. Last Scorer означает последнюю забившую команду. В этом случае: AC Milan. Если пользователь нажимает кнопку для команды Мадрид, то счет приписывается ей. Победителем становится команда, которая имеет больше кликов кнопку на соответствующую, чем другая.

Решение:

Далее приведена реализация программы (Листинг 1).

Листинг 1 – Код программы

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class FootballFrame extends JFrame{
    private int firstTeamScore = 0;
    private int secondTeamScore = 0;
    private String firstTeamName = "AC Milan";
    private String secondTeamName = "Real Madrid";
    JLabel score = new JLabel("0 x 0");
```

```

JLabel winner = new JLabel("winner: DRAW");
JLabel lastScorer = new JLabel("last scorer: N/A");
JButton firstTeamGoal = new JButton(firstTeamName);
JButton secondTeamGoal = new JButton(secondTeamName);

public FootballFrame() {
    super("Football");
    setSize(600, 400);
    setLayout(new BorderLayout());
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    firstTeamGoal.setBounds(0, 150, 100, 50);
    add(firstTeamGoal, BorderLayout.WEST);
    secondTeamGoal.setBounds(0, 150, 100, 50);
    add(secondTeamGoal, BorderLayout.EAST);
    score.setBounds(0, 175, 100, 50);
    add(score, BorderLayout.CENTER);
    winner.setSize(100, 50);
    add(winner, BorderLayout.SOUTH);
    lastScorer.setSize(100, 50);
    add(lastScorer, BorderLayout.NORTH);

    firstTeamGoal.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            firstTeamScore++;
            score.setText(firstTeamScore + " x " + secondTeamScore);
            lastScorer.setText("last scorer: " + firstTeamName);
            if (firstTeamScore > secondTeamScore) {
                winner.setText("winner: " + firstTeamName);
            }
            else if (firstTeamScore < secondTeamScore) {
                winner.setText("winner: " + secondTeamName);
            }
            else {
                winner.setText("winner: DRAW");
            }
        }
    });

    secondTeamGoal.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            secondTeamScore++;
            score.setText(firstTeamScore + " x " + secondTeamScore);
            lastScorer.setText("last scorer: " + secondTeamName);
            if (firstTeamScore > secondTeamScore) {
                winner.setText("winner: " + firstTeamName);
            }
            else if (firstTeamScore < secondTeamScore) {
                winner.setText("winner: " + secondTeamName);
            }
            else {
                winner.setText("winner: DRAW");
            }
        }
    });
}
}

```

Далее приведен пример работы программы (рис. 1).



Рисунок 1 – Пример работы программы.

Выводы по работе:

Была реализована программа в соответствии с заданием, были освоены базовые навыки событийного программирования в Java.

Практическая работа № 5

Цель работы

Разработка и программирование рекурсивных алгоритмов на языке Java.

Теоретическое введение

В контексте языка программирования рекурсия — это некий активный метод (или подпрограмма) вызываемый сам по себе непосредственно, или вызываемой другим методом (или подпрограммой) косвенно. В первую очередь надо понимать, что рекурсия — это своего рода перебор. Вообще говоря, всё то, что решается итеративно можно решить рекурсивно, то есть с использованием рекурсивной функции.

Так же, как и у перебора (цикла) у рекурсии должно быть условие остановки — базовый случай (иначе также, как и цикл, рекурсия будет работать вечно — infinite). Это условие и является тем случаем, к которому рекурсия идет (шаг рекурсии). При каждом шаге вызывается рекурсивная функция до тех пор, пока при следующем вызове не сработает базовое условие и не произойдет остановка рекурсии (а точнее возврат к последнему вызову функции). Всё решение сводится к поиску решения для базового случая. В случае, когда рекурсивная функция вызывается для решения сложной задачи (не базового случая) выполняется некоторое количество рекурсивных вызовов или шагов, с целью сведения задачи к более простой. И так до тех пор, пока не получим базовое решение.

Выполнения лабораторной работы

Задание:

1. Треугольная последовательность

Дана монотонная последовательность, в которой каждое натуральное число k встречается ровно k раз: 1, 2, 2, 3, 3, 3, 4, 4, 4, 4,...

По данному натуральному n выведите первые n членов этой последовательности. Попробуйте обойтись только одним циклом `for`.

2. От 1 до n

Дано натуральное число n . Выведите все числа от 1 до n .

3. От A до B

Даны два целых числа A и B (каждое в отдельной строке). Выведите все числа от A до B включительно, в порядке возрастания, если $A < B$, или в порядке убывания в противном случае.

Решение:

Далее приведена программа решающая поставленные задачи. (Листинг 1).

Листинг 1 – Код программы

