



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 ИМБО-02-22

Лищенко Т. В.

Принял старший преподаватель

Рачков А.В.

Практические работы работа выполнены «__»_____2023г.

«Зачтено» «__»_____2023г.

Москва 2023

Оглавление

<i>Практическая работа №1</i>	6
Цель работы:.....	6
Теоретическое введение	7
Задача №3 и №4	8
Условие задачи:	8
Решение:	9
Результат выполнения программы:	11
Задача №5	12
Условие задачи:	12
Решение:	13
Результат выполнения программы:	14
Задача №6	15
Условие задачи	15
Решение:	16
Результат выполнения программы:	17
Задача №7	18
Условие задачи:	18
Решение:	19
Результат выполнения программы:	20
Вывод:	21
<i>Практическая работа №2</i>	22
Цель работы:.....	22
Теоретическое введение	23
Задача №1	24
Условие:.....	24
Решение:	25
Класс Author.....	25
Класс TestAuthor.....	27
Результат выполнения программы:	28
Задача №2	29
Условие задачи:	29
Решение:	30
Результат выполнения программы:	31
Задача №5	32
Условие задачи:	32
Решение:	33
Результат выполнения программы:	36
Вывод:	37

<i>Практическая работа №3</i>	38
Цель работы:	38
Теоретическое введение	39
Задача №1	40
Условие задачи:	40
Решение:	41
Результат выполнения программы:	42
Вывод	43
<i>Практическая работа №4</i>	44
Цель работы:	44
Теоретическое введение	45
Задача №1	46
Условие задачи:	46
Решение:	47
Результат выполнения программы:	48
Задача №1 Блок 2	49
Условие задачи:	49
Решение:	50
Результат выполнения программы:	55
Вывод	56
<i>Практическая работа №5</i>	57
Цель работы:	57
Теоретическое введение	58
Задание №1	59
Условие задачи:	59
Решение:	60
Результат выполнения программы:	62
Вывод	63
<i>Практическая работа №6</i>	64
Цель работы:	64
Теоретическое введение	65
Задание №1	67
Условие задачи:	67
Решение:	68
Результат выполнения программы:	70
Вывод	71
<i>Практическая работа №7</i>	72
Цель работы:	72

Теоретическое введение	73
Задание №1	75
Условие задачи:	75
Решение:	76
Результат выполнения программы:	79
Вывод	80
<i>Практическая работа №8</i>	<i>81</i>
Цель работы:	81
Теоретическое введение	82
Задание №1	83
Условие задачи:	83
Решение:	84
Результат выполнения программы:	85
Задание №16	86
Условие задачи:	86
Решение:	87
Результат выполнения программы:	88
Задание №17	89
Условие задачи:	89
Решение:	90
Результат выполнения программы:	91
Вывод	92
<i>Практическая работа №9</i>	<i>93</i>
Цель работы:	93
Теоретическое введение	94
Задание №1	96
Условие задачи:	96
Решение:	97
Результат выполнения программы:	99
Вывод	100
<i>Практическая работа №10</i>	<i>101</i>
Цель работы:	101
Теоретическое введение	102
Задание №1-3	103
Условие задачи:	103
Решение:	104
Результат выполнения программы:	110
Вывод	111
<i>Используемая литература</i>	<i>119</i>

Практическая работа №1

Цель работы:

Познакомиться со средой разработки IntelliJ IDE, реализовать простейшие задачи на языке JAVA.

Теоретическое введение

Язык Джава— это объектно-ориентированный язык программирования, с со строгой инкапсуляцией и типизацией. Программы, написанные на языке, Джава могут выполняться под управлением различных операционныхсистемах при наличии необходимого ПО – Java Runtime Environment. Для того чтобы создать и запускать программы на языке.

Джава необходимо следующее ПО:

- Java Development Kit (JDK);
- Java Runtime Environment (JRE);
- Среда разработки. Например, IDE IntelliJ IDEA или NetBeans

Чтобы начать написание программы необходимо запустить среду разработки. При первом запуске среды обычно нужно указать путь к JDK, чтобы можно было компилировать код и запускать программу. В среде разработки необходимо создать Джава проект, после чего необходимо создать пакет и в нем создать какой-либо класс. По правилам именования пакетов используются только строчные буквы, в названии класса первая буква имени должна быть заглавная. Также в свойствах проекта нужно указать класс, с которого будет начинаться запуск программы.

Задача №3 и №4

Условие задачи:

Написать программу, в результате которой массив чисел создается с помощью инициализации (как в Си) вводится и считается в цикле сумма элементов целочисленного массива, а также среднее арифметическое его элементов результат выводится на экран. Использовать цикл for.

Написать программу, в результате которой массив чисел вводится пользователем с клавиатуры считается сумма элементов целочисленного массива с помощью циклов do while, while, также необходимо найти максимальный и минимальный элемент в массиве, результат выводится на экран.

Решение:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in); // Объявляем Scanner
        System.out.println("Enter array length: ");

        int size = input.nextInt(); // Читаем с клавиатуры размер массива и
записываем в size
        int array[] = new int[size]; // Создаём массив int размером в size

        System.out.println("Insert array elements:");

        /*Пройдёмся по всему массиву, заполняя его*/
        for (int i = 0; i < size; i++)
            array[i] = input.nextInt(); // Заполняем массив элементами,
введёнными с клавиатуры

        System.out.print ("Inserted array elements:");

        for (int i = 0; i < size; i++) {
            System.out.print (" " + array[i]); // Выводим на экран, полученный
массив
        }

        System.out.println();

        int sum = 0;
        for (int i = 0; i < size; i++){
            sum += array[i];
        }

        double average;
        average = (double) sum / array.length;

        System.out.println("Average: " + average);
        System.out.println("Sum: " + sum);

        int j = 0;
        int sum_while = 0;

        do{
```

```

        sum_while += array[j];
        j++;
    } while(j!=size);

    System.out.println("Sum with do while" + sum_while);

    int max = getMax(array);
    System.out.println("Maximum Value is: " + max);

    int min = getMin(array);
    System.out.println("Minimum Value is: " + min);
}
// Здесь находим максимум
public static int getMax(int[] inputArray){
    int maxValue = inputArray[0];
    for(int i=1; i < inputArray.length; i++){
        if(inputArray[i] > maxValue){
            maxValue = inputArray[i];
        }
    }

    return maxValue;
}
// здесь находим минимум
public static int getMin(int[] inputArray){
    int minValue = inputArray[0];
    for(int i=1;i<inputArray.length;i++){
        if(inputArray[i] < minValue){
            minValue = inputArray[i];
        }
    }
    return minValue;
}
}

```

Результат выполнения программы:

```
Enter array length:
3
Insert array elements:
4
5
11
Inserted array elements: 4 5 11
Average: 6.666666666666667
Sum: 20
Sum with do while20
Maximum Value is: 11
Minimum Value is: 4
```

Задача №5

Условие задачи:

Написать программу, в результате которой выводятся на экран аргументы командной строки в цикле for.

Решение:

```
public class Main {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 5; i++) {  
            System.out.println("i = " + i);  
        }  
    }  
}
```

Результат выполнения программы:

```
i = 1  
i = 2  
i = 3  
i = 4  
i = 5
```

Задача №6

Условие задачи

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

Решение:

```
public class Main {  
    public static void main(String[] args) {  
        int num = 10; // number of values we want in a series  
        System.out.println("Гармонический ряд: ");  
        for (int i = 1; i < num; i++){  
            System.out.println("1 / " + i);  
        }  
    }  
}
```


Результат выполнения программы:

Гармонический ряд:

1 / 1

1 / 2

1 / 3

1 / 4

1 / 5

1 / 6

1 / 7

1 / 8

1 / 9

Задача №7

Условие задачи:

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

Решение:

```
import java.math.BigInteger;
import java.util.Scanner;
class factoriall{
    public static BigInteger factorialmath(int n) {
        BigInteger result = BigInteger.ONE;
        for (int i = 2; i <= n; i++)
            result = result.multiply(BigInteger.valueOf(i));
        return result;
    }
}
public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        System.out.println("fact: " + factoriall.factorialmath(number));
    }
}
```

Результат выполнения программы:

