МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

ОТЧЁТ ПО ПРАКТИЧЕСКИМ РАБОТАМ №1-24

по дисциплине «Программирование на языке Джава»

Выполнил:

Студент группы ИКБО-20-22 «19» декабря 2023 г. _____ Воробьев А. О.

(подпись)

Принял:

Преподаватель кафедры ИиППО ИИТ «19» декабря 2023 г.. _____ Волков М. Ю.

(подпись)

Москва 2023 г.

Содержание

1. Практическая работа №1. Классы, как новые типы данных. Поля данных и методы

Цель работы: освоить на практике работу с классами языка программирования Java.

Задание: необходимо реализовать простейший класс «Книнга» на языке программирования Java. Не забудьте добавить метод toString() к вашему классу. Также в программе необходимо предусмотреть класс-тестер для тестирования класса и вывода информации об объекте.

Файл Book.java

```java
package ru.mirea.lab1;

public class Book {
    private String name;
    private int year;
    public Book(String n, int y) {
        name = n;
        year = y;
    }

    public Book(String n) {
        name  = n;
        year = 0;
    }

    public Book(){
        name = "Nothing";
        year = 0;
    }

    public void setYear(int year) {
        this.year = year;
    }

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

    public int getYear() {
```

```
30          return year;
31      }
32
33      public String getName() {
34          return name;
35      }
36
37      @Override
38      public String toString() {
39          return name + ", " + year;
40      }
41
42  }
43
```

Файл TestBook.java

```
1   package ru.mirea.lab1;
2
3   public class TestBook {
4       public static void main(String[] args){
5           Book b1 = new Book("Winnie the Pooh", 1949);
6           Book b2 = new Book("Harry Potter");
7           Book b3 = new Book("Jane Eyre");
8           Book b4 = new Book();
9           System.out.println("Name of the 4 book: " + b4.getName());
10          System.out.println("Year of the 4 book: " + b4.getYear());
11          b2.setName("Harry potter and the philosopher stone");
12          b4.setYear(1002);
13          b3.setYear(1987);
14          System.out.println(b1);
15          System.out.println(b2);
16          System.out.println(b3);
17          System.out.println(b4);
18      }
19  }
```

Результат выполнения программы:

```
Name of the 4 book: Nothing
Year of the 4 book: 0
Winnie the Pooh, 1949
Harry potter and the philosopher stone, 0
Jane Eyre, 1987
Nothing, 1002
```

Вывод: во время выполнения практической работы была освоена работа с классами языка программирования Java.

2. Практическая работа №2. Циклы, условия, переменные и массивы в Java.

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.

Задание: По UML диаграмме класса, представленной на рисунке 3.5. 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.

Файл Factorial.java

```
package ru.mirea.lab2;

public class Factorial {
    int n;

    public Factorial(int n) {
        this.n = n;
    }

    public int factorial(){
        int res = 1;
        for (int i = 1; i ⩽ n; i++) {
            res *= i;
        }
        return res;
    }
}


```

Файл Main.java

```java
package ru.mirea.lab2;

import java.text.DecimalFormat;
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5};
        int sum = 0;
        for (int i = 0; i < array.length; i++) {
            sum += array[i];
        }
        System.out.println("for: " + sum);

        sum = 0;
        int i = 0;
        while (i < array.length) {
            sum += array[i];
            i++;
        }

        System.out.println("while: " + sum);

        sum = 0;
        i = 0;
        do {
            sum += array[i];
            i++;
        } while (i < array.length);
        System.out.println("do while: " + sum);

        System.out.println();
        // Аргументы командной строки
        i = 1;
        for (String arg : args) {
            System.out.println("Аргумент " + i + ": " + arg);
            i++;
        }
        System.out.println();

        System.out.println();

        int n = 10; // Количество чисел в ряду
        double sum2 = 0.0;

        System.out.println("Первые " + n + " чисел гармонического ряда:");

