



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>	9
Цель работы:.....	9
Теоретическое введение	10
Задача №3 и №4	11
Условие задачи:	11
Решение:	12
Результат выполнения программы:	14
Задача №5	15
Условие задачи:	15
Решение:	16
Результат выполнения программы:	17
Задача №6	18
Условие задачи	18
Решение:	19
Результат выполнения программы:	20
Задача №7	21
Условие задачи:	21
Решение:	22
Результат выполнения программы:	23
Вывод:	24
<i>Практическая работа №2</i>	25
Цель работы:.....	25
Теоретическое введение	26
Задача №1	27
Условие:.....	27
Решение:	28
Класс Author.....	28
Класс TestAuthor.....	30
Результат выполнения программы:	31
Задача №2	32
Условие задачи:	32
Решение:	33
Результат выполнения программы:	34
Задача №5	35
Условие задачи:	35
Решение:	36
Результат выполнения программы:	39
Вывод:	40

<i>Практическая работа №3</i>	41
Цель работы:	41
Теоретическое введение	42
Задача №1	43
Условие задачи:	43
Решение:	44
Результат выполнения программы:	45
Вывод	46
<i>Практическая работа №4</i>	47
Цель работы:	47
Теоретическое введение	48
Задача №1	49
Условие задачи:	49
Решение:	50
Результат выполнения программы:	51
Задача №1 Блок 2	52
Условие задачи:	52
Решение:	53
Результат выполнения программы:	58
Вывод	59
<i>Практическая работа №5</i>	60
Цель работы:	60
Теоретическое введение	61
Задание №1	62
Условие задачи:	62
Решение:	63
Результат выполнения программы:	65
Вывод	66
<i>Практическая работа №6</i>	67
Цель работы:	67
Теоретическое введение	68
Задание №1	70
Условие задачи:	70
Решение:	71
Результат выполнения программы:	73
Вывод	74
<i>Практическая работа №7</i>	75
Цель работы:	75

Теоретическое введение	76
Задание №1	78
Условие задачи:	78
Решение:	79
Результат выполнения программы:	82
Вывод	83
<i>Практическая работа №8</i>	84
Цель работы:	84
Теоретическое введение	85
Задание №1	86
Условие задачи:	86
Решение:	87
Результат выполнения программы:	88
Задание №16	89
Условие задачи:	89
Решение:	90
Результат выполнения программы:	91
Задание №17	92
Условие задачи:	92
Решение:	93
Результат выполнения программы:	94
Вывод	95
<i>Практическая работа №9</i>	96
Цель работы:	96
Теоретическое введение	97
Задание №1	99
Условие задачи:	99
Решение:	100
Результат выполнения программы:	102
Вывод	103
<i>Практическая работа №10</i>	104
Цель работы:	104
Теоретическое введение	105
Задание №1-3	106
Условие задачи:	106
Решение:	107
Результат выполнения программы:	113
Вывод	114
<i>Практическая работа №11</i>	115

Цель работы:.....	115
Теоретическое введение	116
Задание №1	117
Условие задачи:	117
Решение:	118
Результат выполнения программы:	119
Задание №2.....	120
Условие задачи:	120
Решение:	121
Результат выполнения программы:	122
Задание №3.....	123
Условие задачи:	123
Решение:	124
Результат выполнения программы:	128
Задание №4.....	129
Условие задачи:	129
Решение:	130
Результат выполнения программы:	131
Задание №5.....	132
Условие задачи:	132
Решение:	133
Результат выполнения программы:	135
Вывод	136
<i>Практическая работа №12</i>	<i>137</i>
Цель работы:.....	137
Теоретическое введение	138
Задание 1-3	139
Условие задачи:	139
Решение:	140
Результат выполнения программы:	144
Вывод	145
<i>Практическая работа №13</i>	<i>146</i>
Цель работы:.....	146
Теоретическое введение	147
Задание №1	148
Условие задачи:	148
Решение:	149
Результат выполнения программы	150
Задание №2.....	151
Условие задачи:	151
Решение:	152

Результат выполнения программы	153
Вывод	154
<i>Практическая работа №14</i>	155
Цель работы:.....	155
Теоретическое введение	156
Задание №5.....	157
Условие задачи:	157
Решение	158
Результат выполнения программы	159
Вывод	160
<i>Практическая работа №15</i>	161
Цель работы:.....	161
Теоретическое введение	162
Задача №1	163
Условие задачи	163
Решение	164
Результат выполнения программы	166
Вывод	167
<i>Практическая работа №16</i>	168
Цель работы.....	168
Теоретическое введение	169
Задание №1	170
Условие задачи:	170
Решение	171
Результат выполнения программы	173
Вывод	174
<i>Практическая работа №17</i>	175
Цель работы.....	175
Теоретическое введение	176
Задание №3.....	177
Условие задачи	177
Решение	178
Результат выполнения программы	187
Вывод	188
<i>Практическая работа №18</i>	189
Цель работы:.....	189
Теоретическое введение.....	190

Задание №4.....	191
Условие задачи	191
Решение	192
Результат выполнения программы	193
Вывод	194
<i>Практическая работа №19</i>	<i>195</i>
Цель работы.....	195
Теоретическое введение	196
Задание №1	197
Условие задачи	197
Решение	198
Результат выполнения программы	199
Вывод	200
<i>Практическая работа №20</i>	<i>201</i>
Цель работы.....	201
Теоретическое введение	202
Задание №4.....	203
Условие задачи	203
Решение	204
Вывод	207
<i>Практическая работа №21</i>	<i>208</i>
Цель работы.....	208
Теоретическое введение	209
Задание №2.....	210
Условие задачи	210
Решение	211
Результат выполнения программы	212
Вывод	213
<i>Практическая работа №22</i>	<i>214</i>
Цель работы.....	214
Теоретическое введение	215
Задание №1	216
Условие задачи	216
Решение	217
Результат выполнения программы	219
Вывод	220
<i>Практическая работа №23</i>	<i>221</i>

Цель работы.....	221
Теоретическое введение.....	222
Задание №1.....	223
Условие задачи	223
Решение	224
Результат выполнения программы	230
Вывод	231
<i>Практическая работа №24</i>	<i>232</i>
Цель работы.....	232
Теоретическое введение.....	233
Задание №2.....	234
Условие задачи	234
Решение	235
Результат выполнения программы	238
Вывод	239
<i>Используемая литература</i>	<i>240</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*.

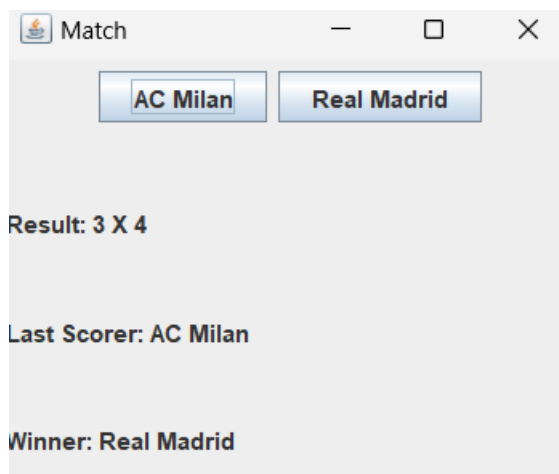


Рисунок 1

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)

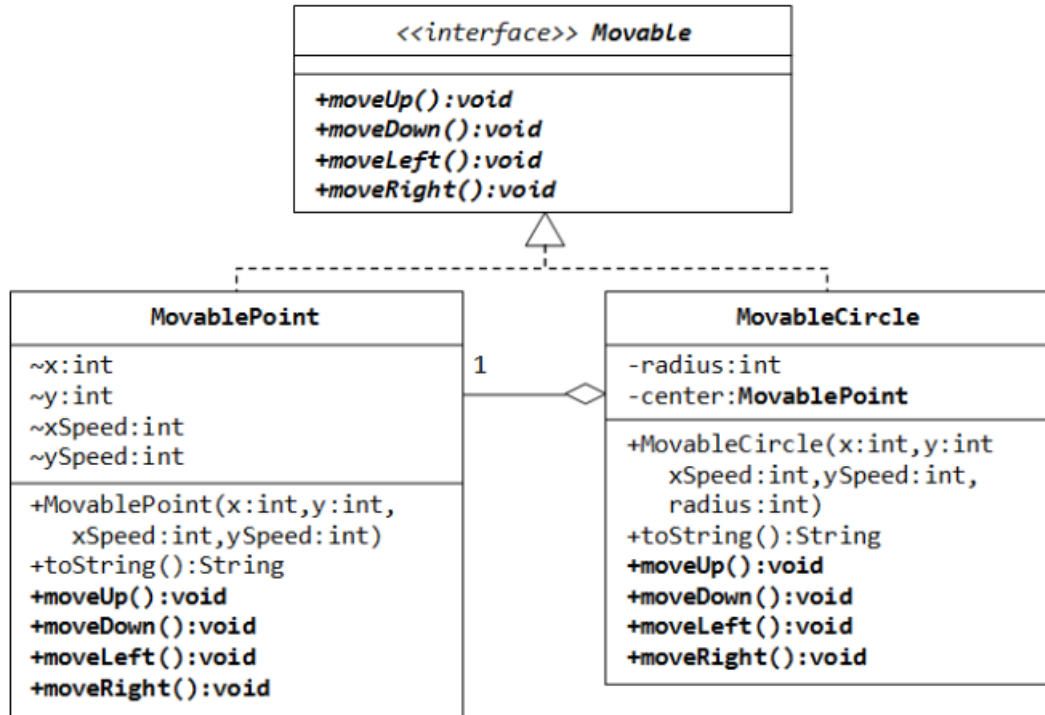


Рисунок 1

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.

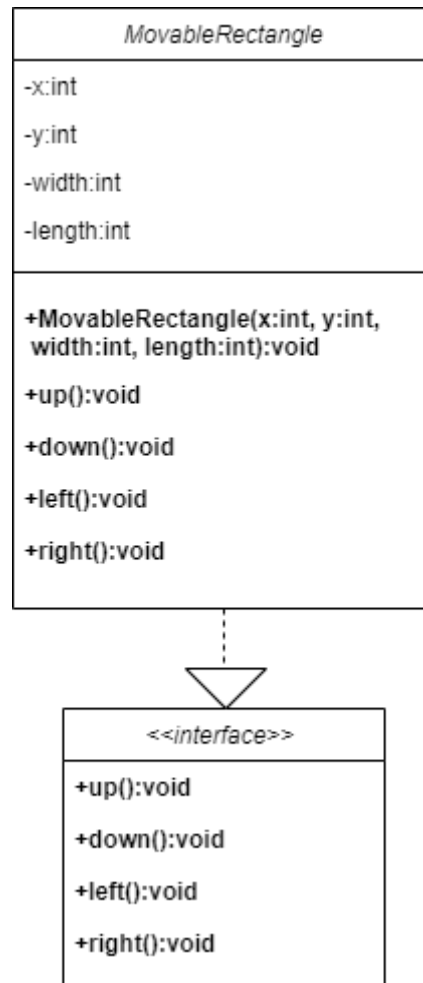


Рисунок 1.