```
import java.util.Scanner;  
  
//todo more tasks
```

```

public class RecursionTasks {
    private static Scanner in = new Scanner(System.in);

    public static void taskPicker(){
        System.out.println("Choose a task to run: 1-3 (print number (no fool
proof))");
        int task = in.nextInt();
        switch (task){
            case 1:
                task1();
                break;
            case 2:
                task2();
                break;
            case 3:
                task3();
                break;
            default:
                break;
        }
    }

    private static void task1(){
        System.out.println("Enter a number");
        int n = in.nextInt();
        task1Rec(n);
    }
    private static void task1Rec(int n){
        if(n > 0){
            task1Rec(n-1);
            for(int i = 0; i < n; i++){
                System.out.print(n + " ");
            }
        }
        return;
    }

    private static void task2(){
        System.out.println("Enter a number");
        int n = in.nextInt();
        task2Rec(n);
    }
    private static void task2Rec(int n){
        if(n > 0){
            task2Rec(n-1);
            System.out.print(n + " ");
        }
        return;
    }

    private static void task3(){
        System.out.println("Enter two numbers");
        int a = in.nextInt();
        int b = in.nextInt();
        task3Rec(a, b);
    }
    private static void task3Rec(int a, int b){
        if(a != b){
            System.out.print(a + " ");
            task3Rec(a + (b-a)/Math.abs(b-a), b); //I think this is less
efficient than if... but... it's more fun

```



```

    }
    else {
        System.out.println(a);
    }
}
}

```

Далее приведены примеры работы программы, решение задачи 1-3 (рис. 1-3).

```

"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:D:\JetBrains\IntelliJ IDEA Commu
Choose a task to run: 1-3 (print number (no fool proof))
1
Enter a number
7
1 2 2 3 3 3 4 4 4 4 5 5 5 5 6 6 6 6 6 6 7 7 7 7 7 7
Process finished with exit code 0

```

Рисунок 1 – Задача 1.

```

"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:D:\JetBrains\Int
Choose a task to run: 1-3 (print number (no fool proof))
2
Enter a number
5
1 2 3 4 5
Process finished with exit code 0
|

```

Рисунок 2 – Задача 2.

```

"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:D:\JetE
Choose a task to run: 1-3 (print number (no fool proof))
3
Enter two numbers
7 2
7 6 5 4 3 2

Process finished with exit code 0

```

Рисунок 3 – Задача 3.

Выводы по работе:

Были выполнены поставленные задачи по использованию рекурсии.

Практическая работа № 18

Цель работы

Целями 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.

Теоретическое введение

Чтобы объявить переменную, необходимо указать тип переменной и ее имя. Типы переменной могут быть разные: целочисленный(long, int, short, byte), число с плавающей запятой(double, float), логический(boolean), перечисление, объектный(Object).

Массив — это конечная последовательность упорядоченных элементов одного типа, доступ к каждому элементу в которой осуществляется по его индексу.

Цикл - это конструкция, позволяющая выполнять определенную часть кода несколько раз. В Java есть три типа циклов for, while, do while. Цикл for - это цикл со счетчиком, обычно используется, когда известно, сколько раз должна выполняться определенная часть кода.

Для ввода данных используется класс Scanner из библиотеки пакетов

Этот класс надо импортировать в той программе, где он будет использоваться. Это делается до начала открытого класса в коде программы.

В классе есть методы для чтения очередного символа заданного типа со стандартного потока ввода, а также для проверки существования такого символа.

Для работы с потоком ввода необходимо создать объект класса Scanner, при создании указав, с каким потоком ввода он будет связан. Стандартный поток ввода (клавиатура) в Java представлен объектом — System.in. А стандартный поток вывода (дисплей) — уже знакомым вам объектом System.out. Есть ещё стандартный поток для вывода ошибок — System.err

Выполнения лабораторной работы

Задание:

Сгенерировать массив целых чисел случайным образом, вывести его на экран, отсортировать его, и снова вывести на экран.

Решение:

Далее приведен код программы выполняющей поставленную задачу (Листинг 1).

Листинг 1 – Код программы

```
import java.util.Arrays;

//variant 4
public class ArrGenerator {
    public static void generate(){ //main
        int n = 10;
        int[] a = new int[n];
        for(int i = 0; i < n; i++){
            a[i] = -100 + (int) (Math.random() * 200);
        }
    }
}
```

```
    }  
    for(int i =0; i < n; i++){  
        System.out.print(a[i] + " ");  
    }  
    System.out.println();  
    Arrays.sort(a);  
    System.out.println(Arrays.toString(a));  
}  
}
```

Далее приведены примеры работы программы (рис. 1).

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:D:\JetBrains\In  
10 48 -99 -23 -60 45 -20 2 5 15  
[-99, -60, -23, -20, 2, 5, 10, 15, 48]  
  
Process finished with exit code 0
```

Рисунок 1 – Работа программы.