```

```
48          DecimalFormat df = new DecimalFormat("#.##");
49
50          for (i = 1; i ⩽ n; i++) {
51              sum2 += 1.0 / i;
52              System.out.println("Число " + i + ": " + df.format(sum2));
53          }
54
55          System.out.println();
56
57          int[] array1 = new int[5];
58
59          System.out.print("Unsorted: ");
60          for (i = 0; i < 5; i++) {
61              array1[i] = (int) (Math.random() * (100 - 10) + 10);
62          }
63          for (int value : array1) {
64              System.out.print(value + " ");
65          }
66
67          System.out.print("\nSorted: ");
68          Arrays.sort(array1);
69          for (int value : array1) {
70              System.out.print(value + " ");
71          }
72
73          System.out.println();
74          Factorial fact = new Factorial(6);
75          System.out.println("\nФакториал 6: " + fact.factorial());
76
77      }
78  }
79
```

Результат выполнения программы:

```
for: 15
while: 15
do while: 15



Первые 10 чисел гармонического ряда:
Число 1: 1
Число 2: 1,5
Число 3: 1,83
Число 4: 2,08
Число 5: 2,28
Число 6: 2,45
Число 7: 2,59
Число 8: 2,72
Число 9: 2,83
Число 10: 2,93

Unsorted: 51 87 73 24 75
Sorted: 24 51 73 75 87

Факториал 6: 720

Process finished with exit code 0
```

Вывод: во время выполнения практической работы были получены практические навыки разработки программ, изучен синтаксис языка Java, освоены основные конструкции языка Java (циклы, условия, создание переменных и массивов, создание методов, вызов методов), а также возможность осуществления стандартного ввода/вывода данных.

3. Практическая работа №3. Использование UML диаграмм в объектно- ориентированном программировании

Цель работы: работа с UML-диаграммами классов.

Задание 1: 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 3.4.

Файл Book.java

```java
package ru.mirea.lab3.task2;

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

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

    public String getName() {
        return name;
    }

    public String getEmail() {
        return email;
    }

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

    public char getGender() {
        return gender;
    }
```

```
29
30      @Override
31      public String toString() {
32          return name + " (" + gender + ") at " + email;
33      }
34  }
```

Файл TestBook.java

```
1   package ru.mirea.lab3.task2;
2
3   public class TestAuthor {
4       public static void main(String[] args){
5           Author author1 = new Author("Jane Smith", "jane.smith@email.com", 'F');
6           Author author2 = new Author("John Doe", "john.doe@email.com", 'M');
7           Author author3 = new Author("Emily Johnson", " emily.johnson@email.com", 'U
8
9           System.out.println(author1.getName());
10          System.out.println(author2.getEmail());
11          System.out.println(author3.getGender());
12          author2.setEmail("test@somewhere.com");
13          System.out.println(author1);
14          System.out.println(author2);
15          System.out.println(author3);
16      }
17  }
```

Результат выполнения программы:

```
Jane Smith
john.doe@email.com
U
Jane Smith (F) at jane.smith@email.com
John Doe (M) at test@somewhere.com
Emily Johnson (U) at  emily.johnson@email.com
```

Задание 2: по UML диаграмме класса, представленной на рисунке 3.5. 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.

Файл Ball.java

```java
package ru.mirea.lab3.task1;

public class Ball {
    private double x = 0.0;
    private double y = 0.0;


    public Ball(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public Ball() {
    }

    public double getX() {
        return x;
    }

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

    public double getY() {
        return y;
    }

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

    public void setXY(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public void move(double xDisp, double yDisp) {
        x += xDisp;
        y += yDisp;
    }

```

```
42
43      @Override
44      public String toString() {
45          return "Ball @ (" + this.x + ", " + this.y + ").";
46      }
47  }
48
49
```

Файл TestBall.java

```
1   package ru.mirea.lab3.task1;
2
3   public class TestBall {
4       public static void main(String[] args) {
5           Ball b1 = new Ball(100, 100);
6           System.out.println(b1);
7           b1.move(30, 15);
8           System.out.println(b1);
9           Ball b2 = new Ball();
10          b2.setX(100);
11          b2.setY(50);
12          System.out.println(b2.getX());
13          System.out.println(b2.getY());
14          b1.setXY(20,50);
15          System.out.println(b2);
16      }
17  }
```

Результат выполнения программы:

```
Ball @ (100.0, 100.0).
Ball @ (130.0, 115.0).
100.0
50.0
Ball @ (100.0, 50.0).
```

Вывод: во время выполнения практической работы была освоена работа с UML- диаграммами классов.

4. Практическая работа №4. ООП в Java. Понятие класса.