```
5  
fact: 120
```

```
10000  
fact: 2846259680917054518906413212119868
```

Вывод:

Познакомились со средой разработки IntelliJ IDE, научились решать простейшие задачи на языке программирования JAVA.

Практическая работа №2

Цель работы:

Реализовать классы объектов, их сеттеры, геттеры.

Теоретическое введение

Для начала разберем, что такое модификаторы доступа в Java. В Java существуют следующие модификаторы доступа:

- `private`: данные класса доступны только внутри класса;
- `protected`: данные класса доступны внутри пакета и в наследниках;
- `public`: данные класса доступны всем.

В качестве примера рассмотрим программу, описывающую окружность. Создадим класс `Circle`. Каждая окружность имеет несколько параметров:

- Координаты по оси `x`;
- Координаты по оси `y`;
- Радиус `r`;
- Цвет `colour`.

Объявим переменные класса:

```
private double x;  
private double y;  
private double r;  
private String colour;
```

Задача №1

Условие:

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 2.4.

Решение:

Класс Author

```
public class Author {
    private String name;
    private String email;
    private char gender;

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

    public String getName(){
        return name;
    }

    public String getEmail(){
        return email;
    }

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

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

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

    public char getGender() {
        return gender;
    }

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

$\}$
$\}$

Класс TestAuthor

```
import java.util.Objects;
import java.util.Scanner;

public class TestAuthor {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        Author a1 = new Author("", "", '-');
        System.out.println(a1.toString());

        System.out.println("Вы хотите добавить нового пользователя?\nВведите
Y - да, N - нет");
        String check = in.nextLine();

        if (!Objects.equals(check, "Y")) {
            System.out.println("Ладно, тогда до скорой встречи");
        } else {
            System.out.println("Хорошо, давайте введем данные");

            System.out.println("Введите ФИО:");
            String name = in.nextLine();
            a1.setName(name);

            System.out.println("Введите почту:");
            String email = in.nextLine();
            a1.setEmail(email);

            System.out.println("Введите гендер (даже если вы вертолет:");
            char gender = in.next().charAt(0);
            a1.setGender(gender);
        }
        System.out.println("Имя: "+a1.getName());
        System.out.println("Почта: "+a1.getEmail());
        System.out.println("Гендер: "+a1.getGender());
    }
}
```

Результат выполнения программы:

```
Коспот {name=" ", email=" ", gender=" "}  
Вы хотите добавить нового пользователя?  
Введите Y - да, N - нет  
Y  
Хорошо, давайте введем данные)  
Введите ФИО:  
Лищенко Тимофей Викторович  
Введите почту:  
timmmofey@itachi-uchiha.ru  
Введите гендер (даже если вы вертолет):  
ВЕРТОЛЕТИЩЕ  
Имя: Лищенко Тимофей Викторович  
Почта: timmmofey@itachi-uchiha.ru  
Гендер: В
```

Задача №2

Условие задачи:

По UML диаграмме класса, представленной на рис. 2.5 написать программу, которая состоит из двух классов. Один из них Ball должен реализовывать сущность мяч, а другой с названием TestBall тестировать работу созданного класса. Класс Ball должен содержать реализацию методов, представленных на UML. Диаграмма на рисунке описывает сущность Мяч написать программу. Класс Ball моделирует движущийся мяч.

Решение:

```
public class Main {  
  
    public static void main(String[] args) {  
  
        // Ball and book classes test  
  
        Ball ball = new Ball(4);  
  
        ball.display_info();  
  
    }  
  
}  
  
class Ball {  
  
    int radius;  
  
    int diameter;  
  
  
    Ball(int radius) {  
  
        this.radius = radius;  
  
        this.diameter = this.radius * 2;  
  
        System.out.println("\nBall object was created");  
  
    }  
  
  
    void display_info() {  
  
        System.out.printf("Radius: %s \tDiameter: %d\n", this.radius, this.diameter);  
  
    }  
  
}
```

Результат выполнения программы:

```
Ball object was created  
Radius: 4   Diameter: 8
```

Задача №5

Условие задачи:

Разработайте и реализуйте класс Dog (Собака), поля класса описывают кличку и возраст собаки. Необходимо выполнить следующие действия: определить конструктор собаки, чтобы принять и инициализировать данные экземпляра., включить стандартные методы (аксессоры) для получения и установки для имени и возраста, включить метод для перевода возраста собаки в “человеческий” возраст (возраст семь раз собаки), включите метод ToString, который возвращает описание экземпляра собаки в виде строки. Создание класса тестера под названием ПитомникСобак, реализует массив собак и основной метод этого класса позволяет добавить в него несколько объектов собаки.

Решение:

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

public class Main {
    public static void main(String[] args) {
        // Dog class test
        Dog dog_1 = new Dog("Dog_1", 5);
        dog_1.set_age(6);
        System.out.println(dog_1);

        Dog dogs[] = {new Dog("Dog_2", 9), new Dog("Dog_3", 10), new
Dog("Dog_4", 11)};

        // Dog kennel test
        Dog_kennel dog_kennel = new Dog_kennel();
        System.out.println(dog_kennel);

        dog_kennel.add_dog(dog_1);
        System.out.println(dog_kennel);

        dog_kennel.add_dogs(dogs);
        System.out.println(dog_kennel);
    }
}

class Dog {
    private String name;
    private int age;

    // Init
    Dog(String name, Integer age) {
        this.name = name;
        this.age = age;
        System.out.println("\nDog object with name: " + this.name + " was
created");
    }

    // Age setter
    public void set_age(int age) {
        if (age > 0 && age < 100) {
            this.age = age;
            System.out.println("Dog age setted as: " + this.age);
        }
    }
}
```

```

    }
}

// Age getter
public int get_age() {
    return this.age;
}

// Name setter
public void set_name(String name) {
    this.name = name;
    System.out.println("Dog name setted as: " + this.name);
}

// Name getter
public String get_name() {
    return this.name;
}

// Dog age as human age getter
public int get_human_age() {
    return this.age * 7;
}

public String toString() {
    return "Dog " + this.name + " with age " + this.age;
}
}

class Dog_kennel {
    private List<Dog> dog_kennel_array = new ArrayList<Dog>();

    {
        System.out.println("\nDog kennel object was created");
    }

    // Add one dog
    public void add_dog(Dog dog) {
        this.dog_kennel_array.add(dog);
        System.out.println(dog + ". This dog added into the dog kennel
successfully");
    }

    // Add many dogs

```

```
public void add_dogs(Dog dogs[]) {  
    this.dog_kennel_array.addAll(new ArrayList<Dog>(Arrays.asList(dogs)));  
    System.out.println("Dogs: " + Arrays.toString(dogs) + " added into the dog  
kennel successfully");  
}  
  
public String toString() {  
    if (!this.dog_kennel_array.isEmpty()) return "Dog kennel: " +  
this.dog_kennel_array;  
    else return "Dog kennel empty!";  
}  
}
```

Результат выполнения программы:

```
Dog object with name: Dog_1 was created
Dog age setted as: 6
Dog Dog_1 with age 6

Dog object with name: Dog_2 was created

Dog object with name: Dog_3 was created

Dog object with name: Dog_4 was created

Dog kennel object was created
Dog kennel empty!
Dog Dog_1 with age 6. This dog added into the dog kennel successfully
Dog kennel: [Dog Dog_1 with age 6]
Dogs: [Dog Dog_2 with age 9, Dog Dog_3 with age 10, Dog Dog_4 with age 11] added into the dog kennel successfully
Dog kennel: [Dog Dog_1 with age 6, Dog Dog_2 with age 9, Dog Dog_3 with age 10, Dog Dog_4 with age 11]
```

Вывод:

Научились работать с классами в Java.

Практическая работа №3

Цель работы:

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

Теоретическое введение

Класс Math Класс Java Math предоставляет ряд методов для работы выполнения математических вычислений, например таких как `min()`, `max()`, `sqrt()`, `pow()`, `sin()`, `cos()`, `tan()`, `round()`, `abs()` так далее. В классе также есть константа число `PI`. Все методы класса публичные и статические, поэтому вы можете вызывать их, обратившись напрямую через точку к классу, не создавая объект типа класс. Если размер равен `int` или `long` и результаты выходят за пределы диапазона значений, методы `addExact()`, `subtractExact()`, `multiplyExact()` и `toIntExact()` вызывают исключение `ArithmeticException`.

Для других арифметических операций, таких как увеличение, уменьшение, деление, абсолютное значение и отрицание, переполнение происходит только с определенным минимальным или максимальным значением. При необходимости его следует сравнивать с максимальным и минимальным значением. Класс Java Math имеет множество методов, которые позволяют решать на Джава математические задачи.

Задача №1

Условие задачи:

Создать массив вещественных чисел случайным образом, вывести его на экран, отсортировать его, и снова вывести на экран (использовать два подхода к генерации случайных чисел – метод `random()` класса `Math` и класс `Random`).

Решение:

