



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 № 6 .....	19
Практическая работа № 7 .....	21
Практическая работа № 18 .....	24
Практическая работа № 19 .....	26
Практическая работа № 20 .....	31
Практическая работа № 21 .....	34
Практическая работа № 22 .....	37
Практическая работа № 23 .....	42
Практическая работа № 24 .....	45

## 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

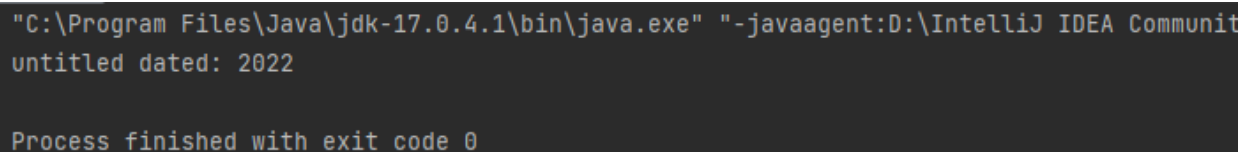
```
public class Book {  
    private String title;  
    private int date;  
  
    public Book(String title, int date) {  
        this.title = title;  
        this.date = date;  
    }  
    public Book(String title) {  
        this.title = title;  
        this.date = 2022;  
    }  
    public 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 Community Edition\lib\idea_rt.jar=2022:C:\Program Files\Java\jdk-17.0.4.1\bin\java.exe"
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.

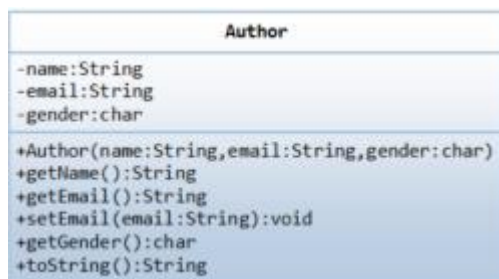


Рисунок 1 - Диаграмма класса Author.

#### Решение:

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

#### Листинг 1 – Код программы

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

```

private String email;
private char gender;

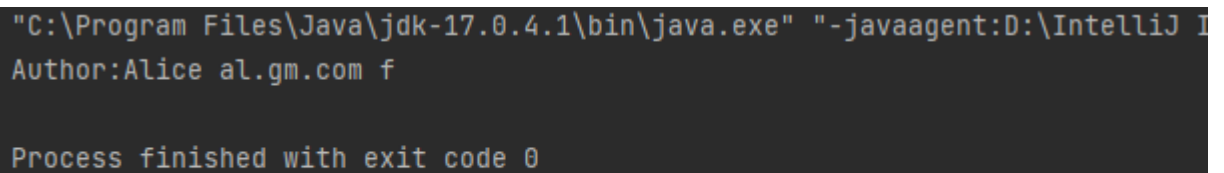
public Author(String name, String email, char gender){
    this.name = name;
    this.email = email;
    this.gender = gender;
}

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
6.0
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=6.6, which is a subclass of A shapeTask.Rectangle with width=6.6 and length=6.6, which is a subclass of A shapeTask.Shape with color of green and filled
43.559999999999995
green
A shapeTask.Square with side=6.6, which is a subclass of A shapeTask.Rectangle with width=6.6 and length=6.6, which is a subclass of A shapeTask.Shape with color of green and filled
43.559999999999995
green
6.6
A shapeTask.Square with side=6.6, which is a subclass of A shapeTask.Rectangle with width=6.6 and length=6.6, which is a subclass of A shapeTask.Shape with color of green and filled
43.559999999999995
green
6.6
6.6
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:

Был 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 № 6

### Цель работы

Освоение на практике методов сортировки с использованием приемов программирования на объектно-ориентированном языке Java.

### Теоретическое введение

Сортировка — это процесс упорядочивания списка элементов (организация в определенном порядке) исходного списка элементов, который возможно организован в виде контейнера или храниться в виде коллекции.

Техника программирования сортировок в Java отличается от написания алгоритмов на процедурных языках программирования. При написании кода большим преимуществом является использование основного принципа ООП — полиморфизма. Напомним, что класс, который реализует интерфейс Comparable определяет метод compareTo(), чтобы определить относительный порядок своих объектов.

Таким образом мы можем использовать полиморфизм, чтобы разработать обобщенную сортировку для любого набора Comparable объектов.

### Выполнения лабораторной работы

#### Задание:

Написать тестовый класс, который создает массив класса Student и сортирует массив idNumber и сортирует его вставками.

#### Решение:

Далее приведена реализация класса «Студент», с методом сортировки (Листинг 1).