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 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.

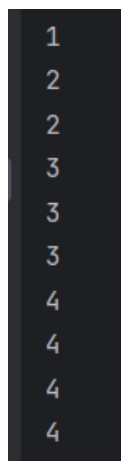


Рисунок 1.

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.

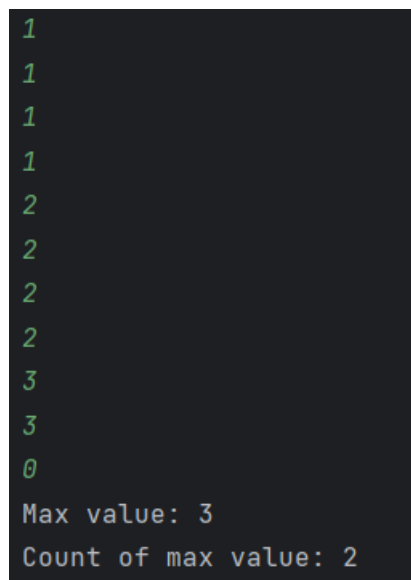
Листинг 2.

```
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:

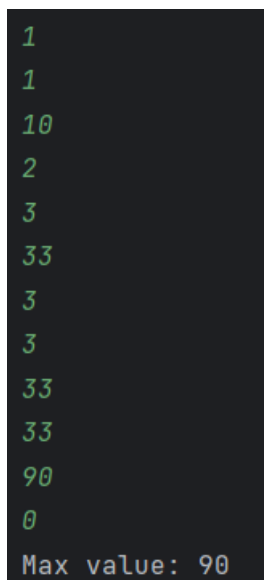
Решение задачи представлено в листинге 3.

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.



```
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 - Family.

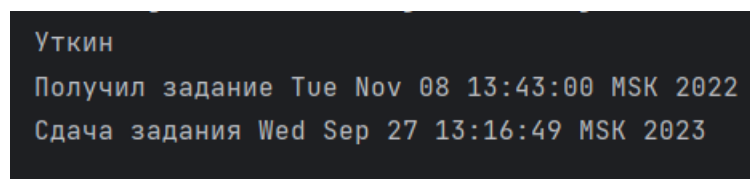
Листинг 1 - Family.

```
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 – Время выполнения задания.



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

Рисунок 1 – Время выполнения задания.

Задание №2

Условие задачи:

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

Решение:

Решение представлено в листинге 2 - Time.

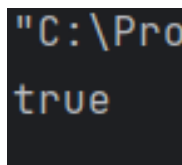
Листинг 2 - Time.

```
import java.text.SimpleDateFormat;
import java.util.Date;

public class Main {
    public static void main(String[] args) {
        String currentTime = new SimpleDateFormat("HH:mm").format(new Date());
        String timeToCompare = "15:19";
        boolean x = currentTime.equals(timeToCompare);
        System.out.println(x);
    }
}
```

Результат выполнения программы:

Результат выполнения программы представлен на рисунке 2 – Проверка времени.



```
"C:\Pro  
true
```

Рисунок 2 – Проверка времени.

Задание №3

Условие задачи:

Доработайте класс `Student` предусмотрите поле для хранения даты рождения, перепишите метод `toString()` таким образом, чтобы он разработайте метод, возвращал строковое представление даты рождения по вводимому в метод формату даты (например, короткий, средний и полный формат даты).

Решение:

Результат выполнения программы представлен в листинге 3 – Student и листинге 4 - Type.

Листинг 3 - Student

```
import java.util.ArrayList;

public class Student {
    private String name;
    private String data;
    private String form;

    Type type;

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

    public String getName() {
        return name;
    }

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

    public void setData(String data) {
        this.data = data;
    }

    public String getData() {
        return data;
    }

    public void setForm(String form) {
        this.form = form;
    }

    public String Format(String data, String format) {
        String str = null;
        switch (format) {
            case "short":
                for (int i = 0; i < data.length(); i++) {
                    //System.out.println(data.substring(0, 2));
                    str = data.substring(0, 5);
                }
                break;
        }
    }
}
```


Продолжение листинга 3 – Student

```
        case "default":
            for (int i = 0; i < data.length(); i++) {
                //System.out.println(data.substring(0, 2));
                str = data;

            }
            break;
        case "full":
            Type[] types = Type.values();

            int kill = Integer.parseInt(String.valueOf(data.charAt(4)));

            str = data.substring(0, 2) + " " + types[kill - 1].toString() + " " + data.substring(6,
10);
            break;
        }
        return str;
    }

    @Override
    public String toString() {
        return Format(data, form);
    }

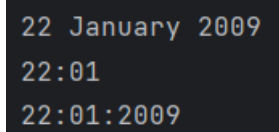
    public static void main(String[] args) {
        Student student = new Student("Tim", "22:01:2009");
        student.setForm("full");
        System.out.println(student);
    }
}
```

Листинг 4 – Type.

```
public enum Type {  
    January,  
    February,  
    March,  
    April,  
    May,  
    June,  
    July,  
    August,  
    September,  
    October,  
    November,  
    December  
}
```

Результат выполнения программы:

Результат выполнения программы представлен на рисунке 3 – Месяцы.



```
22 January 2009
22:01
22:01:2009
```

Рисунок 3- Месяцы.

Задание №4

Условие задачи:

Напишите пользовательский код, который формирует объекты Date и Calendar по следующим данным, вводимым пользователем:

<Год><Месяц><Число>

<Часы><Минуты>

Решение:

Решение задания представлено на листинге 4 – Time.

```
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

public class Time {
    public static void main(String[] args) {
        GregorianCalendar calendar = new GregorianCalendar(2017, Calendar.JULY , 25, 10,
42, 12);
        //calendar.set(Calendar.HOUR, 10);
        //calendar.set(Calendar.MINUTE, 42);
        //calendar.set(Calendar.SECOND, 12);

        //System.out.println("Год: " + calendar.get(Calendar.YEAR));
        //System.out.println("Месяц: " + (calendar.get(Calendar.MONTH) + 1));
        //System.out.println("Порядковый номер недели в месяце: " +
calendar.get(Calendar.WEEK_OF_MONTH)); //порядковый номер недели в месяце

        //System.out.println("Число: " + calendar.get(Calendar.DAY_OF_MONTH));

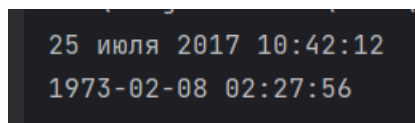
        //System.out.println("Часы: " + calendar.get(Calendar.HOUR));
        //System.out.println("Минуты: " + calendar.get(Calendar.MINUTE));
        //System.out.println("Секунды: " + calendar.get(Calendar.SECOND));
        //System.out.println("Миллисекунды: " + calendar.get(Calendar.MILLISECOND));
        DateFormat df = new SimpleDateFormat("dd MMMMMMMMMMMM yyyy hh:mm:ss");
        System.out.println(df.format(calendar.getTime()));

        SimpleDateFormat formater = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        Date date = new Date(97975676756L);

        System.out.println(formater.format(date));
    }
}
```

Результат выполнения программы:

Результат выполнения программы представлен на рисунке 4 – Time.



```
25 июля 2017 10:42:12
1973-02-08 02:27:56
```

Рисунок 4 – Time.

Задание №5

Условие задачи:

Сравнить время выполнения кода в реализации кода в виде различных структур данных из предыдущих заданий (сравнить ArrayList и LinkedList по производительности – операции вставки, удаления, добавления и поиска по образцу).

Решение:

Решение задачи представлено на листинге 5 – Сравнение времени выполнения и листинг 6 - Student.

Листинг 5 – Сравнение времени выполнения.

```
import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {
        long startTime = System.currentTimeMillis();

        ArrayList<Student> task = new ArrayList<>();

        Student robert = new Student("Robert");
        Student omar = new Student("Omar");
        Student philip = new Student("Philip");
        Student test = new Student("TEST");

        task.add(robert);
        task.add(omar);
        task.add(test);

        int position = task.indexOf(omar);
        task.remove(2); //удаляет объект под индексом 2
        System.out.println(task + "\n");
        System.out.println("Размер массива: " + task.size()); //размер массива
        System.out.println("Position Omar: " + position);
        System.out.println(task.get(0)); // хотим узнать имя под индексом 0

        task.add(1, philip); //добавляем philip в индекс 1 с смещением старого
        System.out.println(task + "\n");

        task.clear(); //очищает все элементы массива
        System.out.println(task);
        long estimatedTime = System.currentTimeMillis() - startTime;
        System.out.println(estimatedTime);
    }
}
```

Листинг 6 – Student.

```
public class Student {  
    private String name;  
  
    public Student(String name) {  
        this.name = name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    @Override  
    public String toString() {  
        return "Name: " + name;  
    }  
}
```

Результат выполнения программы:

Результат выполнения программы представлен на рисунке 5 – Сравнение времени.

```
[Name: Robert, Name: Omar]

Размер массива: 2
Position Omar: 1
Name: Robert
[Name: Robert, Name: Philip, Name: Omar]

[]
12
```

Рисунок 5 – Сравнение времени.

Вывод

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

Практическая работа №12

Цель работы:

Научиться создавать графический интерфейс пользователя, освоить на практике работу с различными объектами для создания GUI, менеджерами размещения компонентов.

Теоретическое введение

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

Компонент 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

Пример создания объекта класса JTextField:

```
JTextField jta = new JTextField (10);
```

В параметре конструктора задано число 10, это количество символов, которые могут быть видны в текстовом поле. Текст введенный в поле JText может быть возвращен с помощью метода getText(). Также в поле можно записать новое значение с помощью метода setText(String s).

Как и у других компонентов, мы можем изменять цвет и шрифт текста в текстовом поле.

Задание 1-3

Условие задачи:

1. 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.
2. Создать окно, отобразить в нем картинку, путь к которой указан в аргументах командной строки.
3. Создать окно, реализовать анимацию, с помощью картинки, состоящей из нескольких кадров.

Решение:

Решение данной задачи представлено в листинге 1 – JFrame

Листинг 1 - JFrame

```
import javax.swing.*;
import java.awt.*;
import java.io.File;
import java.util.Objects;

// App class
class MyApp extends JFrame {
    int WINDOW_WIDTH = 600;
    int WINDOW_HEIGHT = 600;
    String background_image_path;
    String animation_images_path = "../frames/";
    int method;

    // Init
    MyApp() {
        super("Some shapes");
        setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
        //setBackground(Color.BLACK);
        setLocation(300, 300);
        setLayout(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
        this.method = 0;

        // Btn "Start animation"
        Button btn = new Button("Start animation");
        btn.setSize(200, 100);
        btn.setLocation(0, 0);
        btn.addActionListener(
            event -> {
                method = 1;
                setTitle("Animation");
                setSize(WINDOW_WIDTH + 1, WINDOW_HEIGHT);
                setSize(WINDOW_WIDTH - 1, WINDOW_HEIGHT);
            }
        );
        add(btn);
    }

    void set_background_image_path(String path) {
        this.background_image_path = path;
    }

    void set_animation_images_path(String path) {
        this.animation_images_path = path;
    }
}
```

Продолжение листинга 1.