```
import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        Random rand = new Random();
        System.out.println("Введите размер массива: ");
        int size = in.nextInt();
        while (size <= 0) {
            System.out.println("Размер массива не может быть <= 0. Попробуйте снова: ");
            size = in.nextInt();
        }
        double[] arr = new double[size];
        for (int i = 0; i < size; i++) {
            arr[i] = rand.nextDouble();
        }
        System.out.println("Массив созданный классом Random: " + Arrays.toString(arr));
        Arrays.sort(arr);
        System.out.println("Отсортированный массив созданный классом Random: " + Arrays.toString(arr));

        for (int i = 0; i < size; i++) {
            arr[i] = Math.random();
        }
        System.out.println("Массив созданный методом random: " + Arrays.toString(arr));
        Arrays.sort(arr);
        System.out.println("Отсортированный массив созданный методом random: " + Arrays.toString(arr));
    }
}
```

Результат выполнения программы:

```
Введите размер массива:
5
Массив созданный классом Random: [0.45724022790574503, 0.18954382301998696, 0.751505765432021, 0.8530402909355606, 0.6842043401877567]
Отсортированный массив созданный классом Random: [0.18954382301998696, 0.45724022790574503, 0.6842043401877567, 0.751505765432021, 0.8530402909355606]
Массив созданный методом random: [0.5219682273253652, 0.8982640400278387, 0.5725358471773752, 0.9559478811280876, 0.8752417812842814]
Отсортированный массив созданный методом random: [0.5219682273253652, 0.5725358471773752, 0.8752417812842814, 0.8982640400278387, 0.9559478811280876]
```

Вывод

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

Практическая работа №4

Цель работы:

Познакомиться с новым ссылочным типом данных перечислением, научиться разрабатывать перечисления и использовать их в своих программах.

Теоретическое введение

Перечисления это один из объектных типов в Джава. Они являются безопасными типами, поскольку переменная тип перечисление может принимать значение только константу из перечисления. Рассмотрим пример объявления перечисления в языке Джва: `public enum Level { HIGH, MEDIUM, LOW }` Обратите внимание на `enum` - ключевое слово перед именем перечисления, которое используется вместо `class` или `interface`. Ключевое слово `enum` сигнализирует компилятору Java, что это определение типа является перечислением. Вы можете ссылаться на константы в приведенном выше перечислении следующим образом: `Level level = Level.HIGH;` В этом случае переменная `level` принимает значение `HIGH`. Обратите внимание, что `level` переменная имеет тип, `Level` который является типом перечисления Java, определенным в приведенном выше примере. `Level`. Переменная может принимать одно значение из констант перечисления `Level`, то есть в качестве значения может принимать значения `HIGH`, `MEDIUM`, или `LOW`.

Задача №1

Условие задачи:

Создать перечисление, содержащее названия времен года.

- 1) Создать переменную, содержащую ваше любимое время года и распечатать всю информацию о нем.
- 2) Создать метод, который принимает на вход переменную созданного вами enum типа. Если значение равно Лето, выводим на консоль “Я люблю лето” и так далее. Используем оператор switch.
- 3) Перечисление должно содержать переменную, содержащую среднюю температуру в каждом времени года.
- 4) Добавить конструктор, принимающий на вход среднюю температуру.
- 5) Создать метод getDescription, возвращающий строку “Холодное время года”. Переопределить метод getDescription - для константы Лето метод должен возвращать “Теплое время года”.
- 6) В цикле распечатать все времена года, среднюю температуру и описание времени года.

Решение:

```
enum Season {
    WINTER(0),
    SPRING(10),
    SUMMER(20),
    AUTUMN(5);
    private int avgtemp;

    Season(int avgtemp) {
        this.avgtemp = avgtemp;
    }

    public String getInfo() {
        switch (this) {
            case AUTUMN:
                return "Осень есть осень";
            case WINTER:
                return "Зима есть зима";
            case SUMMER:
                return "Лето есть лето";
            case SPRING:
                return "Весна есть весна";
            default:
                return "ты дурак?";
        }
    }

    public int getAvgtemp() {
        return this.avgtemp;
    }
}

public class Main {
    public static void main(String[] args) {
        System.out.println("Да все времена хороши вы че?!");

        for (Season season : Season.values()) {
            System.out.println("Время года: " + season);
            System.out.println("Средняя температура: " + season.getAvgtemp() + "°C");
            System.out.println("Описание: " + season.getInfo());
            System.out.println();
        }
    }
}
```

Результат выполнения программы:

Да все времена хороши вы че?!

Время года: WINTER

Средняя температура: 0°C

Описание: Зима есть зима

Время года: SPRING

Средняя температура: 10°C

Описание: Весна есть весна

Время года: SUMMER

Средняя температура: 20°C

Описание: Лето есть лето

Время года: AUTUMN

Средняя температура: 5°C

Описание: Осень есть осень

Задача №1 Блок 2

Условие задачи:

Необходимо реализовать простейший класс Shape (Фигура). Добавьте метод класса `getType()` (тип фигуры, возвращает строку тип String название фигуры). С помощью наследования создайте дочерние классы Circle, Rectangle и Square. (из предыдущей практической работы). Также реализуйте во всех классах методы `getArea()` (возвращает площадь фигуры), `getPerimeter()` (возвращает периметр фигуры). Переопределите в дочерних классах методы класса родителя `toString()`, `getArea()`, `getPerimeter()` и `getType()`. Создать класс-тестер для вывода информации об объекте и продемонстрировать вызов методов используя родительскую ссылку. Объяснить работу программы.

Решение:

/* Задания на практическую работу № 4.1
Задания на абстрактные классы
Перепишите суперкласс Shape из задания 1, сделайте его
абстрактным и наследуйте подклассы, так как это представлено на UML
диаграмме на рис. 4.1.1 Circle, Rectangle и Square. */

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        // Circle class test
        Circle circle = new Circle(0, "black", true);
        System.out.println("Enter radius: ");
        int radius_circle = in.nextInt();
        circle.set_radius(radius_circle);
        System.out.println("Enter color: ");
        String color_circle = in.nextLine();
        circle.set_color(color_circle);
        System.out.println(circle);

        // Rectangle class test
        Rectangle rectangle = new Rectangle();
        System.out.println("Enter length: ");
        int length_rect = in.nextInt();
        rectangle.set_length(length_rect);
        System.out.println("Enter width: ");
        int width_rect = in.nextInt();
        rectangle.set_width(width_rect);
        System.out.println("Rectangle area: " + rectangle.get_area());
        System.out.println(rectangle);

        // Square class test
        Square square = new Square(10, "white", false);
        square.set_side(11);
        System.out.println(square);
    }
}

abstract class Shape {
    private String color;
    private boolean filled;

    Shape() {
        System.out.println("\nShape object was created");
    }

    // Color getter
```

```

String get_color() {
    return this.color;
}

// Color setter
void set_color(String color) {
    this.color = color;
    System.out.println("Shape color setted as: " + this.color);
}

// Filled getter
boolean is_filled() {
    return this.filled;
}

// Filled getter
void set_filled(boolean filled) {
    this.filled = filled;
    System.out.println("Shape filledness setted as: " + this.filled);
}

// Area getter
double get_area() {
    return 0.0;
}

// Perimetr getter
double get_perimeter() {
    return 0.0;
}

public String toString() {
    return "Shape object: is filled: " + this.filled + ", color: " + this.color;
}
}

class Circle extends Shape {
    protected double radius;

    Circle(double radius, String color, boolean filled) {
        super.set_color(color);
        super.set_filled(filled);
        this.radius = radius;
        System.out.println("Circle object was created");
    }

    // Radius getter
    double get_radius() {
        return this.radius;
    }

    // Radius setter