#### Листинг 1 – Код программы

```
public class Student {  
    private String name;  
    private int idNumber;  
  
    public Student(String name, int idNumber) {  
        this.name = name;  
        this.idNumber = idNumber;  
    }  
    public Student() {  
        this.name = "John Doe";  
        this.idNumber = 0;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
    public int getIdNumber() {  
        return idNumber;  
    }  
}
```

```

public void setIdNumber(int idNumber) {
    this.idNumber = idNumber;
}
public String toString() {
    return "[" + name + ", idNumber " + idNumber + "]";
}

public static void studentTest(){
    Student[] students = new Student[10];
    for (int i = 0; i < students.length; i++) {
        students[i] = new Student("Student " + i, (int) (Math.random()*100));
    }
    System.out.println(Arrays.toString(students));
    insertionSort(students);
    System.out.println(Arrays.toString(students));
}
public static void insertionSort(Student[] arr){
    for (int i = 1; i < arr.length; i++) {
        Student temp = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j].idNumber > temp.idNumber) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = temp;
    }
}
}

```

Далее приведен пример работы 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=52692:D:\JetBrains\IntelliJ IDEA Community Edition 2021.2.3\bin" -Dfile.encoding=UTF-8 -classpath "D:\Sergey Kh
[[Student 0, idNumber 44], [Student 1, idNumber 76], [Student 2, idNumber 38], [Student 3, idNumber 2], [Student 4, idNumber 44], [Student 5, idNumber 29], [Student 6, idNumber 69], [Student 7, idNumber 44], [Student 8, idNumber 16], [Student
[[Student 3, idNumber 2], [Student 8, idNumber 16], [Student 9, idNumber 21], [Student 5, idNumber 29], [Student 2, idNumber 38], [Student 0, idNumber 44], [Student 4, idNumber 44], [Student 7, idNumber 44], [Student 6, idNumber 69], [Student
Process finished with exit code 0

```

Рисунок 1 – Тестирование.

### Выводы по работе:

Был реализован класс «Студент», и сортировка.

## Практическая работа № 7

### Цель работы

Изучение на практике приемов работы со стандартными контейнерными классами Java Collection Framework.

### Теоретическое введение

Java Collections Framework — это набор связанных классов и интерфейсов, реализующих широко используемые структуры данных — коллекции. На вершине иерархии в Java Collection Framework располагаются 2 интерфейса: Collection и Map. Эти интерфейсы разделяют все коллекции, входящие в фреймворк на две части по типу хранения данных: простые последовательные наборы элементов и наборы пар «ключ — значение» (словари).

### Выполнения лабораторной работы

#### Задание:

Напишите программу в виде консольного приложения, которая моделирует карточную игру «пьяница» и определяет, кто выигрывает. В игре участвует 10 карт, имеющих значения от 0 до 9, большая карта побеждает меньшую; карта «0» побеждает карту «9».

Карточная игра “ В пьяницу”. В этой игре карточная колода раздается поровну двум игрокам. Далее они открывают по одной верхней карте, и тот, чья карта старше, забирает себе обе открытые карты, которые кладутся под низ его колоды. Тот, кто остается без карт, - проигрывает.

Для простоты будем считать, что все карты различны по номиналу и что самая младшая карта побеждает самую старшую карту (“шестерка берет туз”).

Игрок, который забирает себе карты, сначала кладет под низ своей колоды карту первого игрока, затем карту второго игрока (то есть карта второго игрока оказывается внизу колоды).

Входные данные.

Программа получает на вход две строки: первая строка содержит 5 карт первого игрока, вторая - 5 карт второго игрока. Карты перечислены сверху вниз, то есть каждая строка начинается с той карты, которая будет открыта первой.

Выходные данные.

Программа должна определить, кто выигрывает при данной раздаче, и вывести слово first или second, после чего вывести количество ходов, сделанных до выигрыша. Если на протяжении 106 ходов игра не заканчивается, программа должна вывести слово botva.

#### Решение:

Далее приведена реализация класса игры в Пьяницу (Листинг 1).

#### Листинг 1 – Код программы

```
public class DrunkGame {  
    public static void play() {  
        int[] arr = {1, 3, 5, 7, 9, 2, 4, 6, 8, 0};  
        for (int i = 0; i < arr.length; i++) {  
            int rand = (int) (Math.random() * arr.length);
```

```

        int temp = arr[i];
        arr[i] = arr[rand];
        arr[rand] = temp;
    }
    System.out.println("cards are:");
    for(int i = 0; i < 5; i++){
        System.out.print(arr[i] + " ");
    }
    System.out.println();
    for(int i = 5; i < 10; i++){
        System.out.print(arr[i] + " ");
    }
    System.out.println();
    ArrayDeque<Integer> player1 = new ArrayDeque<>();
    ArrayDeque<Integer> player2 = new ArrayDeque<>();
    for(int i = 0; i < 5; i++){
        player1.add(arr[i]);
    }
    for(int i = 5; i < 10; i++){
        player2.add(arr[i]);
    }
    for(int turn = 1; turn <= 106; turn++){
        int card1 = player1.poll();
        int card2 = player2.poll();
        if(card1 == 0 && card2 == 9){
            player1.add(card1);
            player1.add(card2);
        }
        else if(card1 == 9 && card2 == 0){
            player2.add(card1);
            player2.add(card2);
        }
        else if(card1 > card2){
            player1.add(card1);
            player1.add(card2);
        }
        else if(card1 < card2){
            player2.add(card1);
            player2.add(card2);
        }
        if(player1.isEmpty()){
            System.out.println("second " + turn);
            return;
        }
        else if(player2.isEmpty()){
            System.out.println("first " + turn);
            return;
        }
    }

    //print turn
    System.out.println("turn: " + turn);
    System.out.println("player 1 card: " + card1);
    System.out.println("player 2 card: " + card2);
}
System.out.println("botva");
}
}

```

Далее приведен пример работы программы (рис. 1).

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:D:\JetBrains\IntelliJ IDEA Communi  
cards are:  
5 6 9 7 1  
3 2 0 8 4  
second 13  
  
Process finished with exit code 0
```

Рисунок 1 – Тестирование.