```
// Paint method
@Override
public void paint(Graphics g2) {
    Graphics2D g = (Graphics2D) g2;
    Image img = Toolkit.getDefaultToolkit().getImage(this.background_image_path);
    g.drawImage(img, 0, 0, WINDOW_WIDTH, WINDOW_HEIGHT, this);

    switch (this.method) {
        case 0 -> random_shapes_paint(g);
        case 1 -> animate(this.animation_images_path, g);
        default -> {
        }
    }
}

// Random shapes paint method
void random_shapes_paint(Graphics2D g) {
    for (int i = 0; i < 50; i++) {
        int choice = (int) (Math.random() * 7);

        switch (choice) {

            // Oval
            case 1 -> {
                g.setColor(new Color((int) (Math.random() * 0x1000000)));
                g.fillOval(
                    (int) (Math.random() * WINDOW_WIDTH),
                    (int) (Math.random() * WINDOW_HEIGHT),
                    (int) (Math.random() * 200),
                    (int) (Math.random() * 200)
                );
            }

            // Rect
            case 2 -> {
                g.setColor(new Color((int) (Math.random() * 0x1000000)));
                g.fillRect(
                    (int) (Math.random() * WINDOW_WIDTH),
                    (int) (Math.random() * WINDOW_HEIGHT),
                    (int) (Math.random() * 200),
                    (int) (Math.random() * 200)
                );
            }
        }
    }
}
```

```
// Line
case 3 -> {
    g.setColor(new Color((int) (Math.random() * 0x1000000)));
    g.drawLine(
        (int) (Math.random() * WINDOW_WIDTH),
        (int) (Math.random() * WINDOW_WIDTH),
        (int) (Math.random() * WINDOW_HEIGHT),
        (int) (Math.random() * WINDOW_HEIGHT)
    );
}

// Arc
case 4 -> {
    g.setColor(new Color((int) (Math.random() * 0x1000000)));
    g.drawArc(
        (int) (Math.random() * WINDOW_WIDTH),
        (int) (Math.random() * WINDOW_HEIGHT),
        (int) (Math.random() * WINDOW_WIDTH),
        (int) (Math.random() * WINDOW_HEIGHT),
        (int) (Math.random() * 360),
        (int) (Math.random() * 360)
    );
}

// Rounded rect
case 5 -> {
    g.setColor(new Color((int) (Math.random() * 0x1000000)));
    g.fillRoundRect(
        (int) (Math.random() * WINDOW_WIDTH),
        (int) (Math.random() * WINDOW_HEIGHT),
        (int) (Math.random() * 200),
        (int) (Math.random() * 200),
        (int) (Math.random() * 200),
        (int) (Math.random() * 200)
    );
}

// Circle
case 6 -> {
    g.setColor(new Color((int) (Math.random() * 0x1000000)));
    g.fillOval(
        (int) (Math.random() * WINDOW_WIDTH),
        (int) (Math.random() * WINDOW_HEIGHT),
        100,
        100
    );
}
default -> {
}
}
```

Продолжение листинга 1.

```
void animate(String frames_path, Graphics2D g) {
    File dir = new File(frames_path);
    for (File file : Objects.requireNonNull(dir.listFiles())) {
        Image frame = Toolkit.getDefaultToolkit().getImage(file.getPath());
        g.drawImage(frame, 0, 0, WINDOW_WIDTH, WINDOW_HEIGHT, this);
        try {
            Thread.sleep(500);
        } catch (InterruptedException ignored) {
        }
    }
}

// Start app
public static void main(String[] args) {
    MyApp app = new MyApp();
    if (args.length == 1) {
        app.set_background_image_path(args[0]);
    } else if (args.length == 2) {
        app.set_background_image_path(args[0]);
        app.set_animation_images_path(args[1]);
    } else {
        System.out.println("Background image path is empty");
    }
}
}
```

Результат выполнения программы:

Результат выполнения программы представлен на рисунке 1 – JFrame

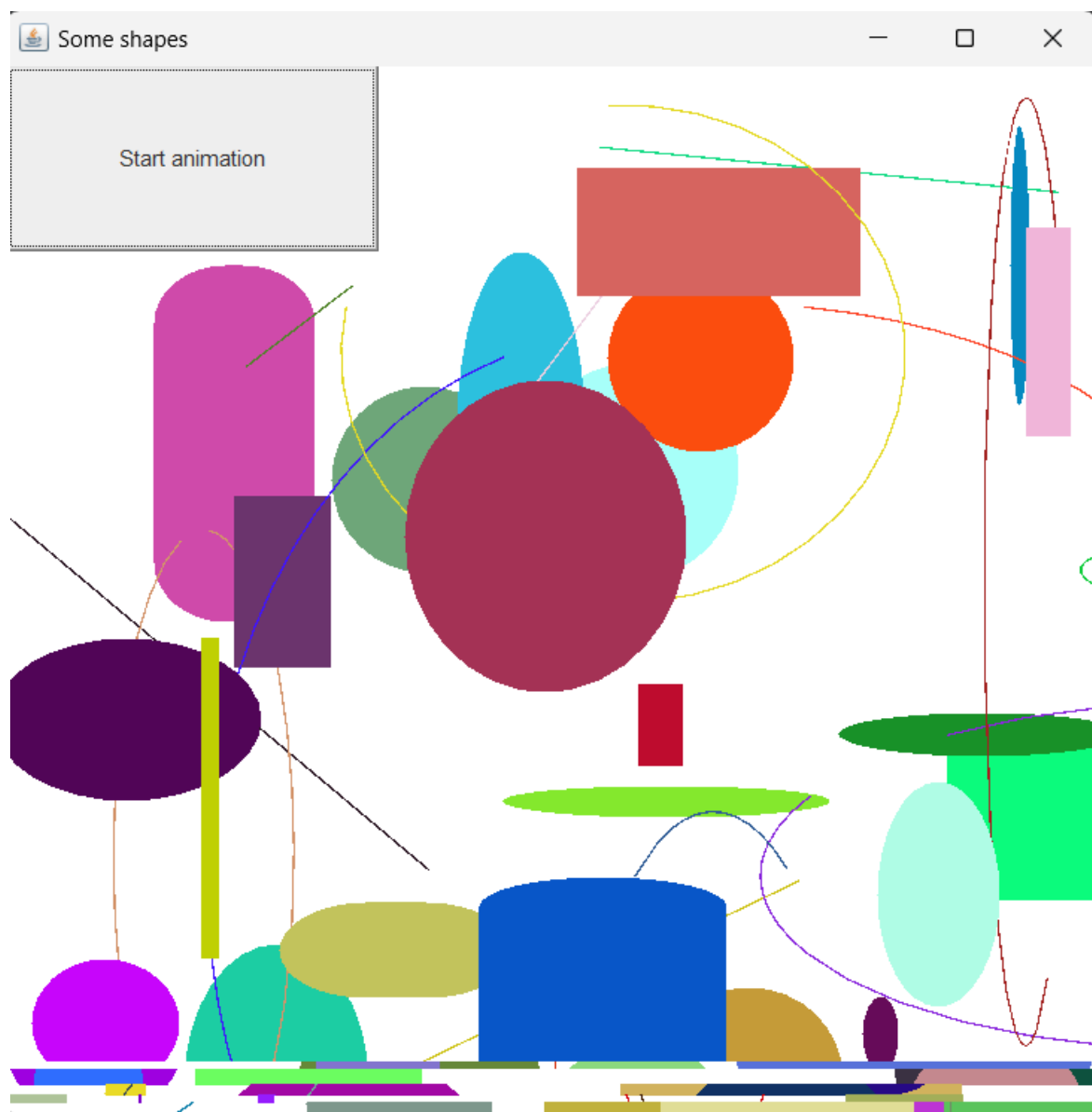


Рисунок 1 – JFrame

Вывод

Научились создавать графический интерфейс пользователя, освоили на практике работу с различными объектами для создания GUI, менеджерами размещения компонентов.

Практическая работа №13

Цель работы:

Закрепить знания в области обработки строк, научиться применять методы класса `String` и других классов для обработки строк.

Теоретическое введение

Для работы с текстовыми данными в Джава есть три класса: String, StringBuffer и StringBuilder

Особенности использования строк в Джава

В Джава строки представляют собой неизменяемую последовательность символов Unicode. В отличие от представления в C / C ++, где строка является просто массивом типа char, любая Java, строка является объектом класса java.lang.

Однако Джава строка, представляет собой в отличие от других используемых классов особый класс, который обладает довольно специфичными характеристиками. Отличия класса строк от обычных классов:

строка в Джава представляет из себя строку литералов (текст), помещенных в двойные кавычки, например:

"Hello, World! ". Вы можете присвоить последовательность строковых литералов непосредственно переменной типа String, вместо того чтобы вызывать конструктор для создания экземпляра класса String

Оператор '+' является перегруженным, для объектов типа String, и всегда используется, чтобы объединить две строки операндов. В данном контексте мы говорим об операции конкатенации или сложения строк. Хотя '+' не работает как оператор сложения для любых других объектов, кроме строк, например, таких как Point и Circle.

Строка является неизменяемой, то есть, символьной константой. Это значит, что ее содержание не может быть изменено после ее (строки как объекта) создания. Например, метод toUpperCase () – преобразования к верхнему регистру создает и возвращает новую строку вместо изменения содержания существующей строки.

Задание №1

Условие задачи:

Вырезать строку Java с помощью метода `String.substring()`.

Решение:

Решение задачи представлено в листинге 1 – Scanner

Листинг 1 - Scanner

```
import java.util.Scanner;

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

        System.out.println("Enter you string: ");
        String string = in.nextLine();

        System.out.println("Enter first index: ");
        int index_1 = in.nextInt();

        System.out.println("Enter second index: ");
        int index_2 = in.nextInt();

        System.out.println("Final string: ");
        System.out.println(string.substring(index_1, index_2));

    }
}
```

Результат выполнения программы

Результат выполнения программы представлен на рисунке 1 - String.