Выводы по работе:

Была выполнена поставленная задача.

Практическая работа № 19

Цель работы

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

Теоретическое введение

Язык Java - объектно-ориентированный язык программирования. В центре ООП находится понятие объекта. Объект — это сущность, которой можно посылать сообщения и которая может на них реагировать, используя свои данные. Объект — это экземпляр класса. Данные объекта скрыты от остальной программы. Скрытие данных называется инкапсуляцией.

Наличие инкапсуляции достаточно для объектности языка программирования, но ещё не означает его объектной ориентированности — для этого требуется наличие наследования.

Но даже наличие инкапсуляции и наследования не делает язык программирования в полной мере объектным с точки зрения ООП. Основные преимущества ООП проявляются только в том случае, когда в языке программирования реализован полиморфизм подтипов — возможность единообразно обрабатывать объекты с различной реализацией при условии наличия общего интерфейса.

Класс в ООП — это в чистом виде абстрактный тип данных, создаваемый программистом. С этой точки зрения объекты являются значениями данного абстрактного типа, а определение класса задаёт внутреннюю структуру значений и набор операций, которые над этими значениями могут быть выполнены. Желательность иерархии классов (а значит, наследования) вытекает из требований к повторному использованию кода — если несколько классов имеют сходное поведение, нет смысла дублировать их описание, лучше выделить общую часть в общий родительский класс, а в описании самих этих классов оставить только различающиеся элементы.

Выполнения лабораторной работы

Задание:

Создать класс, описывающий тело человека(Human). Для описания каждой части тела создать отдельные классы(Head, Leg, Hand). Описать необходимые свойства и методы для каждого класса. Протестировать работу класса Human.

Решение:

Далее приведена реализация класса «Human» и дополнительных классов (Листинг 1).

Листинг 1 – код программы

```
import java.util.ArrayList;
import java.util.List;

public class Human {
    private List<Head> heads;
```

```

private List<Leg> legs;
private List<Hand> hands;
private int health;

public Human() {
    this.heads = new ArrayList<>();
    this.heads.add(new Head());
    this.legs = new ArrayList<>();
    this.legs.add(new Leg(10, "right"));
    this.legs.add(new Leg(10, "left"));
    this.hands = new ArrayList<>();
    this.hands.add(new Hand(5, "right"));
    this.hands.add(new Hand(5, "left"));
    this.health = 100;
}

public Human(List<Head> heads, List<Leg> legs, List<Hand> hands, int
health) {
    this.heads = new ArrayList<>(heads);
    this.legs = new ArrayList<>(legs);
    this.hands = new ArrayList<>(hands);
    this.health = health;
}

public Head cutHead() {
    if (heads.size() > 0) return (heads.remove(0));
    return (null);
}

public boolean cutHead(Head head) {
    return (heads.remove(head));
}

public void addHead(Head head) {
    this.heads.add(head);
}

public Hand cutHand() {
    if (hands.size() > 0) return (hands.remove(0));
    return (null);
}

public boolean cutHand(Hand hand) {
    return (hands.remove(hand));
}

public void addHand(Hand hand) {
    this.hands.add(hand);
}

public Leg cutLeg() {
    if (legs.size() > 0) return (legs.remove(0));
    return (null);
}

public boolean cutLeg(Leg leg) {
    return (legs.remove(leg));
}

public void addLeg(Leg leg) {
    this.legs.add(leg);
}

public int getHealth() {
    return health;
}

public void setHealth(int health) {
    this.health = health;
}

```

```

@Override
public String toString() {
    return "Human{\n" +
        "  heads=" + heads +
        ",\n legs=" + legs +
        ",\n hands=" + hands +
        ",\n health=" + health +
        "\n}";
}

public void fight(){
    for(Hand hand : hands){
        hand.grab();
    }
    for(Leg leg : legs){
        leg.kick();
    }
    this.health -= 10;
}

public static void humanTest(){
    Human human = new Human();
    System.out.println(human.toString());
    System.out.println();
    human.fight();
    System.out.println();
    Hand thirdHand = new Hand(3, "middle");
    human.addHand(thirdHand);
    human.cutHead();
    System.out.println(human);
}
}

public class Head {
    private int eyeCount;
    private int intelligence;

    public Head(){
        this.eyeCount = 2;
        this.intelligence = 10;
    }
    public Head(int eyeCount, int intelligence){
        this.eyeCount = eyeCount;
        this.intelligence = intelligence;
    }

    public int getEyeCount() {
        return eyeCount;
    }

    public void setEyeCount(int eyeCount) {
        this.eyeCount = eyeCount;
    }

    public int getIntelligence() {
        return intelligence;
    }

    public void setIntelligence(int intelligence) {
        this.intelligence = intelligence;
    }

    @Override
    public String toString() {