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

Задание: создать класс, описывающий модель окружности (Circle). В классе должны быть описаны нужные свойства окружности и методы для получения, изменения этих свойств. Протестировать работу класса в классе CircleTest, содержащим метод статический main(String[] args).

Файл Circle.java

```
package ru.mirea.lab4;

public class Circle {
    private double radius;
    private int centerX;
    private int centerY;

    public Circle(double radius, int centerX, int centerY) {
        this.radius = radius;
        this.centerX = centerX;
        this.centerY = centerY;
    }

    public Circle() {
    }

    public double getRadius() {
        return radius;
    }

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

    public int getCenterX() {
        return centerX;
    }

    public void setCenterX(int centerX) {
```

```
        this.centerX = centerX;
    }

    public int getCenterY() {
        return centerY;
    }

    public void setCenterY(int centerY) {
        this.centerY = centerY;
    }

    @Override
    public String toString() {
        return "Circle{" +
                "radius=" + radius +
                ", centerX=" + centerX +
                ", centerY=" + centerY +
                '}';
    }
}
```

Файл CircleBook.java

```
package ru.mirea.lab4;

public class CircleTest {
    public static void main(String[] args) {
        Circle circle1 = new Circle(5, 0, 0);
        System.out.println(circle1);

        Circle circle2 = new Circle();
        circle2.setRadius(5.3);
        circle2.setCenterX(4);
        circle2.setCenterY(2);

        System.out.println("Radius: " + circle2.getRadius());
        System.out.println("Center: " + circle2.getCenterX() + " " + circle2.getCen
    }
}
```

Результат выполнения программы:

```
Circle{radius=5.0, centerX=0, centerY=0}
Radius: 5.3
Center: 4 2
```

Вывод: во время выполнения практической работы были освоены основные концепции объектно-ориентированного программирования, изучено понятие класса и освоено создание класса.

5. Практическая работа №5. Наследование. Абстрактные суперклассы и их подклассы в Java.

Цель работы: освоить на практике работу с абстрактными классами и наследованием на Java.

Задание 4: Напишите два класса MovablePoint и MovableCircle - которые реализуют интерфейс Movable.

Файл Movable.java

```
package ru.mirea.lab5.task4;

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


```

Файл Shape.java

```
package ru.mirea.lab5.task4;

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

    public Shape() {
    }

    public Shape(String color, boolean filled) {
        this.filled = filled;
        this.color = color;
    }

```

```
15      public String getColor() {
16          return color;
17      }
18
19      public void setColor(String color) {
20          this.color = color;
21      }
22
23      public boolean isFilled() {
24          return filled;
25      }
26
27      public void setFilled(boolean filled) {
28          this.filled = filled;
29      }
30
31      abstract double getArea();
32      abstract double getPerimeter();
33      @Override
34      public String toString() {
35          return "Shape{" +
36                  "filled=" + filled +
37                  ", color='" + color + '\'' +
38                  '}';
39      }
40  }
41
```

Файл Circle.java

```
1
2   package ru.mirea.lab5.task4;
3
4   public class Circle extends Shape {
5       protected double radius;
6
7       // Конструкторы
8       public Circle() {
9           this.filled = false;
10          this.color = "blue";
11          radius = 1;
12      }
13
14      public Circle(double radius) {
15          this.filled = false;
16          this.color = "blue";
17          this.radius = radius;
18      }
```

```
19
20      public Circle(double radius, String color,
21      boolean filled) {
22          this.radius = radius;
23          this.color = color;
24          this.filled = filled;
25      }
26
27      public double getRadius() {
28          return radius;
29      }
30
31      public void setRadius(double radius) {
32          this.radius = radius;
33      }
34
35      @Override
36      public double getArea() {
37          return Math.PI * radius * radius;
38      }
39
40      @Override
41      public double getPerimeter() {
42          return 2 * Math.PI * radius;
43      }
44
45      @Override
46      public String toString() {
47          return "Shape: circle, radius: " + this.radius
48          + ", color: " + this.color;
49      }
50  }
51
```