**Выводы по работе:**

Была реализована игра в Пьяницу.

## 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 + '\'' +
            '}';
    }

    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».

## Практическая работа № 20

### Цель работы

Цель данной практической работы - изучить понятие наследования, и научиться реализовывать наследование в Java.

### Теоретическое введение

Одним из ключевых аспектов объектно-ориентированного программирования является наследование. С помощью наследования можно расширить функционал уже имеющихся классов за счет добавления нового функционала или изменения старого.

Хотя наследование очень интересный и эффективный механизм, но в некоторых ситуациях его применение может быть нежелательным. И в этом случае можно запретить наследование с помощью ключевого слова `final`.

### Выполнения лабораторной работы

#### Задание:

Создать абстрактный класс, описывающий собак(Dog). С помощью наследования реализовать различные породы собак. Протестировать работу классов.

#### Решение:

Далее приведена реализация абстрактного класса (Листинг 1).

#### Листинг 1 – Код собаки

```
public abstract class Dog {  
    protected String name;  
    protected int age;  
  
    public Dog() {  
        this.name = "Dog";  
        this.age = 0;  
    }  
  
    public Dog(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
  
    public String getName() {  
        return this.name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public int getAge() {  
        return this.age;  
    }  
}
```

```

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "A dog with name of " + this.name + ", age of " + this.age;
    }

    public abstract void bark();

    public static void dogTest() {
        Dog[] dogs = new Dog[4];
        dogs[0] = new FancyDog("MeatStick", 5, "Poodle");
        dogs[1] = new FancyDog("FunnyEars", 5, "Poodle with strange ears");
        dogs[2] = new HunterDog("Alfred", 5, "Butler hound");
        dogs[3] = new HunterDog("Roberto", 5, "Hound");

        for (Dog dog : dogs) {
            System.out.println(dog.toString());
            dog.bark();
        }
    }
}

```

Далее приведена реализация классов пород собак (Листинг 2).

#### Листинг 2 – Код пород собак

```

public class FancyDog extends Dog {
    private String breed;

    public FancyDog() {
        super();
        this.breed = "Poodle";
    }

    public FancyDog(String name, int age, String breed) {
        super(name, age);
        this.breed = breed;
    }

    public String getBreed() {
        return this.breed;
    }

    public void setBreed(String breed) {
        this.breed = breed;
    }

    @Override
    public String toString() {
        return "A fancy dog with name of " + this.name + ", age of " +
this.age + ", breed of " + this.breed;
    }

    @Override
    public void bark() {
        System.out.println("Woof! Woof! Woof!");
    }
}

```



```

public class HunterDog extends Dog {
    private String breed;

    public HunterDog() {
        super();
        this.breed = "Hound";
    }

    public HunterDog(String name, int age, String breed) {
        super(name, age);
        this.breed = breed;
    }

    public String getBreed() {
        return this.breed;
    }

    public void setBreed(String breed) {
        this.breed = breed;
    }

    @Override
    public String toString() {
        return "A hunter dog with name of " + this.name + ", age of " +
this.age + ", breed of " + this.breed;
    }

    @Override
    public void bark() {
        System.out.println("Bark! Bark! Bark!");
    }
}

```

Далее приведен пример работы программы (рис. 1).

```

"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:D:\JetBrains\IntelliJ IDEA Community Edition 2
A fancy dog with name of MeatStick, age of 5, breed of Poodle
Woof! Woof! Woof!
A fancy dog with name of FunnyEars, age of 5, breed of Poodle with strange ears
Woof! Woof! Woof!
A hunter dog with name of Alfred, age of 5, breed of Butler hound
Bark! Bark! Bark!
A hunter dog with name of Roberto, age of 5, breed of Hound
Bark! Bark! Bark!

Process finished with exit code 0

```

Рисунок 1 – Тестирование.

### Выводы по работе:

Был реализован класс Собака и наследование от него.

## Практическая работа № 21

### Цель работы

Цель данной практической работы - изучить понятие интерфейса, научиться создавать интерфейсы в Java и применять их в программах.

### Теоретическое введение

Механизм наследования очень удобен, но он имеет свои ограничения. В частности мы можем наследовать только от одного класса, в отличие, например, от языка C++, где имеется множественное наследование.

В языке Java подобную проблему позволяют решить интерфейсы.

Интерфейсы определяют некоторый функционал, не имеющий конкретной реализации, который затем реализуют классы, применяющие эти интерфейсы. И один класс может применить множество интерфейсов.

Чтобы определить интерфейс, используется ключевое слово `interface`.

### Выполнения лабораторной работы

#### Задание:

Реализовать интерфейс `Priceable`, имеющий метод `getPrice()`, возвращающий некоторую цену для объекта. Проверить работу для различных классов, сущности которых могут иметь цену.

#### Решение:

Далее приведена реализация класса интерфейса `priceable` и классов реализующих его (Листинг 1).