```

```

void set_radius(double radius) {
    if (radius > 0) {
        this.radius = radius;
        System.out.println("Circle radius setted as: " + this.radius);
    } else {
        System.out.println("Circle radius must be > 0");
    }
}

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

@Override
double get_perimeter() {
    return 2 * Math.PI * radius;
}

@Override
public String toString() {
    System.out.println(super.toString());
    return "Shape: circle, radius: " + this.radius;
}
}

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

    Rectangle() {
        super.set_color("blue");
        super.set_filled(false);
        this.width = get_width();
        this.length = get_length();
        System.out.println("Rectangle object was created");
    }

    // Width getter
    double get_width() {
        return this.width;
    }

    // Width setter
    void set_width(double width) {
        if (width > 0) {
            this.width = width;
            System.out.println("Rectangle width setted as: " + this.width);
        } else {
            System.out.println("Rectangle width must be > 0");
        }
    }
}

```

```

    }

    // Length getter
    double get_length() {
        return this.length;
    }

    // Width setter
    void set_length(double length) {
        if (length > 0) {
            this.length = length;
            System.out.println("Rectangle length setted as: " + this.length);
        } else {
            System.out.println("Rectangle length must be > 0");
        }
    }

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

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

    @Override
    public String toString() {
        System.out.println(super.toString());
        return "Shape: rectangle, length: " + this.length + ", width: " + this.width;
    }
}

class Square extends Rectangle {

    Square(double side, String color, boolean filled) {
        super.set_color("blue");
        super.set_filled(false);
        this.width = side;
        this.length = side;
        System.out.println("[+] Square object was created");
    }

    // Side getter
    double get_side() {
        return this.width;
    }

    // Side setter
    void set_side(double side) {

```

```
    if (side > 0) {
        this.width = side;
        this.length = side;
        System.out.println("Square side setted as: " + this.width);
    } else {
        System.out.println("Square side must be > 0");
    }
}

@Override
public String toString() {
    System.out.println(super.toString());
    return "Shape: square, side: " + this.width;
}
}
```

Результат выполнения программы:

```
Shape object was created
Shape color setted as: black
Shape filledness setted as: true
Circle object was created
Enter radius:
10
Circle radius setted as: 10.0
Enter color:
Shape color setted as:
Shape object: is filled: true, color:
Shape: circle, radius: 10.0

Shape object was created
Shape color setted as: blue
Shape filledness setted as: false
Rectangle object was created
Enter length:
10
Rectangle length setted as: 10.0
Enter width:
5
Rectangle width setted as: 5.0
Rectanle area: 50.0
Shape object: is filled: false, color: blue
Shape: rectangle, length: 10.0, width: 5.0

Shape object was created
Shape color setted as: blue
Shape filledness setted as: false
Rectangle object was created
Shape color setted as: blue
Shape filledness setted as: false
[+] Square object was created
Square side setted as: 11.0
Shape object: is filled: false, color: blue
Shape: rectangle, length: 11.0, width: 11.0
Shape: square, side: 11.0
```

Вывод

Познакомились с новым ссылочным типом данных перечислением, научиться разрабатывать перечисления и использовать их в своих программах, а также с абстрактными классами.

Практическая работа №5

Цель работы:

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

Теоретическое введение

Swing в Джава — это набор инструментов графического интерфейса пользователя (GUI), который включает компоненты GUI. Swing предоставляет богатый набор виджетов и пакетов для создания сложных компонентов графического интерфейса пользователя для приложений Java. Swing является частью Java Foundation Classes (JFC), который представляет собой API для программирования Java GUI, который предоставляет GUI. Библиотека Java Swing построена на основе Java Abstract Widget Toolkit (AWT), более старого, зависящего от платформы инструментария графического интерфейса пользователя. Вы можете использовать простые программные компоненты Java с графическим интерфейсом пользователя, такие как кнопка, текстовое поле и т. д. Из библиотеки, и вам не нужно создавать компоненты с нуля. Схема иерархии классов графической библиотеки Swing.

Задание №1

Условие задачи:

Напишите интерактивную программу с использованием GUI имитирует таблицу результатов матчей между командами Милан и Мадрид.

Порядок работы:

1. Создайте пользовательское JFrame приложение, у которого есть следующие компоненты GUI:
 - 1.1 Одна кнопка JButton подписана “AC Milan”
 - 1.2 Другая JButton подписана “Real Madrid”
 - 1.3 Надпись JLabel содержит текст “Result: 0 X 0”
 - 1.4 Надпись JLabel содержит текст “Last Scorer: N/A”
 - 1.5 Надпись Label содержит текст “Winner: DRAW”;

Решение:

Решение поставленной задачи представлено в листинге программы №1.

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

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

        final int[] milanScore = {0};
        final int[] rmScore = {0};

        JFrame frame = new JFrame("Match");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 200);
        frame.setLayout(new GridLayout(4, 1));

        JButton button_1 = new JButton("AC Milan");
        JButton button_2 = new JButton("Real Madrid");

        JLabel result = new JLabel("Result: 0 X 0");
        JLabel lastScorer = new JLabel("Last Scorer: N/A");
        JLabel winner = new JLabel("Winner: DRAW");
        JPanel panel = new JPanel();

        panel.add(button_1);
        panel.add(button_2);

        frame.getContentPane().add(panel);
        frame.add(result);
        frame.add(lastScorer);
        frame.add(winner);
        button_1.addActionListener(e -> {
            milanScore[0]++;
            result.setText("Result: " + milanScore[0] + " X " + rmScore[0]);
            lastScorer.setText("Last Scorer: AC Milan");
            if (milanScore[0] > rmScore[0]) {
                winner.setText("Winner: AC Milan");
            } else if (rmScore[0] > milanScore[0]) {
                winner.setText("Winner: Real Madrid");
            } else {
                winner.setText("Winner: DRAW");
            }
        });
    }
}
```

Листинг №1

```
button_2.addActionListener(e -> {
    rmScore[0]++;
    result.setText("Result: " + milanScore[0] + " X " + rmScore[0]);
    lastScorer.setText("Last Scorer: Real Madrid");
    if (milanScore[0] > rmScore[0]) {
        winner.setText("Winner: AC Milan");
    } else if (rmScore[0] > milanScore[0]) {
        winner.setText("Winner: Real Madrid");
    } else {
        winner.setText("Winner: DRAW");
    }
});

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

Результат выполнения программы:

Результат выполнения программы представлен на *Рисунке 1*.

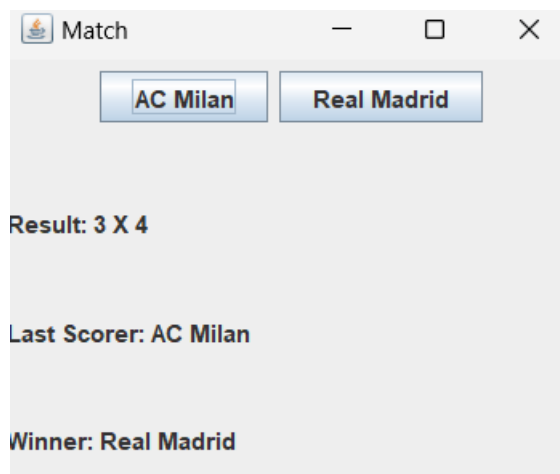


Рисунок 1

Вывод

Написали интерактивную программу с использованием GUI имитирует таблицу результатов матчей между командами Милан и Мадрид.

Практическая работа №6

Цель работы:

Научится разрабатывать в практике пользовательские интерфейсы, и применять их в программах на языке Джава.