Файл MovableCircle.java

```
1   package ru.mirea.lab5.task4;
2
3   public class MovableCircle implements Movable{
4       private int radius;
5       private MovablePoint center;
6
7       public MovableCircle(int x, int y, int xSpeed,
8       int ySpeed, int radius){
9           this.center = new MovablePoint(x, y,
10          xSpeed, ySpeed);
11          this.radius = radius;
12      }
```

```
13
14      @Override
15      public String toString() {
16          return "MovableCircle{" +
17                  "radius=" + radius +
18                  ", center=" + center +
19                  '}';
20      }
21
22      @Override
23      public void moveUp() {
24          this.center.moveUp();
25      }
26
27      @Override
28      public void moveDown() {
29          this.center.moveDown();
30      }
31
32      @Override
33      public void moveLeft() {
34          this.center.moveLeft();
35      }
36
37      @Override
38      public void moveRight() {
39          this.center.moveRight();
40      }
41  }
42
```

Файл MovablePoint.java

```
1   package ru.mirea.lab5.task4;
2
3   public class MovablePoint implements Movable{
4       int x;
5       int y;
6       int xSpeed;
7       int ySpeed;
8
9       public MovablePoint(int x, int y, int xSpeed, int ySpeed) {
10          this.x = x;
11          this.y = y;
12          this.xSpeed = xSpeed;
13          this.ySpeed = ySpeed;
14      }
15
```

```
16      @Override
17      public String toString() {
18          return "MovablePoint{" +
19                  "x=" + x +
20                  ", y=" + y +
21                  ", xSpeed=" + xSpeed +
22                  ", ySpeed=" + ySpeed +
23                  '}';
24      }
25
26      public void moveUp(){
27          this.y+=ySpeed;
28      }
29
30      @Override
31      public void moveDown() {
32          this.y-=ySpeed;
33      }
34
35      @Override
36      public void moveLeft() {
37          this.x-=xSpeed;
38      }
39
40      @Override
41      public void moveRight() {
42          this.x+=xSpeed;
43      }
44  }
45
```

Файл Rectangle.java

```
1   package ru.mirea.lab5.task4;
2
3
4   public class Rectangle extends Shape {
5       protected double width, length;
6
7       public Rectangle() {
8       }
9
10      public Rectangle(double width, double length) {
11          this.width = width;
12          this.length = length;
13      }
14
15      public Rectangle(double width, double length,
```

```
16      String color, boolean filled) {
17          super(color, filled);
18          this.width = width;
19          this.length = length;
20      }
21
22      public double getWidth() {
23          return width;
24      }
25
26      public void setWidth(double width) {
27          this.width = width;
28      }
29
30      public double getLength() {
31          return length;
32      }
33
34      public void setLength(double length) {
35          this.length = length;
36      }
37
38      @Override
39      double getArea() {
40          return width*length;
41      }
42
43      @Override
44      double getPerimeter() {
45          return (width + length) * 2;
46      }
47
48      @Override
49      public String toString() {
50          return "Shape: rectangle, length: " +
51          this.length + " width: " + this.width;
52      }
53  }
54
```

Файл Square.java

```
1   package ru.mirea.lab5.task5;
2
3   public class Square extends Rectangle {
4       protected double side;
5
6       public Square() {
```

```
    }

    public Square(double side) {
        this.side = side;
    }

    public Square(double side, String color,
    boolean filled) {
        this.side = side;
        this.color = color;
        this.filled = filled;
    }

    public double getSide() {
        return side;
    }

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

    @Override
    public void setWidth(double width) {
        super.setWidth(width);
    }

    @Override
    public void setLength(double length) {
        super.setLength(length);
    }

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

Задание 5: Напишите новый класс MovableRectangle (движущийся прямоугольник). Его можно представить как две движущиеся точки MovablePoints (представляющих верхняя левая и нижняя правая точки) и реализующие интерфейс Movable. Убедитесь, что две точки имеет одну и ту же скорость (нужен метод это проверяющий).

Файл Movable.java

```java
package ru.mirea.lab5.task5;

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

```

Файл Shape.java

```java
package ru.mirea.lab5.task5;

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

    public Shape() {
    }

    public Shape(String color, boolean filled) {
        this.filled = filled;
        this.color = color;
    }

    public String getColor() {
        return color;
    }

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

    public boolean isFilled() {
        return filled;
    }

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

    abstract double getArea();
    abstract double getPerimeter();
    @Override
```

```
    public String toString() {
        return "Shape{" +
                "filled=" + filled +
                ", color='" + color + '\'' +
                '}';
    }
}

```

Файл Circle.java