#### Листинг 1 – код программы

```
public interface Priceable {
    int getPrice();
    public static void priceTest(){
        Candy candy = new Candy("Candy", 10);
        System.out.println(candy.toString());
        System.out.println(candy.getPrice());
        ComplicatedCandy complicatedCandy = new ComplicatedCandy(10, 10);
        System.out.println(complicatedCandy.toString());
        System.out.println(complicatedCandy.getPrice());
    }
}

public class Candy implements Priceable {
    private String name;
    private int price;

    public Candy() {
        this.name = "Candy";
        this.price = 0;
    }

    public Candy(String name, int price) {
        this.name = name;
        this.price = price;
    }
}
```

```

    }

    public String getName() {
        return this.name;
    }

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

    public int getPrice() {
        return this.price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    @Override
    public String toString() {
        return "A candy with name of " + this.name + ", price of " +
this.price;
    }
}

public class ComplicatedCandy implements Priceable{
    private int sweetness;
    private int price;
    private int weight;

    public ComplicatedCandy() {
        this.sweetness = 0;
        this.price = 0;
        this.weight = 0;
    }

    public ComplicatedCandy(int sweetness, int weight) {
        this.sweetness = sweetness;
        this.price = sweetness * weight;
        this.weight = weight;
    }

    public int getSweetness() {
        return this.sweetness;
    }

    public void setSweetness(int sweetness) {
        this.sweetness = sweetness;
    }

    public int getPrice() {
        return this.price;
    }

    public int getWeight() {
        return this.weight;
    }

    public void setWeight(int weight) {
        this.weight = weight;
    }

    @Override
    public String toString() {

```

```
        return "A candy with sweetness of " + this.sweetness + ", price of "
+ this.price + ", weight of " + this.weight;
    }
}
```

Далее приведен пример тестирования программы (рис. 1).

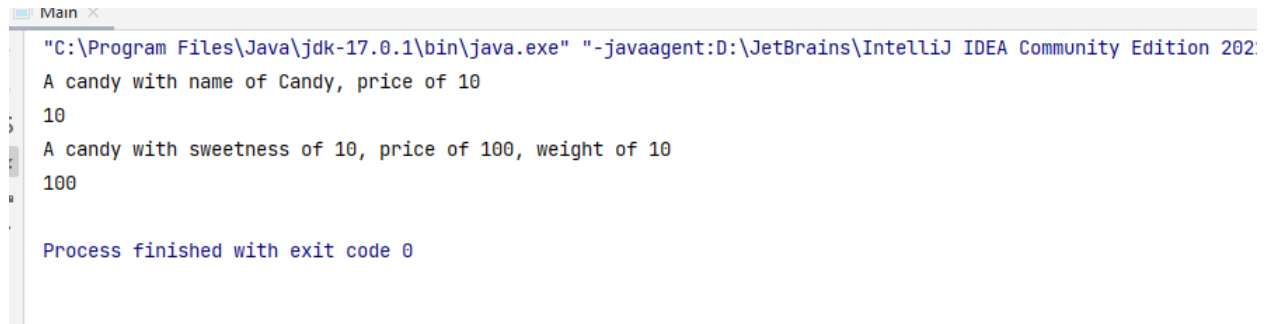
A screenshot of a Java IDE console window. The title bar shows 'Main' with a close button. The console output is as follows:  
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:D:\JetBrains\IntelliJ IDEA Community Edition 202...  
A candy with name of Candy, price of 10  
10  
A candy with sweetness of 10, price of 100, weight of 10  
100  
Process finished with exit code 0

Рисунок 1 – Тестирование.

### Выводы по работе:

Был реализован интерфейс `priceable` и классы реализовывающие его.

## Практическая работа № 22

### Цель работы

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

### Теоретическое введение

Для создания графического интерфейса пользователя можно использовать стандартную Java библиотеку Swing или AWT. В этих библиотеках имеются различные классы, позволяющие создавать окна, кнопки, текстовые поля, меню и другие объекты.

### Выполнения лабораторной работы

#### Задание:

Создать окно, нарисовать в нем 20 случайных фигур, случайного цвета. Классы фигур должны наследоваться от абстрактного класса Shape, в котором описаны свойства фигуры: цвет, позиция.

#### Решение:

Далее приведена реализация классов формирующих программу (Листинг 1).

#### Листинг 1 – код программы

```
public class Frame extends JFrame {
    public Frame() {
        super("Frame");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(null);
        setVisible(true);
        //drawing panel
        //JPanel panel = new JPanel();
        Panel2 panel = new Panel2();
        panel.setBounds(0, 0, 600, 400);
        panel.setBackground(Color.WHITE);
        add(panel);
        //drawing
        Graphics g = panel.getGraphics();
        panel.paintComponent(g);
    }

    class Panel2 extends JPanel {

        Panel2() {
            // set a preferred size for the custom panel.
            //setPreferredSize(new Dimension(420,420));
        }

        @Override
        public void paintComponent(Graphics g) {
            super.paintComponent(g);

            for(int i = 0; i < 20; i++){
```

```

        int var = (int) (Math.random() * 3);
        switch (var) {
            case 0:
                Rectangle rect = new Rectangle((int) (Math.random() *
600), (int) (Math.random() * 400), (int) (Math.random() * 50),
(int) (Math.random() * 50));
                rect.draw(g);
                break;
            case 1:
                Circle circle = new Circle((int) (Math.random() *
600), (int) (Math.random() * 400), (int) (Math.random() * 50));
                circle.draw(g);
                break;
            case 2:
                Triangle triangle = new Triangle((int) (Math.random()
* 600), (int) (Math.random() * 400), (int) (Math.random() * 50),
(int) (Math.random() * 100 - 50));
                triangle.draw(g);
                break;
        }
    }
}

abstract class Shape {
    protected Color color;
    protected int x;
    protected int y;

    public void setX(int x) {
        this.x = x;
    }

    public int getX() {
        return x;
    }

    public void setY(int y) {
        this.y = y;
    }

    public int getY() {
        return y;
    }

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

    public Color getColor() {
        return color;
    }

    public abstract void draw(Graphics g);
}

class Rectangle extends Shape {
    protected int width;
    protected int length;
    public Rectangle() {
        this.width = 1;
        this.length = 2;
        this.x = 0;
        this.y = 0;
    }
}