Теоретическое введение

Интерфейс в Java это разновидность класса. В качестве компонентов интерфейс имеет поля данных только статические константы и в качестве методов только абстрактные методы

Интерфейс в Java — это механизм для достижения определенного рода абстракции. Интерфейсе Java не может включать никаких других методов кроме абстрактных. В составе интерфейса методы только объявлены, у них нет реализации или тела метода. Это похоже на механизм виртуальных функций в языке C++. Интерфейс содержит только объявления методов, то есть тело метода отсутствует. В Джава может быть объявлен пустой интерфейс, который не содержит ни одного объявления метода. Такие интерфейсы называются этикетками. Все методы входящие в интерфейс объявляются как `abstract public`, но вы можете не писать это перед самым методом, так как все методы, входящие в интерфейс и так по умолчанию абстрактные и с открытым - `public` доступом (начиная с версии Java 8). В Java 9 методы могут быть объявлены с модификатором `private`. Интерфейсы используются для достижения абстракции и множественного наследования в языке Джава. Потому что один и тот же интерфейс может использоваться для реализации разными классами

Интерфейс Java также представляет отношение классами “IS- a”

Преимущества интерфейсов

Существуют по крайней мере три веские причины использовать интерфейсы:

1. Они используется для достижения абстракции.
2. Благодаря интерфейсам мы можем поддерживать механизм множественного наследования.
3. Они использовать для достижения слабой связанности кода (`low coupling code`)

Объявление интерфейсов Интерфейс объявляется с помощью ключевого слова `interface`. Он обеспечивает полную абстракцию; это

означает, что все методы в интерфейсе объявлены с пустым телом, а все поля по умолчанию являются общедоступными, статическими и окончательными (объявлены с модификатором `final`). Класс, реализующий интерфейс, должен реализовывать все методы, объявленные в интерфейсе.

Задание №1

Условие задачи:

Напишите два класса MovablePoint и MovableCircle - которые реализуют интерфейс Movable (см рис. 1)

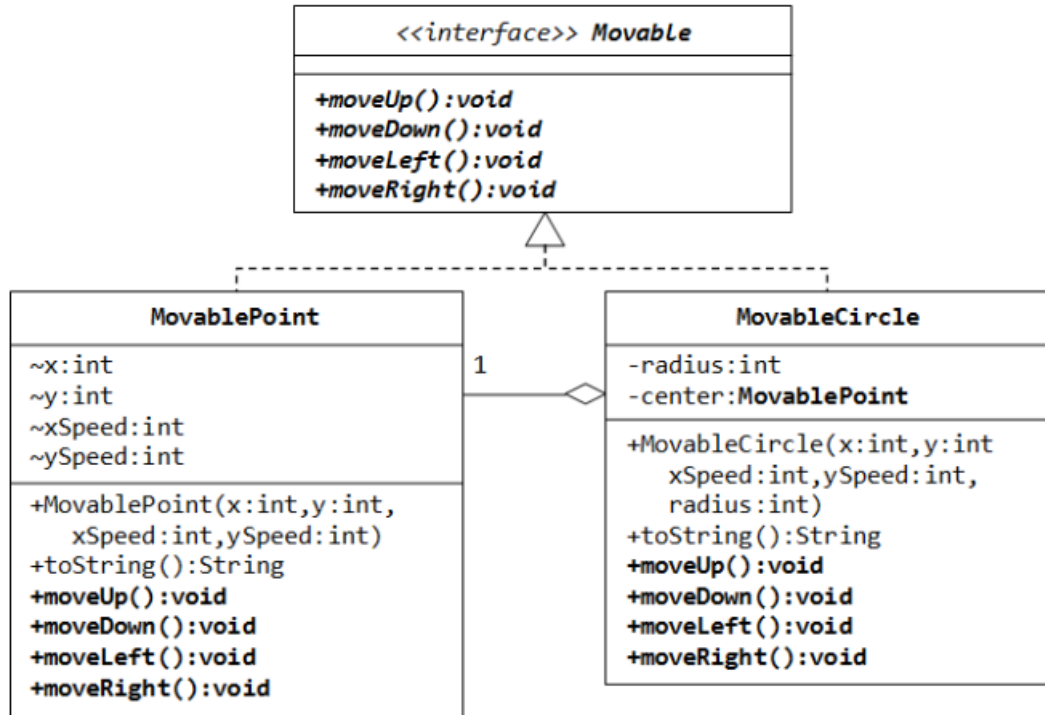


Рисунок 1

Решение:

```
public class Main {
    public static void main(String[] args) {
        MovableCircle movable_circle = new MovableCircle(10, 0, 0, 1, 1);
        System.out.println(movable_circle);
        movable_circle.moveUp();
        movable_circle.moveUp();
        movable_circle.moveRight();
        System.out.println(movable_circle);
    }
}

interface Movable {
    void moveUp();

    void moveDown();

    void moveLeft();

    void moveRight();
}

class MovablePoint implements Movable {
    private int x;
    private int y;
    private final int xSpeed;
    private final int ySpeed;

    MovablePoint(int x, int y, int xSpeed, int ySpeed) {
        this.x = x;
        this.y = y;
        this.xSpeed = xSpeed;
        this.ySpeed = ySpeed;
        System.out.println("\nMovablePoint object was created");
    }

    public String toString() {
        return "MovablePoint object - x: " + this.x + ", y: " + this.y + ", xSpeed: " + this.xSpeed +
            ", ySpeed: " + this.ySpeed;
    }

    public void moveUp() {
        this.y += this.ySpeed;
        System.out.println("MovablePoint object moved up by " + this.ySpeed);
    }

    public void moveDown() {
        this.y -= this.ySpeed;
        System.out.println("MovablePoint object moved down by " + this.ySpeed);
    }
}
```

Листинг 1

```
public void moveLeft() {
    this.x -= this.xSpeed;
    System.out.println("MovablePoint object moved left by " + this.xSpeed);
}

public void moveRight() {
    this.x += this.xSpeed;
    System.out.println("MovablePoint object moved right by " + this.xSpeed);
}
}

class MovableCircle implements Movable {
    private final int radius;
    private final MovablePoint center;

    MovableCircle(int radius, int x, int y, int xSpeed, int ySpeed) {
        this.center = new MovablePoint(x, y, xSpeed, ySpeed);
        if (radius > 0) {
            this.radius = radius;
        } else {
            this.radius = 1;
        }
        System.out.println("MovableCircle object was created");
    }

    public String toString() {
        return "MovableCircle object - radius: " + this.radius + ", center: " + center;
    }

    public void moveUp() {
        this.center.moveUp();
        System.out.println("MovableCircle object moved up");
    }

    public void moveDown() {
        this.center.moveDown();
        System.out.println("MovableCircle object moved down");
    }

    public void moveLeft() {
        this.center.moveLeft();
        System.out.println("MovableCircle object moved left");
    }

    public void moveRight() {
        this.center.moveRight();
        System.out.println("MovableCircle object moved right");
    }
}
```

Результат выполнения программы:

Результат выполнения программы представлен на рисунке 2.

```
MovablePoint object was created
MovableCircle object was created
MovableCircle object - radius: 10, center: MovablePoint object - x: 0, y: 0, xSpeed: 1, ySpeed: 1
MovablePoint object moved up by 1
MovableCircle object moved up
MovablePoint object moved up by 1
MovableCircle object moved up
MovablePoint object moved right by 1
MovableCircle object moved right
MovableCircle object - radius: 10, center: MovablePoint object - x: 1, y: 2, xSpeed: 1, ySpeed: 1
```

Рисунок 2

Вывод

Научились разрабатывать в практике пользовательские интерфейсы, и применять их в программах на языке Джава.

Практическая работа №7

Цель работы:

Научится разрабатывать в практике пользовательские интерфейсы, и применять их в программах на языке Джава.

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 Джава 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`.

Определим следующий интерфейс:

```
public interface Printable{  
    void print();  
}
```

Интерфейс может определять различные методы, которые, так же, как и абстрактные методы абстрактных классов не имеют реализации. В данном случае объявлен только один метод.

Все методы интерфейса не имеют модификаторов доступа, но фактически по умолчанию доступ `public`, так как цель интерфейса - определение функционала для реализации его классом. Поэтому весь функционал должен быть открыт для реализации.

И также при объявлении интерфейса надо учитывать, что только один интерфейс в файле может иметь тип доступа `public`. А его название должно совпадать с именем файла. Остальные интерфейсы (если такие имеются в файле `java`) не должны иметь модификаторов доступа

Интерфейс может определять различные методы, которые, так же как и абстрактные методы абстрактных классов не имеют реализации. В данном случае объявлен только один метод

Все методы интерфейса не имеют модификаторов доступа, но фактически по умолчанию доступ `public`, так как цель интерфейса -

определение функционала для реализации его классом. Поэтому весь функционал должен быть открыт для реализации.