```
Enter you string:  
Hello world!  
Enter first index:  
2  
Enter second index:  
5  
Final string:  
llo
```

Рисунок 1 – String

Задание №2

Условие задачи:

Разработать класс Person, в котором имеется функция, возвращающая Фамилию И.О. Функция должна учитывать возможность отсутствия значений в полях Имя и Отчество. Программу оптимизировать с точки зрения быстродействия.

Решение:

Решение задачи представлено на листинге 2 – Person

Листинг 2 – Person

```
import java.util.Objects;

public class Person {
    private String firstName = "";
    private String lastName = "";
    private String maleName = "";

    public void setFirstName(String first_name) {
        if (!Objects.equals(first_name, "")) {
            this.firstName = first_name;
        } else {
            this.firstName = " ";
        }
    }

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

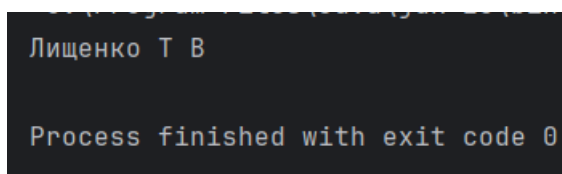
    public void setMaleName(String male_name) {
        if (!Objects.equals(male_name, "")) {
            this.maleName = male_name;
        } else {
            this.maleName = " ";
        }
    }

    public String getInfo() {
        return lastName + " " + firstName.charAt(0) + " " + maleName.charAt(0);
    }

    public static void main(String[] args) {
        Person person = new Person();
        person.setLastName("Jumba");
        person.setFirstName("");
        person.setMaleName("");
        System.out.println(person.getInfo());
    }
}
```


Результат выполнения программы

Результат выполнения программы представлен на рисунке 2 – Person



```
Лиценко Т В  
  
Process finished with exit code 0
```

Рисунок 2 – Person

Вывод

Закрепили знания в области обработки строк, научиться применять методы класса `String` и других классов для обработки строк.

Практическая работа №14

Цель работы:

Понять особенности использования регулярных выражений в Java, научиться работать с строками и применять регулярные выражения для обработки строк в программах.

Теоретическое введение

Регулярные выражения – это система обработки текста, основанная на специальной системе записи образцов для поиска. Образец (pattern), задающий правило поиска, по-русски также иногда называют «шаблоном», «маской». Сейчас регулярные выражения используются многими текстовыми редакторами и утилитами для поиска и изменения текста на основе выбранных правил. Язык программирования Java также поддерживает регулярные выражения для работы со строками.

Основными классами для работы с регулярными выражения являются класс `java.util.regex.Pattern` и класс `java.util.regex.Matcher`.

Для определения шаблона применяются специальные синтаксические конструкции.

Задание №5

Условие задачи:

Написать регулярное выражение, определяющее является ли данная строка датой в формате dd/mm/yyyy. Начиная с 1900 года до 9999 года.

1. Пример правильных выражений: 29/02/2000, 30/04/2003, 01/01/2003.
2. Пример неправильных выражений: 29/02/2001, 30-04-2003, 1/1/1899.

Решение

Решение задачи представлено на листинге 1 – Regular.

Листинг 1 - Regular

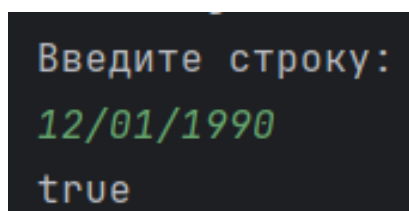
```
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class regular {
    public boolean regex(String input) {
        Pattern regular = Pattern.compile("[0-3][0-9]/[0-1][0-9]/[0-9][0-9][0-9][0-9]");
        Matcher m = regular.matcher(input);
        return m.matches();
    }

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Введите строку: ");
        String s2 = in.nextLine();
        regular reg = new regular();
        System.out.println(reg.regex(s2));
    }
}
```

Результат выполнения программы

Результат выполнения программы представлен на рисунке 1 – Regax.



```
Введите строку:  
12/01/1990  
true
```

Рисунок 1 – Regax

Вывод

Поняли особенности использования регулярных выражений в Java, научиться работать с строками и применять регулярные выражения для обработки строк в программах.

Практическая работа №15

Цель работы:

Изучить использование анонимных и внутренних классов, научиться разрабатывать интерактивные программы на языке Джава с использованием графического интерфейса пользователя.

Теоретическое введение

В Java вложенные классы — это классы, которые определены внутри другого класса. Цель вложенного класса - четко сгруппировать вложенный класс с окружающим его классом, сигнализируя, что эти два класса должны использоваться вместе. Или, возможно, вложенный класс должен использоваться только изнутри его включающего (владеющего) класса. Разработчики на Джава часто ссылаются на вложенные классы в качестве внутренних классов, но внутренние классы (нестатические вложенные классы) только один из нескольких различных типов вложенных классов в Джава. В Джава вложенные классы считаются компонентами своего включающего класса. Таким образом, вложенный класс не может быть объявлен `public`, `package` (без модификатора доступа), `protected` и `private` (см. модификаторов доступа для получения дополнительной информации). Следовательно, вложенные классы в Джава также могут наследоваться подклассами. В Джава можно создать несколько различных типов вложенных классов.

Рассмотрим различные типы вложенных классов Java:

1. Статические вложенные классы
2. Нестатические вложенные классы
3. Местные классы
4. Анонимные классы

Задача №1

Условие задачи

Напишите программу калькулятор, используя пример в листинге 15.6. Реализуйте помимо сложения вычитание, деление и умножение для двух чисел, которые вводятся с клавиатуры.

Решение

Решение задачи представлено в листинге 1 – Calculate

Листинг 1 - Calculate

```
import javax.swing.*;

class Calculator extends JFrame {
    JTextField jta1 = new JTextField(10);
    JTextField jta2 = new JTextField(10);
    JButton plus = new JButton(" Plus");
    JButton minus = new JButton(" Minus");
    JButton multiply = new JButton(" Multiply");
    JButton div = new JButton(" Division");

    Calculator() {
        super("Example");

        setLayout(new BoxLayout(getContentPane(), BoxLayout.Y_AXIS));
        setSize(150, 250);

        add(new JLabel("1st Number"));
        add(jta1);
        add(new JLabel("2nd Number"));
        add(jta2);

        add(plus);
        add(minus);
        add(multiply);
        add(div);

        plus.addActionListener(ae -> {
            try {
                double x1 = Double.parseDouble(jta1.getText().trim());
                double x2 = Double.parseDouble(jta2.getText().trim());

                JOptionPane.showMessageDialog(null, "Result = " + (x1 + x2), "Alert",
JOptionPane.INFORMATION_MESSAGE);
            } catch (Exception e) {

                JOptionPane.showMessageDialog(null, "Error in Numbers", "alert",
JOptionPane.ERROR_MESSAGE);
            }
        });

        minus.addActionListener(ae -> {
            try {
                double x1 = Double.parseDouble(jta1.getText().trim());
                double x2 = Double.parseDouble(jta2.getText().trim());
```

```
        minus.addActionListener(ae -> {
            try {
                double x1 = Double.parseDouble(jta1.getText().trim());
                double x2 = Double.parseDouble(jta2.getText().trim());

                JOptionPane.showMessageDialog(null, "Result = " + (x1 - x2), "Alert",
                JOptionPane.INFORMATION_MESSAGE);
            } catch (Exception e) {

                JOptionPane.showMessageDialog(null, "Error in Numbers", "alert",
                JOptionPane.ERROR_MESSAGE);
            }
        });

        multiply.addActionListener(ae -> {
            try {
                double x1 = Double.parseDouble(jta1.getText().trim());
                double x2 = Double.parseDouble(jta2.getText().trim());

                JOptionPane.showMessageDialog(null, "Result = " + (x1 * x2), "Alert",
                JOptionPane.INFORMATION_MESSAGE);
            } catch (Exception e) {

                JOptionPane.showMessageDialog(null, "Error in Numbers", "alert",
                JOptionPane.ERROR_MESSAGE);
            }
        });

        div.addActionListener(ae -> {
            try {
                double x1 = Double.parseDouble(jta1.getText().trim());
                double x2 = Double.parseDouble(jta2.getText().trim());
                if (x2 != 0) {
                    JOptionPane.showMessageDialog(null, "Result = " + (x1 / x2), "Alert",
                    JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(null, "Result = " + "division by zero", "Alert",
                    JOptionPane.INFORMATION_MESSAGE);
                }
            } catch (Exception e) {

                JOptionPane.showMessageDialog(null, "Error in Numbers", "alert",
                JOptionPane.ERROR_MESSAGE);
            }
        });
        setVisible(true);
    }

    public static void main(String[] args) {
        new Calculator();
    }
}
```

Результат выполнения программы

Результат выполнение программы представлен на рисунке 1 – Calc

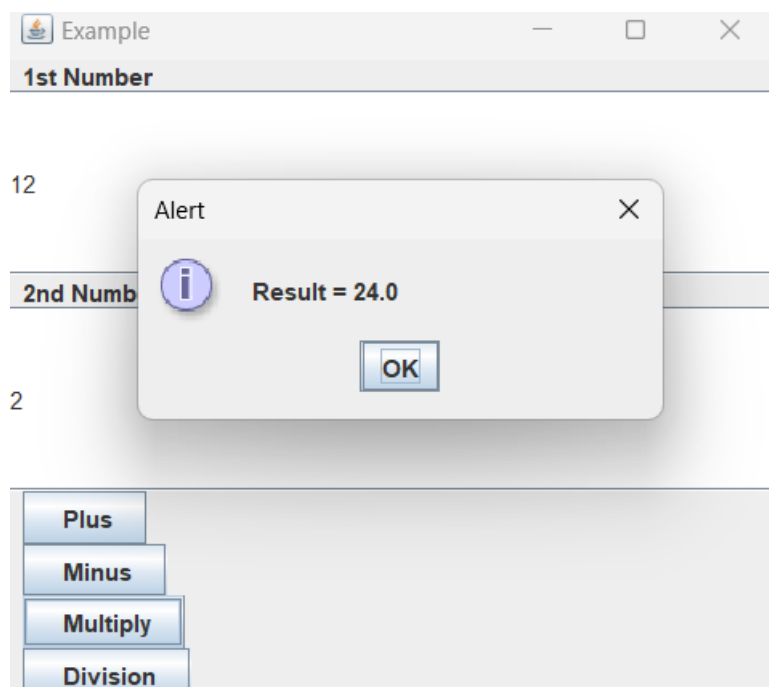


Рисунок 1 – Calc

Вывод

Изучили использование анонимных и внутренних классов, научиться разрабатывать интерактивные программы на языке Джава с использованием графического интерфейса пользователя.

Практическая работа №16

Цель работы

Научиться обрабатывать различные события мыши и клавиатуры для разных компонентов

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.

Задание №1

Условие задачи:

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. Реализация приложения Java, который имеет макет границы и надписи для каждой области в макете.

Решение

Решение задачи представлено в листинге 1 – Guessing

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

public class Guessing extends JFrame {
    JTextField jta = new JTextField(10);
    JButton button = new JButton("Guess");
    int trys = 3;

    Guessing() {
        super("Guessing");

        Random rand = new Random();
        int num = rand.nextInt(21);
        setLayout(new FlowLayout());
        setSize(500, 250);

        add(new JLabel("I made a number from 0 to 20, try to guess"));

        add(jta);
        add(button);

        button.addActionListener(ae -> {
            try {
                int x1 = Integer.parseInt(jta.getText().trim());
                if (trys > 0) {
                    trys--;
                    if (x1 != num) {
                        if (x1 > num) {
                            JOptionPane.showMessageDialog(null, "You did not guess, the hidden
number is less than the entered one. There are still attempts left: " + trys, "Result",
JOptionPane.INFORMATION_MESSAGE);
                        } else {
                            JOptionPane.showMessageDialog(null, "You did not guess, the hidden
number is greater than the entered one. There are still attempts left: " + trys, "Result",
JOptionPane.INFORMATION_MESSAGE);
                        }
                    } else {
                        JOptionPane.showMessageDialog(null, "You guessed it! Congratulations! Let's
try again", "Result", JOptionPane.INFORMATION_MESSAGE);
                        dispose();
                        new Guessing();
                    }
                } else {