```

```

    public Rectangle(int x, int y, int width, int length){
        this.x = x;
        this.y = y;
        this.width = width;
        this.length = length;
        this.color = new Color((int) (Math.random()*255),
(int) (Math.random()*255), (int) (Math.random()*255));
    }
    public Rectangle(int x, int y, int width, int length, Color color){
        this.x = x;
        this.y = y;
        this.width = width;
        this.length = length;
        this.color = color;
    }
    public int getWidth(){
        return this.width;
    }
    public void setWidth(int width){
        this.width = width;
    }
    public int getLength(){
        return this.length;
    }
    public void setLength(int length){
        this.length = length;
    }

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

    public void draw(Graphics g){
        g.setColor(this.color);
        g.fillRect(this.x, this.y, this.width, this.length);
    }
}
class Circle extends Shape{
    protected int radius;
    public Circle(){
        this.radius = 1;
        this.x = 0;
        this.y = 0;
    }
    public Circle(int x, int y, int radius){
        this.x = x;
        this.y = y;
        this.radius = radius;
        this.color = new Color((int) (Math.random()*255),
(int) (Math.random()*255), (int) (Math.random()*255));
    }
    public Circle(int x, int y, int radius, Color color){
        this.x = x;
        this.y = y;
        this.radius = radius;
        this.color = color;
    }
    public int getRadius(){
        return this.radius;
    }
    public void setRadius(int radius){
        this.radius = radius;
    }
}

```

```

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

    public void draw(Graphics g){
        g.setColor(this.color);
        g.fillOval(this.x, this.y, this.radius, this.radius);
    }
}

class Triangle extends Shape{
    protected int width;
    protected int height;
    public Triangle(){
        this.width = 1;
        this.height = 1;
        this.x = 0;
        this.y = 0;
    }
    public Triangle(int x, int y, int width, int height){
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        this.color = new Color((int) (Math.random()*255),
(int) (Math.random()*255), (int) (Math.random()*255));
    }
    public Triangle(int x, int y, int width, int height, Color color){
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        this.color = color;
    }
    public int getWidth(){
        return this.width;
    }
    public void setWidth(int width){
        this.width = width;
    }
    public int getHeight(){
        return this.height;
    }
    public void setHeight(int height){
        this.height = height;
    }
    public String toString(){
        return "A shapeTask.Triangle with width=" + this.width + " and
height=" + this.height + ", which is a subclass of " + super.toString();
    }

    public void draw(Graphics g){
        g.setColor(this.color);
        g.fillPolygon(new int[]{this.x, this.x + this.width, this.x +
this.width / 2}, new int[]{this.y, this.y, this.y + this.height}, 3);
    }
}

```



Далее приведен пример тестирования программы (рис. 1).