```

package ru.mirea.lab5.task5;

public class Circle extends Shape {
    protected double radius;

    // Конструкторы
    public Circle() {
        this.filled = false;
        this.color = "blue";
        radius = 1;
    }

    public Circle(double radius) {
        this.filled = false;
        this.color = "blue";
        this.radius = radius;
    }

    public Circle(double radius,
    String color, boolean filled) {
        this.radius = radius;
        this.color = color;
        this.filled = filled;
    }

    public double getRadius() {
        return radius;
    }

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

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

```
38      }
39
40      @Override
41      public double getPerimeter() {
42          return 2 * Math.PI * radius;
43      }
44
45      @Override
46      public String toString() {
47          return "Shape: circle, radius: "
48          + this.radius + ", color: " + this.color;
49      }
50  }
51
```

Файл MovableCircle.java

```
1   package ru.mirea.lab5.task5;
2
3   public class MovableCircle implements Movable {
4       private int radius;
5       private MovablePoint center;
6
7       public MovableCircle(int x, int y, int
8       xSpeed, int ySpeed, int radius){
9           this.center = new MovablePoint(x, y,
10          xSpeed, ySpeed);
11          this.radius = radius;
12      }
13
14      @Override
15      public String toString() {
16          return "MovableCircle{" +
17                  "radius=" + radius +
18                  ", center=" + center +
19                  '}';
20      }
21
22      @Override
23      public void moveUp() {
24          this.center.moveUp();
25      }
26
27      @Override
28      public void moveDown() {
29          this.center.moveDown();
30      }
31
```

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

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

```

Файл MovablePoint.java

```
package ru.mirea.lab5.task5;

public class MovablePoint implements Movable {
    int x;
    int y;
    int xSpeed;
    int ySpeed;

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

    @Override
    public String toString() {
        return "MovablePoint{" +
                "x=" + x +
                ", y=" + y +
                ", xSpeed=" + xSpeed +
                ", ySpeed=" + ySpeed +
                '}';
    }

    public void moveUp(){
        this.y+=ySpeed;
    }

    @Override
    public void moveDown() {
        this.y-=ySpeed;
    }
```

```
35
36      @Override
37      public void moveLeft() {
38          this.x-=xSpeed;
39      }
40
41      @Override
42      public void moveRight() {
43          this.x+=xSpeed;
44      }
45  }
46
```

Файл MovableRectangle.java

```
1   package ru.mirea.lab5.task5;
2
3   public class MovableRectangle implements Movable {
4       private MovablePoint topLeft;
5       private MovablePoint bottomRight;
6       public MovableRectangle(int x1, int y1, int x2,
7       int y2, int xSpeed, int ySpeed){
8           topLeft = new MovablePoint(x1, y1,
9           xSpeed, ySpeed);
10          bottomRight = new MovablePoint(x2, y2,
11          xSpeed, ySpeed);
12      }
13
14      @Override
15      public String toString() {
16          return "MovableRectangle{" +
17                  "topLeft=" + topLeft +
18                  ", bottomRight=" + bottomRight +
19                  '}';
20      }
21
22      @Override
23      public void moveUp() {
24          if (topLeft.xSpeed == bottomRight.xSpeed
25          && topLeft.ySpeed == bottomRight.ySpeed)
26          {
27              topLeft.moveUp();
28              bottomRight.moveUp();
29          }
30      }
31
32      @Override
33      public void moveDown() {
```

```
34          if (topLeft.xSpeed == bottomRight.xSpeed
35          && topLeft.ySpeed == bottomRight.ySpeed)
36          {
37              topLeft.moveDown();
38              bottomRight.moveDown();
39          }
40
41      }
42
43      @Override
44      public void moveLeft() {
45          if (topLeft.xSpeed == bottomRight.xSpeed
46          && topLeft.ySpeed == bottomRight.ySpeed)
47          {
48              topLeft.moveLeft();
49              bottomRight.moveLeft();
50          }
51      }
52
53      @Override
54      public void moveRight() {
55          if (topLeft.xSpeed == bottomRight.xSpeed
56          && topLeft.ySpeed == bottomRight.ySpeed)
57          {
58              topLeft.moveRight();
59              bottomRight.moveRight();
60          }
61      }
62  }
63
```

Файл Rectangle.java