```

Продолжение листинга 1 – Guessing

```
JOptionPane.showMessageDialog(null, "You have not guessed the number and you have no
attempts left. I'm sorry, but you lost. Let's start over", "Result",
JOptionPane.INFORMATION_MESSAGE);
        dispose();
        new Guessing();
    }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error in Number", "Error",
JOptionPane.ERROR_MESSAGE);
    }
    });
    setVisible(true);
}

public static void main(String[] args) {
    new Guessing();
}
}
```

Результат выполнения программы

Результат выполнения программы представлен на рисунке 1 – Guessing

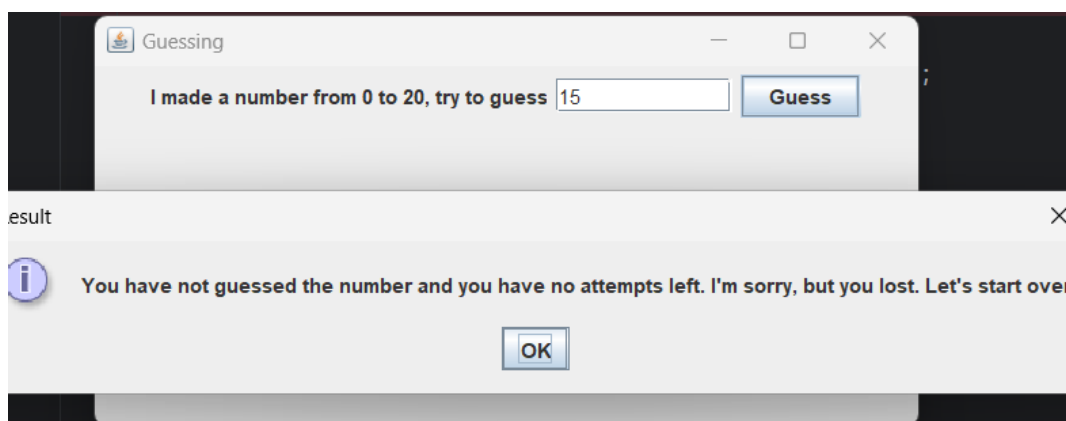


Рисунок 1 – Guessing

Вывод

Научились обрабатывать различные события мыши и клавиатуры для разных компонентов.

Практическая работа №17

Цель работы

Введение в разработку программ с использованием событийного программирования на языке программирования Джава с использованием паттерна MVC.

Теоретическое введение

Шаблон проектирования в программной инженерии — это метод решения часто возникающей проблемы при разработке программного обеспечения. Проектирование по модели указывает, какой тип архитектуры вы используете для решения проблемы или разработки модели

Проекты моделей, основанные на архитектуре MVC (model-viewcontroller), следуют шаблону проектирования MVC и отделяют логику приложения от пользовательского интерфейса при разработке программного обеспечения.

В контексте программирования модель на Java состоит из простых классов Java, уровень представление отображает данные — то что видит пользователь, а контроллер состоит из сервлетов.

Задание №3

Условие задачи

Вы можете написать программную реализацию, используя собственную идею, реализуя паттерн MVC. Выполнение задания предполагает создание GUI.

Решение

Решение задачи представлено на листингах приведенных ниже:

Листинг 1 - Snake

```
package model;

import java.awt.Point;
import java.awt.event.KeyEvent;

public class Snake {
    private final Point[] pointBody = new Point[300];
    private int length;

    private boolean isLeft;
    private boolean isRight;
    private boolean isUp;
    private boolean isDown;

    private int directionX;
    private int directionY;

    public int[] x = new int[300];
    public int[] y = new int[300];

    public Snake() {
        isLeft = false;
        isRight = true;
        isUp = true;
        isDown = true;
        setDirectionX(10);
        setDirectionY(0);
        length = 3;
        y[0] = 100;
        x[0] = 150;
    }

    public void onMove(int side) {
        switch (side) {
            case KeyEvent.VK_LEFT -> {
                if (isLeft) {
                    setDirectionX(-10);
                    setDirectionY(0);
                    isRight = false;
                    isUp = true;
                    isDown = true;
                }
            }
        }
    }
}
```

```

        case KeyEvent.VK_UP -> {
            if (isUp) {
                setDirectionX(0);
                setDirectionY(-10);
                isDown = false;
                isRight = true;
                isLeft = true;
            }
        }
        case KeyEvent.VK_DOWN -> {
            if (isDown) {
                setDirectionX(0);
                setDirectionY(+10);
                isUp = false;
                isRight = true;
                isLeft = true;
            }
        }
        case KeyEvent.VK_RIGHT -> {
            if (isRight) {
                setDirectionX(+10);
                setDirectionY(0);
                isLeft = false;
                isUp = true;
                isDown = true;
            }
        }
        default -> {
        }
    }

    }

    public int getLength() {
        return length;
    }

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

    public Point[] getPointBody() {
        return pointBody;
    }

    public int getDirectionX() {
        return directionX;
    }

```

Продолжение листинга 1 – Snake

```
public void setDirectionX(int directionX) {  
    this.directionX = directionX;  
}  
  
public int getDirectionY() {  
    return directionY;  
}  
  
public void setDirectionY(int directionY) {  
    this.directionY = directionY;  
}  
}
```

Листинг 2 – Fruit

```
package model;  
  
import java.awt.Point;  
  
public class Fruit {  
    private Point point;  
    private boolean food;  
  
    public Fruit() {  
        point = new Point();  
    }  
  
    public Point getPoint() {  
        return point;  
    }  
  
    public void setPoint(Point point) {  
        this.point = point;  
    }  
  
    public boolean isFood() {  
        return food;  
    }  
  
    public void setFood(boolean food) {  
        this.food = food;  
    }  
}
```

Листинг 3 – Controller

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.Timer;

import model.Snake;

public class Controller implements KeyListener, ActionListener {
    private final Renderer renderer;
    private final Timer mainTimer;
    private final Snake snake;

    public Controller() {
        snake = new Snake();
        renderer = new Renderer(this);

        this.renderer.addKeyListener(this);
        this.mainTimer = new Timer(50, this);
        mainTimer.start();
    }

    public void stopGame() {
        mainTimer.stop();
    }

    public void startGame() {
        mainTimer.start();
    }

    public void keyPressed(KeyEvent e) {
        snake.onMove(e.getKeyCode());
    }

    public void keyReleased(KeyEvent e) {
    }

    public void keyTyped(KeyEvent e) {
    }

    @Override
    public void actionPerformed(ActionEvent arg0) {
        renderer.moveForward();
    }

    public Snake getSnake() {
        return snake;
    }
}
```

Листинг 4 – Demo

```
public class Demo {  
    public static void main(String[] args) {  
        new Controller();  
    }  
}
```

Листинг 5 – Render

```
import java.awt.Color;
import java.awt.GridLayout;
import java.util.Random;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;

import model.Snake;
import model.Fruit;

class Renderer extends JFrame {
    private final int WIDTH = 500;
    private final int HEIGHT = 300;
    private final Snake snake;

    private final Controller controller;
    public JPanel panel1, panel2;
    public JButton[] ButtonBody = new JButton[200];
    public JButton bonusfood;
    public JTextArea textArea;
    public Fruit fruit;
    public int score;
    public Random random = new Random();
    public JMenuBar mybar;
    public JMenu game, help;

    public void initializeValues() {
        score = 0;
    }

    @SuppressWarnings("deprecation")
    Renderer(Controller controller) {
        super("Snake: PSU edition");
        this.controller = controller;
        snake = controller.getSnake();

        fruit = new Fruit();
        setBounds(200, 200, 506, 380);
        creatbar();
        initializeValues();
        // GUI панельки
        panel1 = new JPanel();
        panel2 = new JPanel();
    }
}
```

```

// Текстовое поле очков
setResizable(false);
textArea = new JTextArea("Scores : " + score);
textArea.setEnabled(false);
textArea.setBounds(400, 400, 100, 100);
textArea.setBackground(Color.black);
// Змея ест и растет
bonusfood = new JButton();
bonusfood.setEnabled(false);
// Создаем начальную 3-х кнопочную змею
createFirstSnake();

pamel1.setLayout(null);
panel2.setLayout(new GridLayout(0, 1));
pamel1.setBounds(0, 0, WIDTH, HEIGHT);
pamel1.setBackground(Color.blue);
panel2.setBounds(0, HEIGHT, WIDTH, 30);
panel2.setBackground(Color.black);

panel2.add(textArea); // will contain score board
// end of UI design
getContentPane().setLayout(null);
getContentPane().add(pamel1);
getContentPane().add(panel2);
show();
setDefaultCloseOperation(EXIT_ON_CLOSE);

}

// Создаем начальную змейку(из 3 кнопок)
public void createFirstSnake() {

    for (int i = 0; i < 3; i++) {
        ButtonBody[i] = new JButton(" " + i);
        ButtonBody[i].setEnabled(false);
        pamel1.add(ButtonBody[i]);
        ButtonBody[i].setBounds(snake.x[i], snake.y[i], 10, 10);
        snake.x[i + 1] = snake.x[i] - 10;
        snake.y[i + 1] = snake.y[i];
    }
}

// Создаем меню бар
public void creatbar() {
    mybar = new JMenuBar();

    game = new JMenu("Игра");

```


Продолжение листинга 5 – Render

```
JMenuItem newgame = new JMenuItem("Новая игра");
JMenuItem exit = new JMenuItem("Выход");
newgame.addActionListener(e -> reset());

exit.addActionListener(e -> System.exit(0));

game.add(newgame);
game.addSeparator();// Разделитель
game.add(exit);

mybar.add(game);

help = new JMenu("Помощь");