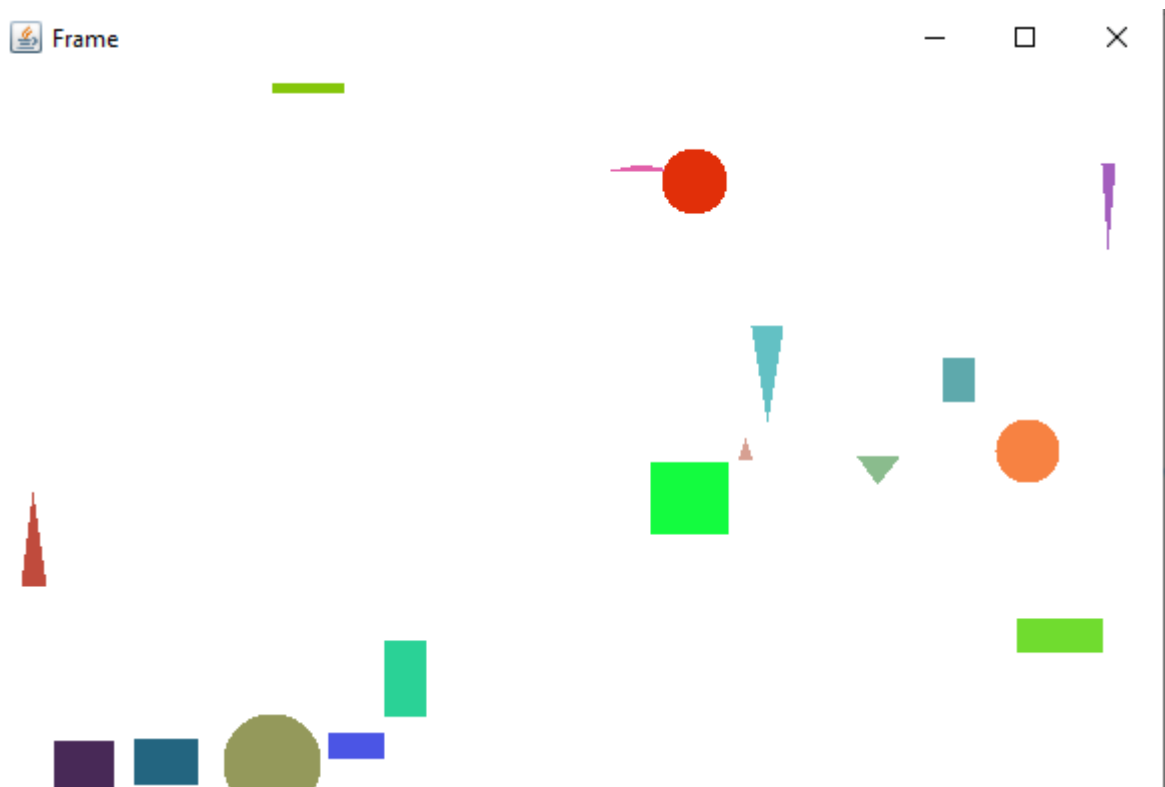


Рисунок 1 – Тестирование.

### Выводы по работе:

Была реализована программа, рисующая фигуры.

## Практическая работа № 23

### Цель работы

Цель данной практической работы - научиться обрабатывать различные события для разных компонентов (кнопок, меню и т. д.).

### Теоретическое введение

В контексте графического интерфейса пользователя наблюдаемыми объектами являются элементы управления: кнопки, флажки, меню и т.д. Они могут сообщить своим наблюдателям об определенных событиях, как элементарных (наведение мышкой, нажатие клавиши на клавиатуре), так и о высокоуровневых (изменение текста в текстовом поле, выбор нового элемента в выпадающем списке и т.д.).

Наблюдателями должны являться объекты классов, поддерживающих специальные интерфейсы (в классе наблюдателя должны быть определенные методы, о которых «знает» наблюдаемый и вызывает их при наступлении события). Такие классы в терминологии Swing называются слушателями.

### Выполнения лабораторной работы

#### Задание:

Реализуйте игру-угадайку, которая имеет одно текстовое поле и одну кнопку. Он предложит пользователю угадать число между 0-20 и дает ему три попытки. Если ему не удастся угадать, то будет выведено сообщение, что пользователь допустил ошибку в угадывании и что число меньше / больше. Если пользователь попытался три раза угадать, то программу завершается с соответствующим сообщением. Если пользователь угадал, то программа должна тоже завершаться с соответствующим сообщением.

#### Решение:

Далее приведена реализация игры-угадалки (Листинг 1).

#### Листинг 1 – код программы

```
public final class GuessGame extends JFrame {
    private int number = (int) (Math.random() * 21);
    private int turn = 0;

    public GuessGame() {
        super("Guess Game");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        JTextArea textArea = new JTextArea();
        add(textArea, BorderLayout.CENTER);
        JButton button = new JButton("Guess");
        add(button, BorderLayout.SOUTH);
        JLabel label = new JLabel("Guess the number from 0 to 20");
        add(label, BorderLayout.NORTH);
        setVisible(true);
        button.addActionListener(e -> {
            String input = textArea.getText();
            try{
                int guess = Integer.parseInt(input);
```

```

        if (guess == number) {
            JOptionPane.showMessageDialog(null, "You guessed the
number!");
            System.exit(0);
        }
        else if (guess > number) {
            label.setText("Guess the number from 0 to 20      " + "Your
guess is too high");
        }
        else {
            label.setText("Guess the number from 0 to 20      " + "Your
guess is too low");
        }
        turn++;
        if (turn == 3) {
            JOptionPane.showMessageDialog(null, "You lost!");
            System.exit(0);
        }
    }
    catch (NumberFormatException ex) {
        label.setText("Guess the number from 0 to 20\n" + "You entered
not a number");
    }
    });
}
}

```

Далее приведен пример тестирования программы (рис. 1).