```
1   package ru.mirea.lab5.task5;
2
3
4   public class Rectangle extends Shape {
5       protected double width, length;
6
7       public Rectangle() {
8       }
9
10      public Rectangle(double width, double length) {
11          this.width = width;
12          this.length = length;
13      }
14
15      public Rectangle(double width,
```

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

    public double getWidth() {
        return width;
    }

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

    public double getLength() {
        return length;
    }

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

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

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

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

Файл Square.java

```
package ru.mirea.lab5.task5;

public class Square extends Rectangle {
    protected double side;

    public Square() {
```

```
7       }
8
9       public Square(double side) {
10          this.side = side;
11      }
12
13      public Square(double side, String color,
14      boolean filled) {
15          this.side = side;
16          this.color = color;
17          this.filled = filled;
18      }
19
20      public double getSide() {
21          return side;
22      }
23
24      public void setSide(double side) {
25          this.side = side;
26      }
27
28      @Override
29      public void setWidth(double width) {
30          super.setWidth(width);
31      }
32
33      @Override
34      public void setLength(double length) {
35          super.setLength(length);
36      }
37
38      @Override
39      public String toString() {
40          return "Shape: square, side: " + this.side;
41      }
42  }
43
```

Вывод: во время выполнения практической работы была освоена на практике работа с абстрактными классами и наследованием на Java.

6. Практическая работа №6. Наследование в java

Цель работы: изучить понятие наследования, и научиться реализовывать наследование в Java.

Задание: Создать абстрактный класс, описывающий посуду(Dish). С помощью наследования реализовать различные виды посуды, имеющие свои свойства и методы. Протестировать работу классов.

Файл Dish.java

```java
package ru.mirea.lab6;

abstract class Dish {
    private String material;
    private int mass;
    public Dish() {
    }

    public Dish(String material, int mass) {
        this.material = material;
        this.mass = mass;
    }

    public String getMaterial() {
        return material;
    }

    public void setMaterial(String material) {
        this.material = material;
    }

    public int getMass() {
        return mass;
    }

    public void setMass(int mass) {
        this.mass = mass;
    }

    public abstract void use();

    public void wash() {
        System.out.println("Мою посуду");
```

```
    }
}

```

Файл Mug.java

```
package ru.mirea.lab6;

public class Mug extends Dish {
    private int capacity;

    public Mug() {
    }

    public Mug(String material, int mass, int capacity) {
        super(material, mass);
        this.capacity = capacity;
    }

    public int getCapacity() {
        return capacity;
    }

    public void setCapacity(int capacity) {
        this.capacity = capacity;
    }

    @Override
    public void use() {
        System.out.println("Пью из кружки");
    }

    public void heatBeverage(){
        System.out.println("Подогреваю напиток в кружке");
    }
}

```

Файл Plate.java

```
package ru.mirea.lab6;

public class Plate extends Dish{
    private boolean isFlat;

```

```
    public Plate() {
    }
    public Plate(String material, int mass, boolean isFlat) {
        super(material, mass);
        this.isFlat = isFlat;
    }

    public boolean isFlat() {
        return isFlat;
    }

    public void setFlat(boolean flat) {
        isFlat = flat;
    }

    @Override
    public void use() {
        System.out.println("Ем с тарелки");
    }

    public void breakDish(){
        System.out.println("Тарелка разбилась!");
    }
}
```

Файл DishTest.java

```
package ru.mirea.lab6;

public class DishTest {
    public static void main(String[] args) {
        Plate plate1 = new Plate("Керамика", 500, true);
        Mug mug1 = new Mug("Стекло", 10, 250);

        plate1.use();
        plate1.wash();
        plate1.breakDish();

        System.out.println();
        mug1.use();
        mug1.wash();
        mug1.heatBeverage();
        System.out.println();

        Plate plate2 = new Plate();
        plate2.setFlat(false);
        plate2.setMass(500);
```

```
21          plate2.setMaterial("Стекло");
22
23          System.out.println("Plate2: \n(is flat) " + plate2.isFlat() +
24          "\nmass " + plate2.getMass() + "\nmaterial " + plate2.getMaterial());
25
26          Mug mug2 = new Mug();
27          mug2.setCapacity(300);
28          mug2.setMass(300);
29          mug2.setMaterial("Фарфор");
30
31          System.out.println("\nMug2: \ncapacity " + mug2.getCapacity() +
32          "\nmass " + mug2.getMass() + "\nmaterial " + mug2.getMaterial());
33      }
34  }
35
```

Результат выполнения программы:

```
Ем с тарелки
Мою посуду
Тарелка разбилась!