JMenuItem creator = new JMenuItem("Автор");

creator.addActionListener(e -> JOptionPane.showMessageDialog(pamel1, "Лищенко
Тимофей Викторович"));

help.add(creator);
mybar.add(help);

setJMenuBar(mybar);
}

// Начало новой игры
void reset() {
    initializeValues();
    pamel1.removeAll();

    controller.stopGame();

    createFirstSnake();
    textArea.setText("Scores: " + score);

    controller.startGame();
}

// кушаем растем
void growup() {
    ButtonBody[snake.getLength()] = new JButton(" " + snake.getLength());
    ButtonBody[snake.getLength()].setEnabled(false);
    pamel1.add(ButtonBody[snake.getLength()]);
    ButtonBody[snake.getLength()].setBounds(snake.getPointBody()[snake.getLength() -
1].x, snake.getPointBody()[snake.getLength() - 1].y, 10, 10);
    snake.setLength(snake.getLength() + 1);
}
```

```

void moveForward() {
    for (int i = 0; i < snake.getLength(); i++) {
        snake.getBodyPoint()[i] = ButtonBody[i].getLocation();
    }

    snake.x[0] += snake.getDirectionX();
    snake.y[0] += snake.getDirectionY();
    ButtonBody[0].setBounds(snake.x[0], snake.y[0], 10, 10);
    // Для эффекта летницы
    for (int i = 1; i < snake.getLength(); i++) {
        ButtonBody[i].setLocation(snake.getBodyPoint()[i - 1]);
    }

    if (snake.x[0] == WIDTH) {
        controller.stopGame();
    } else if (snake.x[0] == 0) {
        controller.stopGame();
    } else if (snake.y[0] == HEIGHT) {
        controller.stopGame();
    } else if (snake.y[0] == 0) {
        controller.stopGame();
    }
    createFruit();

    collisionFruit();
    panel1.repaint();
}

private void collisionFruit() {
    if (fruit.isFood()) {
        if (fruit.getBodyPoint().x == snake.x[0] && fruit.getBodyPoint().y == snake.y[0]) {
            panel1.remove(bonusfood);
            score += 1;
            growup();
            textArea.setText("Scores: " + score);
            fruit.setFood(false);
        }
    }
}

private void createFruit() {
    if (!fruit.isFood()) {
        panel1.add(bonusfood);
        bonusfood.setBounds((10 * random.nextInt(50)), (10 * random.nextInt(25)), 10,
            10);
        fruit.setPoint(bonusfood.getLocation());
        fruit.setFood(true);
    }
}
}

```

Результат выполнения программы

Результат выполнения программы представлен на рисунке 1 – Snake

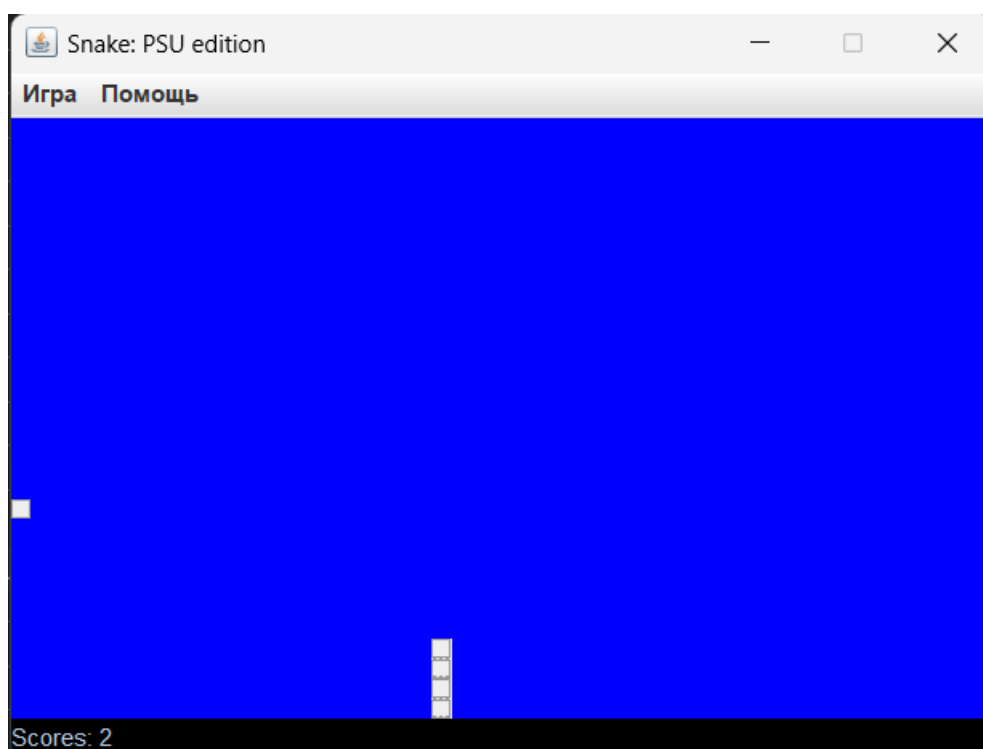


Рисунок 1 – Snake

Вывод

Программы с использованием событийного программирования на языке программирования Джава с использованием паттерна MVC.

Практическая работа №18

Цель работы:

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. Ключевые слова: try, catch, finally, throw, throws

Теоретическое введение

Механизм исключительных ситуаций в Java поддерживается пятью ключевыми словами:

1. Try
2. Catch
3. finally
4. throw
5. throws

В языке Джава Java всего около 50 ключевых слов, и пять из них связано как раз с исключениями, это- try, catch, finally, throw, throws. Из них catch, throw и throws применяются к экземплярам класса, причём работают они только с Throwable и его наследниками.

Задание №4

Условие задачи

Шаг 1. Добавьте блок `finally` к решению Задания №2.

Шаг 2. Повторно запустите программу, чтобы проверить ее поведение.

Объясните новое поведение программы Генерация собственных исключений. На предыдущем шаге при выполнении заданий мы рассмотрели, как Java работает с предопределенными исключениями, теперь перейдем к тому, как генерируется исключение. Все исключения в рассмотренных ранее примерах и заданиях были определены заранее. В этом разделе практической работы вы будете создавать и пробрасывать свои собственные исключения (exceptions).

Решение

Решение данной задачи представлено на листинге 1 – Exception

Листинг 1 - Exception

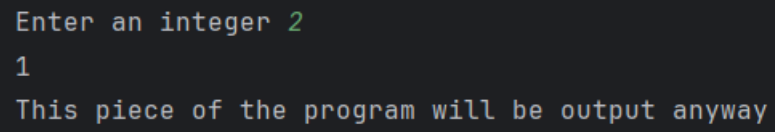
```
import java.util.Scanner;

public class Exception4 {
    public void exceptionDemo() {
        Scanner myScanner = new Scanner(System.in);
        System.out.print("Enter an integer ");
        String intString = myScanner.next();
        int i = 2;
        try {
            i = Integer.parseInt(intString);
            System.out.println(2 / i);
        } catch (ArithmeticException e) {
            System.out.println("Attempted division by zero");
        } catch (NumberFormatException e) {
            System.out.println("The program is waiting for the input of an integer value");
        } finally {
            System.out.println("This piece of the program will be output anyway");
        }
    }

    public static void main(String[] args) {
        Exception4 q = new Exception4();
        q.exceptionDemo();
    }
}
```


Результат выполнения программы

Результат выполнения программы представлен на рисунке 1 – Exception



```
Enter an integer 2
1
This piece of the program will be output anyway
```

Рисунок 1 – Exception

Вывод

Получили практические навыки разработки программ, изучили синтаксиса языка Java, освоили основные конструкции языка Java (циклы, условия, создание переменных и массивов, создание методов, вызов методов), а также научились осуществлять стандартный ввод/вывод данных.

Практическая работа №19

Цель работы

Научиться создавать собственные исключения.

Теоретическое введение

Язык Java предоставляет исчерпывающий набор классов исключений, но иногда при разработке программ вам потребуется создавать новые – свои собственные исключения, которые являются специфическими для потребностей именно вашего приложения. В этой практической работе вы научитесь создавать свои собственные пользовательские классы исключений. Как вы уже знаете, в Java есть два вида исключений- проверяемы и непроверяемые. Для начала рассмотрим создание пользовательских проверяемых исключений.

Задание №1

Условие задачи

Клиент совершает покупку онлайн. При оформлении заказа у пользователя запрашивается фио и номер ИНН. В программе проверяется, действителен ли номер ИНН для такого клиента. Исключение будет выдано в том случае, если введен недействительный ИНН.

Решение

Решение данной задачи представлено на листинге 1 – INN

Листинг 1 - INN

```
import java.util.Scanner;

class INN extends Exception {
    public INN(String message) {
        super(message);
    }

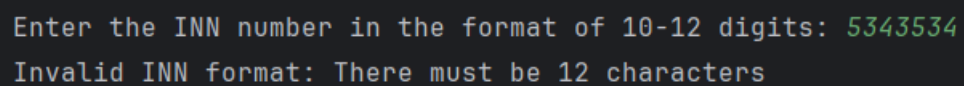
    public static void isValid(String inn) throws INN {
        if (inn.length() != 10 && inn.length() != 12) {
            throw new INN("There must be 12 characters");
        }
        try {
            Long.parseLong(inn);
        } catch (NumberFormatException e) {
            throw new INN("Not a number");
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the INN number in the format of 10-12 digits: ");
        String inn = scanner.next();

        try {
            isValid(inn);
        } catch (INN e) {
            System.out.println("Invalid INN format: " + e.getMessage());
        }
    }
}
```

Результат выполнения программы

Результат выполнения программы представлен на рисунке 1 – INN



```
Enter the INN number in the format of 10-12 digits: 5343534  
Invalid INN format: There must be 12 characters
```

Рисунок 1 – INN

Вывод

Научились создавать собственные исключения.

Практическая работа №20

Цель работы

Научиться работать с обобщенными типами в Java и применять их в программах

Теоретическое введение

Введение в Дженерики. В JDK представлены дженерики (перевод с англ. *generics*), которые поддерживают абстрагирование по типам (или параметризованным типам). В объявлении классов дженерики представлены обобщенными типами, в то время как пользователи классов могут быть конкретными типами, например во время создания объекта или вызова метода. Вы, конечно, знакомы с передачей аргументов при вызове методов в языке C++. Когда вы передаете аргументы внутри круглых скобок () в момент вызова метода, то аргументы подставляются вместо формальных параметров, с которыми объявлен и описан метод. Схожим образом в *generics* вместо передаваемых аргументов мы передаем информацию только о типах аргументов внутри угловых скобок <> (так называемая *diamond notation* или алмазная запись). Основное назначение использования дженериков — это абстракция работы над типами при работе с коллекциями («Java Collection Framework»).

Задание №4

Условие задачи

Написать класс `Matrix`, на основе обобщенного типа, реализовать операции с матрицами

Решение

Решение данной задачи представлено на листинге 1 – Matrix.

Листинг 1.