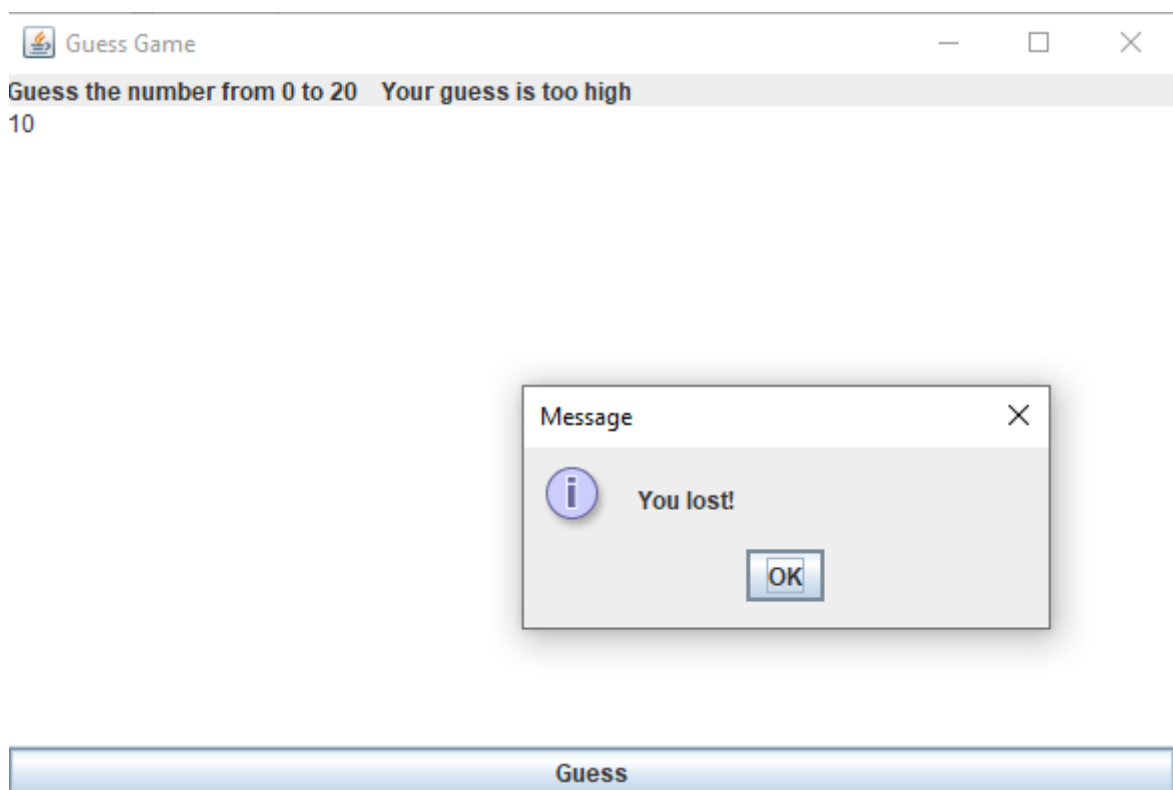


Рисунок 1 – Тестирование.

**Выводы по работе:**

Была реализована игра-угадалка.

## Практическая работа № 24

### Цель работы

Целью данной практической работы является изучение работы с различными коллекциями в Java.

### Теоретическое введение

Для хранения наборов данных в Java предназначены массивы. Однако их не всегда удобно использовать, прежде всего потому, что они имеют фиксированную длину. Эту проблему в Java решают коллекции. Однако суть не только в гибких по размеру наборах объектов, но и в том, что классы коллекций реализуют различные алгоритмы и структуры данных, например, такие как стек, очередь, дерево и ряд других.

### Выполнения лабораторной работы

#### Задание:

Протестировать работу коллекции ArrayList.

Протестировать работу коллекции LinkedList.

#### Решение:

Далее приведена реализация программы (Листинг 1).

#### Листинг 1 – код программы

```
public class ArrayListTester {  
    public static void main(String[] args) {  
        ArrayList<String> list = new ArrayList<String>();  
        list.add("Hello");  
        list.add("World");  
        list.add("!");  
        System.out.println(list);  
        list.remove(1);  
        System.out.println(list);  
        list.add(1, "Strong");  
        System.out.println(list);  
        list.set(1, "Earth");  
        System.out.println(list);  
        System.out.println(list.get(1));  
        System.out.println(list.size());  
        System.out.println(list.indexOf("Earth"));  
        System.out.println(list.indexOf("Strong"));  
        System.out.println(list.contains("Earth"));  
        System.out.println(list.contains("Strong"));  
        System.out.println(list.isEmpty());  
        list.clear();  
        System.out.println(list.isEmpty());  
  
        ArrayList<Integer> list2 = new ArrayList<Integer>(5);  
        list2.add(1);  
        list2.add(2);  
        list2.add(3);  
        list2.add(4);  
        list2.add(5);  
        System.out.println(list2);  
        ArrayList list3 = new ArrayList(list2);
```

```

        list3.add(6);
        list3.add(7);
        System.out.println(list3);
        list3.addAll(list2);
        System.out.println(list3);
        System.out.println(list3.subList(1, 4)); //will be index 1, 2, 3
    }
}

public class LinkedListTester {
    public static void main(String[] args) {
        LinkedList<String> list = new LinkedList<String>();
        list.add("Hello");
        list.add("World");
        list.add("!");
        System.out.println(list);
        list.remove(1);
        System.out.println(list);
        list.add(1, "Strong");
        System.out.println(list);
        list.set(1, "Earth");
        System.out.println(list);
        System.out.println(list.get(1));
        System.out.println(list.size());
        System.out.println(list.indexOf("Earth"));
        System.out.println(list.indexOf("Strong"));
        System.out.println(list.contains("Earth"));
        System.out.println(list.contains("Strong"));
        System.out.println(list.isEmpty());
        list.clear();
        System.out.println(list.isEmpty());

        LinkedList<Integer> list2 = new LinkedList<Integer>();
        list2.add(1);
        list2.add(2);
        list2.add(3);
        list2.add(4);
        list2.add(5);
        System.out.println(list2);
        LinkedList list3 = new LinkedList(list2);
        list3.add(6);
        list3.add(7);
        System.out.println(list3);
        list3.addAll(list2);
        System.out.println(list3);
        System.out.println(list3.subList(1, 4)); //will be index 1, 2, 3

        LinkedList<Integer> list4 = new LinkedList<Integer>();
        list4.addFirst(10);
        list4.addFirst(20);
        list4.addLast(30);
        list4.addLast(40);
        System.out.println(list4);
        System.out.println(list4.getFirst());
        System.out.println(list4.getLast());
        System.out.println(list4);
        System.out.println(list4.removeFirst());
        System.out.println(list4.pollLast());
        System.out.println(list4);
    }
}

```

Далее приведены примеры тестирования программы (рис. 1, 2).