Пью из кружки
Мою посуду
Подогреваю напиток в кружке

Plate2:
(is flat) false
mass 500
material Стекло

Mug2:
capacity 300
mass 300
material Фарфор
```

Вывод: во время выполнения данной работы было изучено понятие наследования, и реализация наследования в Java.

7. Практическая работа №7. Создание GUI. Событийное программирование в java.

Цель работы: введение в событийное программирование на языке Java.

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

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

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

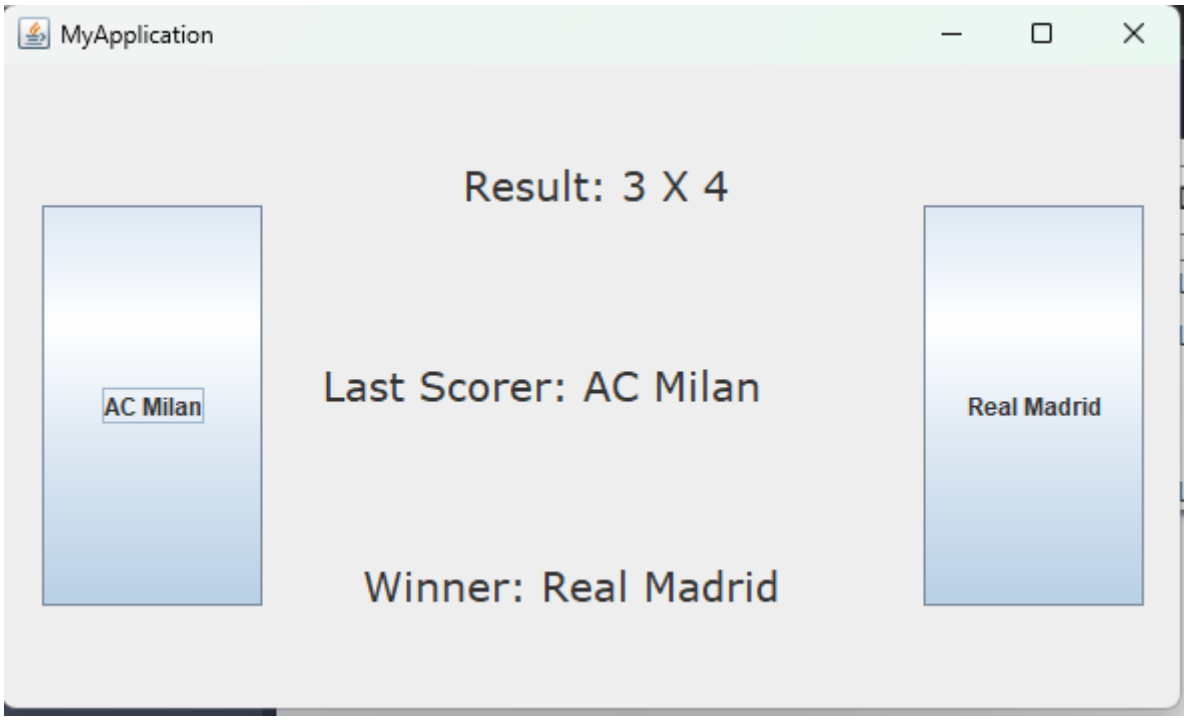
Файл MyApplication.java

```
1  package ru.mirea.lab7;
2  import java.awt.*;
3  import java.awt.event.*;
4  import javax.swing.*;
5
6  public class MyApplication extends JFrame
7  {
```