```
public class Matrix {

    private final int rows;
    private final int columns;
    private final int[][] matrix;

    /***** Constructors *****/
    public Matrix() {
        this.rows = 1;
        this.columns = 1;
        this.matrix = new int[rows][columns];
    }

    public Matrix(int rows, int columns) throws MatrixException {

        if(rows <= 0 || columns <= 0) {
            throw new MatrixException("Заданы неверные размеры матрицы");
        }

        this.rows = rows;
        this.columns = columns;
        this.matrix = new int[rows][columns];
    }

    public Matrix(int size) throws MatrixException {
        if(size <= 0) {
            throw new MatrixException("Заданы неверные размеры матрицы");
        }

        this.rows = this.columns = size;
        this.matrix = new int[size][size];
    }

    public Matrix(int[][] matrix) {
        this.rows = matrix.length;
        this.columns = matrix[0].length;
        this.matrix = matrix;
    }

    /*****/
}
```

Продолжение листинга 1 – Matrix

```
public static Matrix sum(Matrix matrix1, Matrix matrix2) throws MatrixException {
    if (matrix1.rows != matrix2.rows || matrix1.columns != matrix2.columns) {
        throw new MatrixException("Размеры матриц не совпадают");
    } else {
        Matrix matrix = new Matrix(matrix1.rows, matrix1.columns);
        for(int i = 0; i < matrix1.rows; i++) {
            for(int j = 0; j < matrix1.columns; j++) {
                matrix.matrix[i][j] = matrix1.matrix[i][j] + matrix2.matrix[i][j];
            }
        }
        return matrix;
    }
}

public static final Matrix multiply(Matrix matrix1, Matrix matrix2) throws MatrixException
{
    if(!check(matrix1, matrix2, true)) {
        throw new MatrixException("Размеры матриц не совпадают");
    }
    else {

        int size;
        int n;

        if(!matrix1.checkSquare()) {
            size = Math.min(matrix1.rows, matrix1.columns);
            n = Math.max(matrix1.rows, matrix1.columns);
        }
        else {
            size = matrix1.rows;
            n = matrix1.rows;
        }

        Matrix matrix = new Matrix(size);
        matrix.fill(0);

        for(int i = 0; i < size; i++) {
            for(int j = 0; j < size; j++) {
                for(int k = 0; k < n; k++) {
                    matrix.matrix[i][j] += matrix1.matrix[i][k]*matrix2.matrix[k][j];
                }
            }
        }

        return matrix;
    }
}
```

```
}

public final void print() {

    for(int i = 0; i < rows; i++) {
        for(int j = 0; j < columns; j++) {
            System.out.print(matrix[i][j] + " ");
        }
        System.out.println();
    }
}

public final void fill(int number) {

    for(int i = 0; i < rows; i++) {
        for(int j = 0; j < columns; j++) {
            matrix[i][j] = number;
        }
    }
}

private final static boolean check(Matrix matrix1, Matrix matrix2, boolean forMultiply) {
    final boolean b = (matrix1.rows == matrix2.rows) && (matrix1.columns ==
matrix2.columns);
    if(forMultiply) {
        return b ||
            (matrix1.columns == matrix2.rows);
    }
    else {
        return b;
    }
}

public final boolean checkSquare() {
    return rows == columns;
}
}
```

Вывод

Научились работать с обобщенными типами в Java и применять их в программах.

Практическая работа №21

Цель работы

Работать с обобщенными типами в Java и применять их в программах.

Теоретическое введение

Из предыдущего примера может создаться видимость того, что компилятор заменяет параметризованный тип актуальным или фактическим типом (таким как `String`, `Integer`) во время создания экземпляра объекта типа класс. Если это так, то компилятору необходимо будет создавать новый класс для каждого актуального или фактического типа (аналогично шаблону `C++`). На самом же деле происходит следующее - компилятор заменяет всю ссылку на параметризованный тип `E` на ссылку на `Object`, выполняет проверку типа и вставляет требуемые операторы, обеспечивающие понижающее приведения типов.

Задание №2

Условие задачи

Написать класс, который умеет хранить в себе массив любых типов данных (int, long etc.).

Решение

Решение данной задачи представлено на листинге 1 – AnyType

Листинг 1 - AnyType

```
public class Main {

    public static <E> void sid(String s, E[] arr) {

        E[] a = arr;
        pr21_2_AnyTape<E> sid = new pr21_2_AnyTape<E>();
        sid.setArr(a);

    }

    public static void main(String[] args) {

        String[] s = {"Hello", "World", "!"};

        Integer[] intr = new Integer[]{1, 2, 3, 4, 5, 6, 7, 8};

        Double[] ad = {1.2, 1.5, 6.7};

        intr[0] = 5;

        pr21_2_AnyTape<String> s1 = new pr21_2_AnyTape<String>();
        pr21_2_AnyTape<Integer> s2 = new pr21_2_AnyTape<Integer>();
        pr21_2_AnyTape<Double> s3 = new pr21_2_AnyTape<Double>();

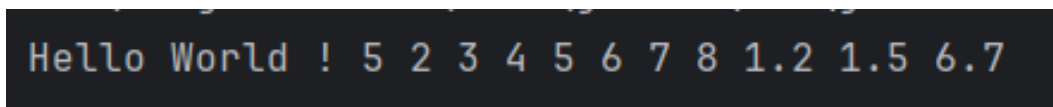
        s1.setArr(s);
        s1.printer();
        s2.setArr(intr);
        s2.printer();
        s3.setArr(ad);
        s3.printer();

    }

}
```

Результат выполнения программы

Результат выполнения программы представлен на рисунке 1 – AnyType



```
Hello World ! 5 2 3 4 5 6 7 8 1.2 1.5 6.7
```

Рисунок 1 – AnyType

Вывод

Научиться работать с обобщенными типами в Java и применять прием стирание типов разработке программ на Джаве.

Практическая работа №22

Цель работы

Научиться разрабатывать программы с абстрактными типами данных на языке Джава и применять паттерн MVC при разработке программ.

Теоретическое введение

Стек — это линейная структура данных, которая следует принципу LIFO (Last In First Out) . Стек имеет один конец, куда мы можем добавлять элементы и извлекать их оттуда, в отличие от очереди, которая имеет два конца (спереди и сзади). Стек содержит только один указатель `top`(верхушка стека), указывающий на самый верхний элемент стека. Всякий раз, когда элемент добавляется в стек, он добавляется на вершину стека, и этот элемент может быть удален только из стека только сверху. Другими словами, стек можно определить как контейнер, в котором вставка и удаление элементов могут выполняться с одного конца, известного как вершина стека.

Примеры стеков – пирамида, стопка тарелок или книг, магазин в пистолете

Стеку присущи следующие характеристики:

Стек — это абстрактный тип данных с заранее определенной емкостью, что означает, что эта структура данных имеет ограниченный размер, то есть может хранить количество элементов, определенное размерностью стека

Это структура данных, в которой строго определен порядок вставки и удаления элементов, и этот порядок может быть LIFO или FILO.

Задание №1

Условие задачи

Напишите программу-калькулятор арифметических выражений, записанных в обратной польской нотации (RPN-калькулятор).

Решение

Решение данной задачи представлено на листинге 1 – RPN

Листинг 1 – RPN

```
import java.util.*;

public class RPNCalc {
    private static final Stack<Integer> stack = new Stack<>();
    private static final Scanner input = new Scanner(System.in);

    public static void calculator() {

        System.out.println("Welcome to the RPN Calculator program!");

        takeInput();
    }

    private static void takeInput() {
        String numOrOperand = " ";
        while (!numOrOperand.equals("x")) {
            System.out.println("Enter next input: ");
            numOrOperand = input.next();
            try {
                int intNumOrOperand = Integer.parseInt(numOrOperand);
                stack.push(intNumOrOperand);
            } catch (Exception e) {
                switch (numOrOperand) {
                    case "*" -> stack.push(stack.pop() * stack.pop());
                    case "/" -> stack.push((int) stack.pop() / stack.pop());
                    case "+" -> stack.push(stack.pop() + stack.pop());
                    case "-" -> stack.push(stack.pop() - stack.pop());
                    case "=" -> System.out.println(stack.pop());
                    case "c" -> {
                        if (!stack.empty()) {
                            for (int i = 0; i < stack.size(); i++) {
                                stack.pop();
                            }
                        }
                    }
                    case "w" -> {
                        for (Integer integer : stack) {
                            System.out.println(integer);
                        }
                    }
                }
            }
        }
    }
}
```

Продолжение листинга 1 – RPN

```
public static void main(String[] args) {  
    try {  
        calculator();  
    } catch (Exception e) {  
        System.out.println("Oops, that doesn't work... ");  
    }  
}  
}
```

Результат выполнения программы

Результат выполнения программы представлен на рисунке 1 – RPN

```
Welcome to the RPN Calculator program!  
Enter next input:  
2  
Enter next input:  
3  
Enter next input:  
+  
Enter next input:  
=  
5
```

Рисунок 1 – RPN

Вывод

Научились разрабатывать программы с абстрактными типами данных на языке Джава и применять паттерн MVC при разработке программ.

Практическая работа №23

Цель работы

Научиться разрабатывать программы с абстрактными типами данных на языке Джава

Теоретическое введение

1. Очередь можно определить как упорядоченный список, который позволяет выполнять операции вставки на одном конце, называемом REAR , и операции удаления, которые выполняются на другом конце, называемом FRONT
2. Очередь называется работает по дисциплине обслуживания «первый пришел — первый обслужен» (FCFS, first come first served)
3. Например, люди, стоящие в кассу магазина образуют очередь оплаты покупок.

Задание №1

Условие задачи

Найдите инвариант структуры данных «очередь». Определите функции, которые необходимы для реализации очереди. Найдите их пред- и постусловия

Реализуйте классы, представляющие циклическую очередь с применением массива

Класс `ArrayQueueModule` должен реализовывать один экземпляр очереди с использованием переменных класса

Класс `ArrayQueueADT` должен реализовывать очередь в виде абстрактного типа данных (с явной передачей ссылки на экземпляр очереди).

Класс `ArrayQueue` должен реализовывать очередь в виде класса (с неявной передачей ссылки на экземпляр очереди).

Должны быть реализованы следующие функции(процедуры)/методы:

`enqueue` – добавить элемент в очередь;

`element` – первый элемент в очереди;

`dequeue` – удалить и вернуть первый элемент в очереди;

`size` – текущий размер очереди;

`isEmpty` – является ли очередь пустой;

`clear` – удалить все элементы из очереди

Инвариант, пред- и постусловия записываются в исходном коде в виде комментариев.

Обратите внимание на инкапсуляцию данных и кода во всех трех реализациях.

Напишите тесты реализованным классам.

Решение

Решение задачи представлено на листингах ниже:

Листинг 1 – AbstractQueue