И также при объявлении интерфейса надо учитывать, что только один интерфейс в файле может иметь тип доступа `public`. А его название должно совпадать с именем файла. Остальные интерфейсы (если такие имеются в файле `java`) не должны иметь модификаторов доступа.

Задание №1

Условие задачи:

Создайте в draw.io UML диаграмму и напишите по ней реализацию.

Диаграмма должна включать в себя следующие элементы: интерфейс Movable, содержащий в себе методы для движения прямоугольника (вверх, вниз, влево, вправо) и класс MovableRectangle (движущийся прямоугольник), реализующий интерфейс Movable.

Решение:

Диаграмма представлена на рисунке 1, код программы – листинг 1.

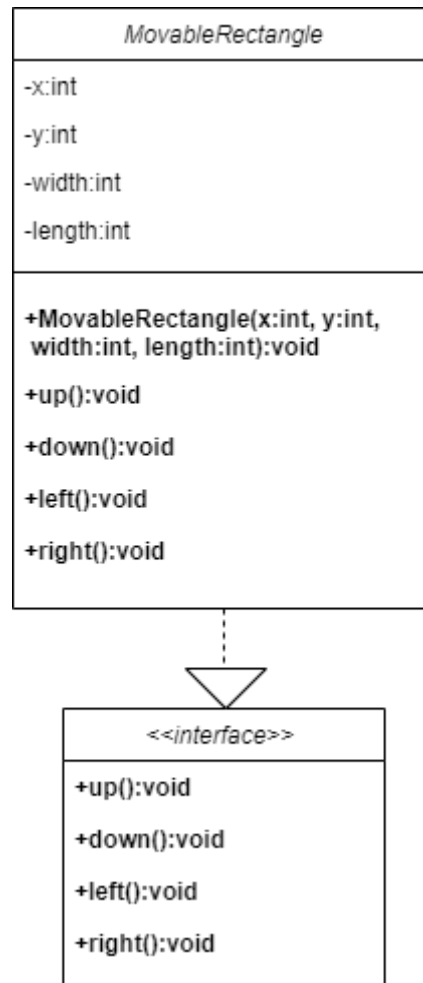


Рисунок 1.

Листинг 1.

```
interface Movable {  
    void up();  
  
    void down();  
  
    void left();  
  
    void right();  
}
```

Продолжение листинга 1.

```
class MovableRectangle implements Movable {  
    private int x;  
    private int y;  
    private final int width;  
    private final int length;  
  
    public MovableRectangle(int x, int y, int width, int length) {  
        this.x = x;  
        this.y = y;  
        this.width = width;  
        this.length = length;  
    }  
  
    public void ToString() {  
        System.out.println("Length rectangle: " + this.length);  
        System.out.println("Width rectangle: " + this.width);  
        System.out.println("Coordinate rectangle: " + this.x + ", " + this.y);  
    }  
  
    @Override  
    public void up() {  
        this.y++;  
    }  
  
    @Override  
    public void down() {  
        this.y--;  
    }  
  
    @Override  
    public void left() {  
        this.x--;  
    }  
}
```

Продолжение листинга 1.

```
@Override  
  
public void right() {  
    this.x++;  
}  
  
public static void main(String[] args) {  
    MovableRectangle rectangle = new MovableRectangle(0, 0, 1, 1);  
    rectangle.ToString();  
    rectangle.up();  
    rectangle.left();  
    rectangle.down();  
    rectangle.right();  
    rectangle.down();  
    rectangle.ToString();  
}  
}
```

Результат выполнения программы:

Результат выполнения программы представлен на рисунке 2.

```
Length rectangle: 1  
Width rectangle: 1  
Coordinate rectangle: 0, 0  
Length rectangle: 1  
Width rectangle: 1  
Coordinate rectangle: 0, -1
```

Рисунок 2.

Вывод

Научились разрабатывать в практике пользовательские интерфейсы, и применять их в программах на языке Джава.

Практическая работа №8

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.

Итак, рекурсивная функция состоит из:

1. условие остановки или же базового случая или условия;
2. условие продолжения или шага рекурсии — способ сведения сложной задачи к более простым подзадачам.

Задание №1

Условие задачи:

Задание 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`.

Решение:

Решение представлено в листинге 1.

Листинг 1.

```
public class Main {  
    public static void recursion(int n, int k) {  
        for (int i = 1; (i <= k) && (n > 0); i++) {  
            System.out.println(k);  
            n--;  
        }  
        if (n > 0) {  
            recursion(n, ++k);  
        }  
    }  
  
    public static void main(String[] args) {  
        recursion(10, 1);  
    }  
}
```

Результат выполнения программы:

Результат выполнения программы представлен на рисунке 1.

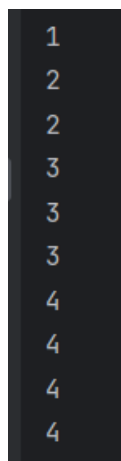


Рисунок 1.

Задание №16

Условие задачи:

Задание Количество элементов, равных максимуму.

Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом ноль. Определите, какое количество элементов этой последовательности, равны ее наибольшему элементу.

В этой задаче нельзя использовать глобальные переменные. Функция получает данные, считывая их с клавиатуры, а не получая их в виде параметра.

72 В программе на языке Python функция возвращает результат в виде кортежа из нескольких чисел, и функция вообще не получает никаких параметров. В программе на языке C++ результат записывается в переменные, которые передаются в функцию по ссылке. Других параметров, кроме как используемых для возврата значения, функция не получает. Гарантируется, что последовательность содержит хотя бы одно число (кроме нуля).

Решение:

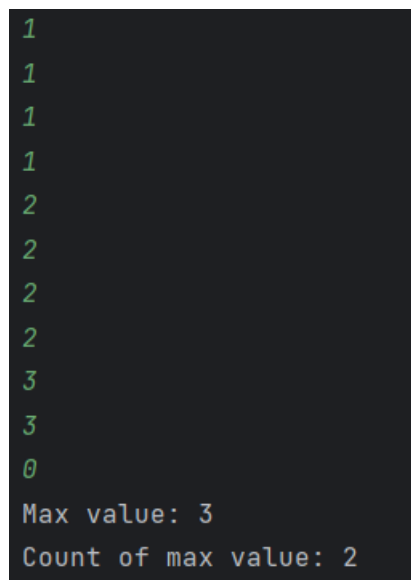
Решение задания представлено в листинге 2.

Листинг 2.

```
public class Main {  
    public static void recursion(int max, int count) {  
        java.util.Scanner in = new java.util.Scanner(System.in);  
        int n = in.nextInt();  
        if (n == 0) {  
            System.out.println("Max value: " + max);  
            System.out.println("Count of max value: " + count);  
        } else if (n > max) {  
            recursion(n, 1);  
        } else if (n == max) {  
            recursion(max, count + 1);  
        } else {  
            recursion(max, count);  
        }  
    }  
  
    public static void main(String[] args) {  
        recursion(0, 1);  
    }  
}
```

Результат выполнения программы:

Результат выполнения программы представлен на рисунке 2.



The image shows a terminal window with a dark background. It displays a list of numbers: 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 0. Below the list, it shows the results of a calculation: 'Max value: 3' and 'Count of max value: 2'.

```
1  
1  
1  
1  
2  
2  
2  
2  
3  
3  
0  
Max value: 3  
Count of max value: 2
```

Рисунок 2.

Задание №17

Условие задачи:

Задание Максимум последовательности.

Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Определите наибольшее значение числа в этой последовательности

В этой задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция возвращает единственное значение: максимум считанной последовательности.

Гарантируется, что последовательность содержит хотя бы одно число (кроме нуля).