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:D:\JetBrains\In
[Hello, World, !]
[Hello, !]
[Hello, Strong, !]
[Hello, Earth, !]
Earth
3
1
-1
true
false
false
true
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, 6, 7]
[1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5]
[2, 3, 4]

Process finished with exit code 0
```

Рисунок 1 – Тестирование.

```
LinkedListTester ×
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:D:\JetBrains\IntelliJ IDEA Community Edition 2021.2.3\lib\ic
[Hello, World, !]
[Hello, !]
[Hello, Strong, !]
[Hello, Earth, !]
Earth
3
1
-1
true
false
false
true
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, 6, 7]
[1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5]
[2, 3, 4]
[20, 10, 30, 40]
20
40
[20, 10, 30, 40]
20
40
[10, 30]

Process finished with exit code 0
```

Рисунок 2 – Тестирование.

### Выводы по работе:

Было проведено тестирование классов ArrayList и LinkedList.



## Практическая работа № 25

### Цель работы

Освоить на практике работу с файлами на языке Java. Получить практические навыки по чтению и записи данных в файл.

### Теоретическое введение

Класс `FileWriter` является производным от класса `Writer`. Он используется для записи текстовых файлов.

Класс `FileReader` наследуется от абстрактного класса `Reader` и предоставляет функциональность для чтения текстовых файлов.

### Выполнения лабораторной работы

#### Задание:

Реализовать запись в файл введенной с клавиатуры информации.

Реализовать вывод информации из файла на экран.

Заменить информацию в файле на информацию, введенную с клавиатуры.

Добавить в конец исходного файла текст, введенный с клавиатуры.

#### Решение:

Далее приведена реализация программы (Листинг 1).

#### Листинг 1 – код программы

```
public class FileWorker {
    public static void main(String[] args) throws FileNotFoundException {
        //well I thought why not make this thing able to throw exception instead of
        //writing try-catch block
        File file = new File("src/Prac25/file.txt");

        Scanner in = new Scanner(System.in);
        try{
            Scanner fin = new Scanner(file);
        }
        catch(FileNotFoundException e){
            System.out.println("file not found, creating new one");
            try{
                file.createNewFile();
            }
            catch(Exception e1){
                System.out.println("something went wrong");
            }
        }

        System.out.println("insert your text here");
        String text = in.nextLine();
        System.out.println("do you want to append or rewrite file? (a/r)
        (default is rewrite)");
        String answer = in.nextLine();
        if (answer.equals("a")) {
            try {
                FileWriter writer = new FileWriter(file, true);
                writer.write((text + "\n")); //what the fuck, for it to actually
                concatenate strings i need to add additional () around them
            }
            catch (IOException e) {
                System.out.println("error writing to file");
            }
        }
    }
}
```

```

        writer.close();
    } catch (Exception e) {
        System.out.println("something went wrong");
    }
} else {
    try {
        FileWriter writer = new FileWriter(file);
        writer.write((text + "\n"));
        writer.close();
    } catch (Exception e) {
        System.out.println("something went wrong");
    }
}
System.out.println("your file contains:");
try{
    Scanner fin = new Scanner(file);
    while(fin.hasNextLine()){
        System.out.println(fin.nextLine());
    }
}
catch(FileNotFoundException e){
    System.out.println("file not found");
}
}
}

```

Далее приведены примеры тестирования программы (рис. 1, 2).

```

"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:D:\JetBrains\IntelliJ
insert your text here
some text that just came up to my mind
do you want to append or rewrite file? (a/r)    (default is rewrite)
r
your file contains:
some text that just came up to my mind

Process finished with exit code 0
|

```

Рисунок 1 – Тестирование.

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:D:\JetBrains\IntelliJ
insert your text here
some other text that i will append
do you want to append or rewrite file? (a/r) (default is rewrite)
a
your file contains:
some text that just came up to my mind
some other text that i will append

Process finished with exit code 0
```

Рисунок 2 – Тестирование.

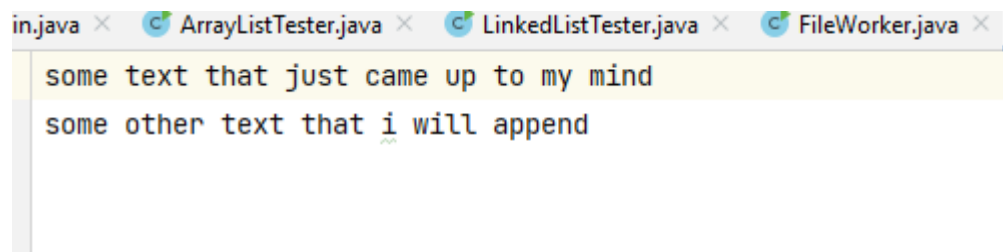


Рисунок 3 – Тестирование.

### Выводы по работе:

Была реализована программа работающая с файлом.

**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. Фрэйн 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](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.*