```
package Java_task_26.ex1;

public abstract class AbstractQueue implements Queue {
    protected int head, tail;
    protected static final int START_CAPACITY = 16;
}
```

Листинг 2 - ArrayQueue

```
package Java_task_26.ex1;

// INV: FIFO (First in - first out)
// 0 <= size
// queue[head]..queue[tail] - queue
public class ArrayQueue {
    private ArrayQueueModule queue;

    public ArrayQueue() {
        this.queue = ArrayQueueModule.getInstance();
    }

    // PRE: size > 0
    // POST: R = queue[head]
    // queue[head] = queue[head+1]
    // queue[head+1]..queue[tail] - immutable
    public Object dequeue() {
        return queue.dequeue();
    }

    // PRE: size > 0
    // POST: R = queue[head]
    // queue - immutable
    public Object element() {
        return queue.element();
    }

    // PRE: None
    // POST: queue[tail] = element
    // queue[head]..queue[tail-1] - immutable
    public void enqueue(Object o) {
        queue.enqueue(o);
    }
}
```


Продолжение листинга 2

```
// PRE: None
// POST: queue - immutable
//    R = (head == tale)
public boolean isEmpty() {
    return queue.isEmpty();
}

// PRE: None
// POST: size == 0
public boolean clear() {
    return queue.clear();
}
}
```

Листинг 3 – ArrayQueueADT

```
package Java_task_26.ex1;

// INV: FIFO (First in - first out)
//    0 <= size
//    queue[head]..queue[tail] - queue
public class ArrayQueueADT {

    private Java_task_26.ex1.ArrayQueueModule queue;

    public ArrayQueueADT(ArrayQueueModule queue) {
        this.queue = queue;
    }

    // PRE: size > 0
    // POST: R = queue[head]
    //    queue[head] = queue[head+1]
    //    queue[head+1]..queue[tail] - immutable
    public Object dequeue() {
        return queue.dequeue();
    }

    // PRE: size > 0
    // POST: R = queue[head]
    //    queue - immutable
    public Object element() {
        return queue.element();
    }
}
```

Продолжение листинга 3

```
// PRE: None
// POST: queue[tail] = element
//    queue[head]..queue[tail-1] - immutable
public void enqueue(Object o) {
    queue.enqueue(o);
}

// PRE: None
// POST: queue - immutable
//    R = (head == tale)
public boolean isEmpty() {
    return queue.isEmpty();
}

// PRE: None
// POST: size == 0
public boolean clear() {
    return queue.clear();
}
}
```

Листинг 4 - ArrayQueueModule

```
package Java_task_26.ex1;

// INV: FIFO (First in - first out)
//    0 <= size <= q.length - 1
//    queue[head]..queue[tail] - queue
//    queue - Singleton
//    q.length = 2**x
public class ArrayQueueModule extends AbstractQueue {
    private Object[] q;

    private static ArrayQueueModule instance;

    // PRE: queue - null
    // POST: queue.size = 0;
    //    q.length = 16;
    private ArrayQueueModule() {
        q = new Object[START_CAPACITY];
        head = tail = 0;
    }

    // Pre: none;
    // Post: instance - Singleton;
    //    queue - immutable
}
```

Продолжение листинга 4

```
public static ArrayQueueModule getInstance() {
    if (instance == null)
        instance = new ArrayQueueModule();
    return instance;
}

// PRE: size > 0
// POST: R = queue[head]
//     queue[head] = queue[head+1]
//     queue[head+1]..queue[tail] - immutable
@Override
public Object dequeue() {
    if (isEmpty()) throw new IndexOutOfBoundsException("Queue is empty!");
    Object r = q[head++];
    if (head == q.length) head = 0;
    if (Math.abs(head - tail) < q.length / 2) resize(q.length / 2);
    return r;
}

// PRE: None
// POST: queue - immutable
//     R = (head == tale)
@Override
public boolean isEmpty() {
    return head == tail;
}

// PRE: None
// POST: queue[tail] = element
//     queue[head]..queue[tail-1] - immutable
@Override
public void enqueue(Object element) {
    q[tail++] = element;
    if (tail == q.length) tail = 0;
    if (tail == head) resize(q.length * 2);
}

// Pre: none;
// Post: q.length == nSize;
//     queue - immutable;
private void resize(int nSize) {
    Object[] nq = new Object[nSize];
    System.arraycopy(q, head, nq, 0, Math.abs(head - tail));
    tail = Math.abs(head - tail);
    head = 0;
    q = nq;
}
```

Продолжение листинга 4

```
// PRE: size > 0
// POST: R = queue[head]
//    queue - immutable
@Override
public Object element() {
    if (isEmpty()) throw new IndexOutOfBoundsException("Queue is empty!");
    return q[head];
}

// PRE: None
// POST: size == 0
//    q.length = 16
@Override
public boolean clear() {
    boolean r = !isEmpty();
    head = tail = 0;
    resize(START_CAPACITY);
    return r;
}
}
```

Листинг 5 - Queue

```
package Java_task_26.ex1;

public interface Queue {
    Object dequeue();

    Object element();

    void enqueue(Object o);

    boolean isEmpty();

    boolean clear();
}
```

Листинг 6 – QueueTest

```
package Java_task_26.ex1;

public class QueueTest {
    public static void main(String[] args) {
        Java_task_26.ex1.ArrayQueueModule aqm = ArrayQueueModule.getInstance();
        aqm.enqueue("Text1");
        aqm.enqueue("Text2");
        aqm.enqueue("Text3");
        System.out.println(aqm.dequeue());

        ArrayQueueADT adt = new ArrayQueueADT(aqm);
        System.out.println(adt.dequeue());

        ArrayQueue aq = new ArrayQueue();
        System.out.println(aq.dequeue());
    }
}
```

Результат выполнения программы

Результат выполнения программы представлен на рисунке 1 – Array.

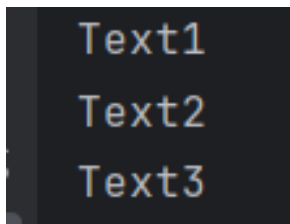


Рисунок 1 – Array

Вывод

Научились разрабатывать программы с абстрактными типами данных на языке Джава

Практическая работа №24

Цель работы

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

Теоретическое введение

Паттерны (или шаблоны) проектирования описывают типичные способы решения часто встречающихся проблем при проектировании программ. Некоторые из преимуществ использования шаблонов проектирования:

1) Шаблоны проектирования уже заранее определены и обеспечивают стандартный отраслевой подход к решению повторяющихся в программном коде проблем, вследствие этого разумное применение шаблона проектирования экономит время на разработку.

2) Использование шаблонов проектирования способствует реализации одного из преимуществ ООП – повторного использования кода, что приводит к более надежному и удобному в сопровождении коду. Это помогает снизить общую стоимость владения программным продуктом.

3) Поскольку шаблоны проектирования заранее определены то это упрощает понимание и отладку нашего кода. Это приводит к более быстрому развитию проектов, так как новые члены команды понимают код. Шаблоны проектирования Джава делятся на три категории: порождающие, структурные и поведенческие шаблоны проектирования.

Так же есть еще шаблоны проектирования, например MVC – model-view-controller. Шаблон проектирования Фабрика (factory), его также называют фабричный метод используется, когда у нас есть суперкласс с несколькими подклассами, и на основе ввода нам нужно вернуть один из подклассов. Этот шаблон снимает с себя ответственность за создание экземпляра класса из клиентской программы в класс фабрики. Давайте сначала узнаем, как реализовать фабричный шаблон проектирования в java, а затем мы рассмотрим преимущества фабричного шаблона. Мы увидим некоторые примеры использования фабричного шаблона проектирования в JDK. Обратите внимание, что этот шаблон также известен как шаблон проектирования фабричный метод.

Задание №2

Условие задачи

Реализовать класс Абстрактная фабрика для различных типов стульев: Викторианский стул, Многофункциональный стул, Магический стул, а также интерфейс Стул, от которого наследуются все классы стульев, и класс Клиент, который использует интерфейс стул в своем методе Sit (Chair chair)

Решение

Решение данной задачи представлено на листингах ниже:

Листинг 1 – Chair

```
public interface Chair {  
    void sit(Chair chair);  
    String GetType();  
}
```

Листинг 2 – ChairTypes

```
public enum ChairTypes {  
    VIC,  
    MULTIF,  
    MAGIC  
}
```

Листинг 3 – Client

```
public class Client implements Chair {  
    private final Factory chairFactory;  
    private String name;  
  
    public Client(Factory chairFactory, String name) {  
        this.chairFactory = chairFactory;  
        this.name = name;  
    }  
  
    public Chair orderChair(ChairTypes type) {  
        Chair chair = chairFactory.createChair(type);  
  
        System.out.println("Thanks for order: " + chair.GetType() + " see you again!");  
        return chair;  
    }  
  
    @Override  
    public void sit(Chair chair) {  
        System.out.println("So you sit on " + " " + chair.GetType());  
    }  
  
    @Override  
    public String GetType() {  
        return null;  
    }  
}
```

Листинг 4 – Factory

```
public class Factory {  
    public Chair createChair(ChairTypes type) {  
  
        return switch (type) {  
            case VIC -> new VIC();  
            case MULTIF -> new MULTIF();  
            case MAGIC -> new MAGIC();  
        };  
    }  
}
```

Листинг 5 – MAGIC

```
public class MAGIC implements Chair {  
    @Override  
    public void sit(Chair chair) {  
        System.out.println("this MAGIC");  
    }  
  
    @Override  
    public String GetType() {  
        return "magic";  
    }  
}
```

Листинг 6 – MULTIF

```
public class MULTIF implements Chair {  
    @Override  
    public void sit(Chair chair) {  
        System.out.println("This MULTIF");  
    }  
  
    @Override  
    public String GetType() {  
        return "multif";  
    }  
}
```

Листинг 7 – Main

```
public class Main {  
    public static void main(String[] args) {  
  
        Factory factory = new Factory() {  
        };  
  
        Chair chair1 = factory.createChair(ChairTypes.VIC);  
        Chair chair2 = factory.createChair(ChairTypes.MULTIF);  
        Chair chair3 = factory.createChair(ChairTypes.MAGIC);  
  
        Client person = new Client(factory, "Timofey");  
        person.sit(chair1);  
        person.sit(chair2);  
        person.sit(chair3);  
        person.orderChair(ChairTypes.VIC);  
    }  
}
```

Листинг 8 – VIC

```
public class VIC implements Chair {  
  
    @Override  
    public void sit(Chair chair) {  
        System.out.println("This VIC");  
    }  
  
    @Override  
    public String GetType() {  
        return "vic";  
    }  
}
```

Результат выполнения программы

Результат выполнения программы представлен на рисунке 1 – ChairFactory

```
So you sit on vic  
So you sit on multif  
So you sit on magic  
Thanks for order: vic see you again!
```

Рисунок 1 – ChairFactory

Вывод

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

Используемая литература

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