Решение:

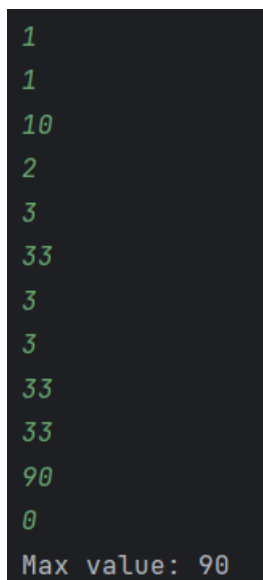
Решение задачи представлено в листинге 3.

Листинг 3.

```
public class Main {  
    public static void recursion(int max) {  
        java.util.Scanner in = new java.util.Scanner(System.in);  
        int n = in.nextInt();  
        if (n == 0) {  
            System.out.println("Max value: " + max);  
        } else if (n > max) {  
            recursion(n);  
        } else {  
            recursion(max);  
        }  
    }  
  
    public static void main(String[] args) {  
        recursion(-1030302032);  
    }  
}
```

Результат выполнения программы:

Результат выполнения программы представлен на рисунке 3.

A screenshot of a terminal window with a dark background. It displays a list of numbers in green text: 1, 1, 10, 2, 3, 33, 3, 3, 33, 33, 90, 0. At the bottom, in white text, it says "Max value: 90".

```
1
1
10
2
3
33
3
3
33
33
90
0
Max value: 90
```

Рисунок 3

Вывод

Разработали рекурсивные алгоритмы на языке Java.

Практическая работа №9

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.

Процесс сортировки основан на упорядочивании конкретных значений, например:

1. Сортировка списка результатов экзаменов баллов в порядке возрастания результата;
2. Сортировка списка людей в алфавитном порядке по фамилии

Есть много алгоритмов для сортировки списка элементов, которые различаются по эффективности

Алгоритм сортировки вставками.

Работа метода сортировки состоит из следующих шагов:

1. выбрать любой элемент из списка элементов и вставить его в надлежащее место в отсортированный подсписок;
2. повторять предыдущий шаг, до тех пор, пока все элементы не будут вставлены.

Более детально:

1. рассматриваем первый элемент списка как отсортированный подсписок (то есть первый элемент списка)
2. вставим второй элемент в отсортированный подсписок, сдвигая первый элемент по мере необходимости, чтобы освободить место для вставки нового элемента;
3. вставим третий элемент в отсортированный подсписок (из двух элементов), сдвигая элементы по мере необходимости;
4. повторяем до тех пор, пока все значения не будут вставлены на свои соответствующие позиции.

Алгоритм быстрой сортировки (Quick Sort).

Состоит из последовательного выполнения двух шагов:

1. массив $A[1..n]$ разбивается на два непустых подмассивов по отношению к "опорному элементу";
2. два подмассива сортируются рекурсивно посредством Quick Sort. Алгоритм сортировка слиянием (Merge Sort).

Состоит из последовательного выполнения трех шагов:

1. разделить массив $A[1..n]$ на 2 равные части;
2. провести сортировку слиянием двух подмассивов (рекурсивно);
3. объединить (соединить) два отсортированных подмассива.

Задание №1

Условие задачи:

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

Решение:

Решение представлено в листингах 1 и 2.

Листинг 1 - Student

```
public class Student implements Comparable<Student> {
    private final int id;
    private final String name;

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

    public int getId() {
        return this.id;
    }

    public int compareTo(Student otherStudent) {
        // Сравниваем объекты Student на основе idNumber
        return Integer.compare(this.getId(), otherStudent.getId());
    }

    public static void main(String[] args) {

    }
}
```

```
public class TesterStudent {
    public static void main(String[] args) {
        Student[] students = {
            new Student(4, "S_1"),
            new Student(3, "S_2"),
            new Student(1, "S_3"),
            new Student(2, "S_4")
        };
        System.out.println(printID(students));

        selectionSort(students);

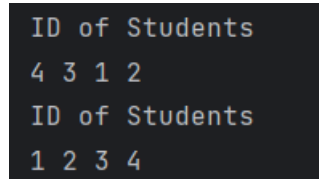
        System.out.println(printID(students));
    }

    private static String printID(Student[] array) {
        System.out.println("ID of Students");
        StringBuilder s = new StringBuilder();
        for (Student student : array) {
            s.append(student.getId()).append(" ");
        }
        return s.toString();
    }

    public static void selectionSort(Student[] arr) {
        int n = arr.length;
        for (int i = 1; i < n; i++) {
            Student key = arr[i];
            int j = i - 1;
            while (j >= 0 && arr[j].compareTo(key) > 0) {
                arr[j + 1] = arr[j];
                j--;
            }
            arr[j + 1] = key;
        }
    }
}
```

Результат выполнения программы:

Результат выполнения программы представлен на рисунке 1.



```
ID of Students  
4 3 1 2  
ID of Students  
1 2 3 4
```

Рисунок 1.

Вывод

Освоили на практике методы сортировки с использованием приемов программирования на объектно-ориентированном языке Java.

Практическая работа №10

Цель работы:

Закрепить знания в области использования стандартных интерфейсов языка Джава, научиться применять интерфейсы для разработки практических программ на Джаве.

Теоретическое введение

Comparator и Comparable в Java

Интерфейс Comparable содержит один единственный метод `int compareTo(E item)`, который сравнивает текущий объект с объектом, переданным в качестве параметра. Если этот метод возвращает отрицательное число, то текущий объект будет располагаться перед тем, который передается через параметр. Если метод вернет положительное число, то, наоборот, после второго объекта. Если метод возвратит ноль, значит, оба объекта равны.

Интерфейс Comparator.

Однако перед нами может возникнуть проблема, что, если разработчик не реализовал в своем классе, который мы хотим использовать, интерфейс Comparable, либо реализовал, но нас не устраивает его функциональность, и мы хотим ее переопределить? На этот случай есть еще более гибкий способ, предполагающий применение интерфейса Comparator.

Задание №1-3

Условие задачи:

1. Создать свой класс Student со всеми переменными экземпляра, конструктором, включающим все переменные, предпочтительно использовать 79 геттеры и сеттеры для каждой переменной. Класс студент имеет свойства: Имя, Фамилия, Специальность, Курс, Группа.

2. Напишите класс SortingStudentsByGPA (может у вас называться Tester или Main, так как содержит функцию main()) создайте поле как массив объектов Student с названием idNumber, вы можете использовать как массив, так и ArrayList или TreeSet для хранения данных о студентах. Добавьте методы класса: 1) заполнения массива setArray() 2) метод для сортировки по среднему баллу студентов quicksort() который реализует интерфейс Comparator таким образом, чтобы он сортировал студентов с их итоговым баллом в порядке убывания. В качестве алгоритма сортировки использовать методы сортировок: слиянием и быструю сортировку (добавьте в класс еще один метод). 3) метод для вывода массива студентов outArray() 4) Добавьте в класс возможность сортировать список студентов по другому полю.

3. Напишите программу, которая объединяет два списка данных о студентах в один отсортированный списках.

Решение:

Решение задачи представлено в листингах 1-3.

Листинг 1 - Student

```
public class Student implements Comparable {

    private int idNum;
    private int GPA;

    public Student(int idNum, int GPA) {
        this.idNum = idNum;
        this.GPA = GPA;
    }

    @Override
    public String toString() {
        return "Student{" +
            "idNum=" + idNum +
            ", GPA=" + GPA +
            '}';
    }

    public int getIdNum() {
        return idNum;
    }

    public void setIdNum(int idNum) {
        this.idNum = idNum;
    }

    public int getGPA() {
        return GPA;
    }
}
```


Продолжение листинга 1.

```
public void setGPA(int GPA) {
    this.GPA = GPA;
}

@Override
public int compareTo(Object o) {
    if (!(o instanceof Student))
        throw new IllegalArgumentException("Object is not students!");
    // < 0 -> o, 0 -> ==, >0 -> this;
    return this.idNum - ((Student) o).idNum;
}
}
```