```

```

        return "Head{" +
            "eyeCount=" + eyeCount +
            ", intelligence=" + intelligence +
            '}';
    }

    public void think(){
        System.out.println("some thoughts for " + this.intelligence + "
intelligence");
    }
}

public class Hand {
    private int fingerNumber;
    private String orientation;

    public Hand(int fingerNumber, String orientation){
        this.fingerNumber = fingerNumber;
        orientation.toLowerCase();
        this.orientation = orientation;
    }

    public Hand(){
        this.fingerNumber = 5;
        this.orientation = "right";
    }

    public void setFingerNumber(int fingerNumber){
        this.fingerNumber = fingerNumber;
    }

    public int getFingerNumber(){
        return(this.fingerNumber);
    }

    public void setOrientation(String orientation){
        this.orientation = orientation;
    }

    public String getOrientation(){
        return(this.orientation);
    }

    @Override
    public String toString() {
        return "Hand{" +
            "fingerNumber=" + fingerNumber +
            ", orientation=" + orientation + '\n' +
            '}';
    }

    public void grab(){
        System.out.println(this.orientation + " hand grabbed nothing");
    }
}

public class Leg {
    private int strength;
    private String orientation;

    public Leg(int strength, String orientation) {
        this.strength = strength;
        orientation.toLowerCase();
        this.orientation = orientation;
    }

    public Leg(){
        this.strength = 10;
    }
}

```



```

        this.orientation = "right";
    }

    public int getStrength() {
        return strength;
    }

    public void setStrength(int strength) {
        this.strength = strength;
    }

    public String getOrientation() {
        return orientation;
    }

    public void setOrientation(String orientation) {
        this.orientation = orientation;
    }

    @Override
    public String toString() {
        return "Leg{" +
            "strength=" + strength +
            ", orientation='" + orientation + '\'' +
            '}';
    }

    public void kick() {
        System.out.println(this.orientation + " leg kicked with strength " +
            this.strength);
    }
}

```

Далее приведен пример тестирования программы (рис. 1).

```

"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:D:\JetBrains\IntelliJ IDEA Community Edition 2021.2.3\lib\idea_rt.jar=58534:D:\JetBrains\
Human{
  heads=[Head{eyeCount=2, intelligence=10}],
  legs=[Leg{strength=10, orientation='right'}, Leg{strength=10, orientation='left'}],
  hands=[Hand{fingerNumber=5, orientation='right'}, Hand{fingerNumber=5, orientation='left'}],
  health=100
}

right hand grabbed nothing
left hand grabbed nothing
right leg kicked with strength 10
left leg kicked with strength 10

Human{
  heads=[],
  legs=[Leg{strength=10, orientation='right'}, Leg{strength=10, orientation='left'}],
  hands=[Hand{fingerNumber=5, orientation='right'}, Hand{fingerNumber=5, orientation='left'}, Hand{fingerNumber=3, orientation='middle'}],
  health=90
}

Process finished with exit code 0

```

Рисунок 1 – Тестирование.

Выводы по работе:

Был реализован класс «Human» и дополнительные классы «Head», «Hand», «Leg».

Используемая литература (в конце всего отчета)

Список литературы тоже должен быть оформлен по ГОСТ. Первым пунктом можно указать:

- 1. Конспект лекций по дисциплине «Программирование на языке Джава», РТУ МИРЭА, лектор – старший преподаватель Зорина Н.В.*
- 2. Карпова, И.П. Базы данных: Учебное пособие / И.П. Карпова. –СПб.: Питер, 2013. – 240 с.*
- 3. Фрэйз Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств – Питер: 2016г. Режим доступа свободный: https://www.htbook.ru/kompjutery_i_seti/setevye_tekhnologii/html5-i-css3-razrabotka-sajtov-dlja-ljubyx-brauzerov-i-ustrojstv;*
- 4. Справочник по языку PHP [Электронный ресурс]:php.su— Режим доступа свободный: <http://www.php.su>;*
- 5. И т.д.*