Листинг 2 – StudentComp

```
import java.util.Comparator;

public class StudentComp implements Comparator {
    @Override
    public int compare(Object o1, Object o2) {
        if (!(o1 instanceof Student && o2 instanceof Student))
            throw new IllegalArgumentException("Object is not students.");

        return ((Student) o1).getGPA() - ((Student) o2).getGPA();
    }
}
```

```
import java.util.ArrayList;

public class TestStudent {
    public static void main(String[] args) {
        Student[] students = new Student[]{
            new Student(12, 4),
            new Student(10, 5),
            new Student(189, 81),
            new Student(2, 320)
        };
        for (Student s : students) {
            System.out.println(s);
        }
        System.out.println();
        // Вставки по id
        for (int i = 1; i < students.length; i++) {
            Student current = students[i];
            int j = i - 1;
            for (; j >= 0 && current.compareTo(students[j]) < 0; j--) {
                students[j + 1] = students[j];
            }
            students[j + 1] = current;
        }
        for (Student s : students) {
            System.out.println(s);
        }
        // Быстрая по GPA
        System.out.println();
        qSort(students, students.length - 1, 0);
        for (Student s : students) {
            System.out.println(s);
        }
    }
}
```

Продолжение листинга 3.

```
// merge sort
System.out.println();
Student[] students2 = new Student[]{
    new Student(35, 412),
    new Student(16, 105),
    new Student(18, 128),
    new Student(222, 201)
};
Student[] allStudents = new Student[students.length + students2.length];
System.arraycopy(students, 0, allStudents, 0, students.length);
System.arraycopy(students2, 0, allStudents, students.length, students2.length);
mergeSort(allStudents, allStudents.length);
for (Student s : allStudents) {
    System.out.println(s);
}

}

public static void mergeSort(Student[] a, int n) {
    if (n < 2) {
        return;
    }
    int mid = n / 2;
    Student[] l = new Student[mid];
    Student[] r = new Student[n - mid];

    System.arraycopy(a, 0, l, 0, mid);
    System.arraycopy(a, mid, r, 0, n - mid);
    mergeSort(l, mid);
    mergeSort(r, n - mid);

    merge(a, l, r, mid, n - mid);
}
```

Продолжение листинга 3.

```
public static void merge(
    Student[] a, Student[] l, Student[] r, int left, int right) {

    int i = 0, j = 0, k = 0;
    while (i < left && j < right) {
        if (l[i].compareTo(r[j]) <= 0) {
            a[k++] = l[i++];
        } else {
            a[k++] = r[j++];
        }
    }
    while (i < left) {
        a[k++] = l[i++];
    }
    while (j < right) {
        a[k++] = r[j++];
    }
}

private static final StudentComp comp = new StudentComp();

public static void qSort(Object[] array, int high, int low) {
    if (array == null || array.length == 0) return;
    if (high <= low) return;

    Object middle = array[(low + high) / 2];
    ArrayList<Object> left = new ArrayList<>();
    ArrayList<Object> right = new ArrayList<>();
    ArrayList<Object> eq = new ArrayList<>();
```

Продолжение листинга 3.

```
    for (int i = low; i <= high; i++) {
        if (comp.compare(array[i], middle) > 0) {
            right.add(array[i]);
        } else if (comp.compare(array[i], middle) < 0)
            left.add(array[i]);
        else eq.add(array[i]);
    }
    Object[] leftArr;
    Object[] rightArr;
    if (!left.isEmpty()) {
        leftArr = left.toArray();
        qSort(leftArr, left.size() - 1, 0);
        System.arraycopy(leftArr, 0, array, low, left.size());
    }
    System.arraycopy(eq.toArray(), 0, array, low + left.size(), eq.size());

    if (!right.isEmpty()) {
        rightArr = right.toArray();
        qSort(rightArr, right.size() - 1, 0);
        System.arraycopy(rightArr, 0, array, low + left.size() + eq.size(), right.size());
    }

}

}
```

Результат выполнения программы:

Результат выполнения программы представлен на рисунке 1.

```
Student{idNum=12, GPA=4}
Student{idNum=10, GPA=5}
Student{idNum=189, GPA=81}
Student{idNum=2, GPA=320}

Student{idNum=2, GPA=320}
Student{idNum=10, GPA=5}
Student{idNum=12, GPA=4}
Student{idNum=189, GPA=81}

Student{idNum=12, GPA=4}
Student{idNum=10, GPA=5}
Student{idNum=189, GPA=81}
Student{idNum=2, GPA=320}

Student{idNum=2, GPA=320}
Student{idNum=10, GPA=5}
Student{idNum=12, GPA=4}
Student{idNum=16, GPA=105}
Student{idNum=18, GPA=128}
Student{idNum=35, GPA=412}
Student{idNum=189, GPA=81}
Student{idNum=222, GPA=201}
```

Рисунок 1.

Вывод

Закрепили знания в области использования стандартных интерфейсов языка Джава, научиться применять интерфейсы для разработки практических программ на Джаве.

Практическая работа №11

Цель работы:

Научиться работать с датами и временем, применять методы класса Date и Calendar, других классов для обработки строк.

Теоретическое введение

В Java есть много классов, доступных для работы с датой/временем.

Класс `Date` изначально предоставлял набор функций для работы с датой - для получения текущего года, месяца и т.д. однако сейчас все эти методы не рекомендованы к использованию и практически всю функциональность для этого предоставляет класс `Calendar`. Класс `Date` так же определен в пакете `java.sql` поэтому желательно указывать полностью квалифицированное имя класса `Date`.

Существует несколько конструкторов класса `Date` однако рекомендовано к использованию два:

`Date()` и `Date(long date)` второй конструктор использует в качестве параметра значение типа `long` который указывает на количество миллисекунд прошедшее с 1 Января 1970, 00:00:00 по Гринвичу. Первый конструктор создает дату использует текущее время и дату (т.е. время выполнения конструктора). Фактически это эквивалентно второму варианту `new Date(System.currentTimeMillis)`; Можно уже после создания экземпляра класса `Date` использовать метод `setTime(long time)`, для того, что бы задать текущее время.

Для сравнения дат служат методы `after(Date date)`, `before(Date date)` которые возвращают булевское значение в зависимости от того выполнено условие или нет. Метод `compareTo(Date anotherDate)` возвращает значение типа `int` которое равно -1 если дата меньше сравниваемой, 1 если больше и 0 если даты равны. Метод `toString()` представляет строковое представление даты, однако для форматирования даты в виде строк рекомендуется пользоваться классом `SimpleDateFormat` определенном в пакте `java.text`.

Задание №1

Условие задачи:

Написать программу, выводящую фамилию разработчика, дату и время получения задания, а также дату и время сдачи задания. Для получения последней даты и времени использовать класс `Date` из пакета `java.util.*` (Объявление `Dated=newDate()` или метод `System.currentTimeMillis()`).

Решение:

Решение представлено на листинге 1.

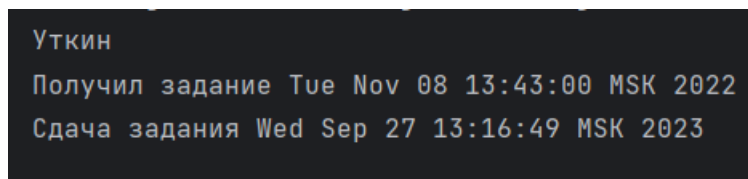
Листинг 1.

```
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

public class Family {
    public static void main(String[] args) {
        System.out.println("Уткин");
        Calendar calendar = new GregorianCalendar(2022, Calendar.NOVEMBER, 8, 13, 43);
        Date date1 = calendar.getTime();
        System.out.println("Получил задание " + date1);
        Date date = new Date();
        System.out.println("Сдача задания " + date);
    }
}
```

Результат выполнения программы:

Результат выполнения программы представлен на рисунке 1.

A screenshot of a terminal window with a dark background and light-colored text. The text displays the name 'Уткин' followed by two lines of task completion information: 'Получил задание Tue Nov 08 13:43:00 MSK 2022' and 'Сдача задания Wed Sep 27 13:16:49 MSK 2023'.

```
Уткин
Получил задание Tue Nov 08 13:43:00 MSK 2022
Сдача задания Wed Sep 27 13:16:49 MSK 2023
```

Рисунок 1.

Задание №2

Условие задачи:

Приложение, сравнивающее текущую дату и дату, введенную пользователем с текущим системным временем.

Решение:

Решение представлено в листинге 2.

--

Используемая литература

Конспект лекций по дисциплине «Программирование на языке Джава»,
РТУ МИРЭА, лектор – старший преподаватель Зорина Н.В.