```
8       private int countMilan;
9       private int countMadrid;
10      JButton but1 = new JButton("AC Milan");
11      JButton but2 = new JButton("Real Madrid");
12      JLabel lbl1 = new JLabel("Result: 0 X 0");
13      JLabel lbl2 = new JLabel("Last Scorer: N/A");
14      JLabel lbl3 = new JLabel("Winner: DRAW");
15
16      public MyApplication() {
17          super("MyApplication");
18          setSize(600,360);
19          lbl1.setFont(new Font("Verdana", Font.PLAIN, 20));
20          lbl2.setFont(new Font("Verdana", Font.PLAIN, 20));
21          lbl3.setFont(new Font("Verdana", Font.PLAIN, 20));
22          setLayout(null);
23
24          but1.setBounds(20,70,110,200);
25          but2.setBounds(460,70,110,200);
26
27          lbl1.setBounds(230,10,200,100);
28          lbl2.setBounds(160,110,300,100);
29          lbl3.setBounds(180,210,300,100);
30
31          but1.addMouseListener(new MouseListener() {
32              @Override
33              public void mouseClicked(MouseEvent e) {
34                  countMilan++;
35                  lbl1.setText("Result: " + countMilan + " X " + countMadrid);
36                  lbl2.setText("Last Scorer: AC Milan");
37                  if (countMadrid > countMilan)
38                      lbl3.setText("Winner: Real Madrid");
39                  else if (countMadrid < countMilan)
40                      lbl3.setText("Winner: AC Milan");
41                  else
42                      lbl3.setText("Winner: unknown");
43              }
44              @Override
45              public void mousePressed(MouseEvent e) {
46
47              }
48              @Override
49              public void mouseReleased(MouseEvent e) {
50
51              }
52              @Override
53              public void mouseEntered(MouseEvent e) {
54
55              }
56              @Override
57              public void mouseExited(MouseEvent e) {
```

```
58
59              }
60          });
61
62          but2.addMouseListener(new MouseListener() {
63              @Override
64              public void mouseClicked(MouseEvent e) {
65                  countMadrid++;
66                  lbl1.setText("Result: " + countMilan + " X " + countMadrid);
67                  lbl2.setText("Last Scorer: Real Madrid");
68                  if (countMadrid > countMilan)
69                      lbl3.setText("Winner: Real Madrid");
70                  else if (countMadrid < countMilan)
71                      lbl3.setText("Winner: AC Milan");
72                  else
73                      lbl3.setText("Winner: unknown");
74              }
75              @Override
76              public void mousePressed(MouseEvent e) {
77
78              }
79              @Override
80              public void mouseReleased(MouseEvent e) {
81
82              }
83              @Override
84              public void mouseEntered(MouseEvent e) {
85
86              }
87              @Override
88              public void mouseExited(MouseEvent e) {
89
90              }
91          });
92
93          add(but1);
94          add(but2);
95          add(lbl1);
96          add(lbl2);
97          add(lbl3);
98      }
99      public static void main(String[] args) {
100         MyApplication frame = new MyApplication();
101         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
102         frame.setVisible(true);
103     }
104 }
105
```

Результат выполнения программы:



Вывод: во время выполнения практической работы было освоено событийное программирование на языке Java.

8. Практическая работа №8. Создание программ с графическим интерфейсом пользователя на Java

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

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

Файл Shape.java

```
package ru.mirea.lab8.task1;

import java.awt.*;

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

    public Shape(Color color, int x, int y) {
        this.color = color;
        this.x = x;
        this.y = y;
    }

    abstract void paint(Graphics graphics);
}

```

Файл Application.java

```
package ru.mirea.lab8.task1;
import javax.swing.*;

```

```
public class Application extends JFrame{

    public Application()
    {
        super("Application");
        add("Center", new MyCanvas());
        setSize(800, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);

    }

}

```

Файл Circle.java

```
package ru.mirea.lab8.task1;

import java.awt.*;

public class Circle extends Shape{
    private int radius;

    public Circle(Color color, int x, int y, int radius) {
        super(color, x, y);
        this.radius = radius;
    }

    @Override
    void paint(Graphics graphics) {
        Graphics2D g = (Graphics2D) graphics;
        g.setColor(color);
        g.fillOval(x, y, radius, radius);
        g.setColor(Color.BLACK);
        g.drawOval(x, y, radius, radius);
    }
}

```

Файл MyCanvas.java

```
package ru.mirea.lab8.task1;

import java.awt.*;
```

```
import java.awt.Graphics;
import java.awt.Graphics2D;

public class MyCanvas extends Canvas {
    public int getRandom(int min, int max) {
        return (int) (Math.random() * ((max + 1) - min) + min);
    }

    public void paint(Graphics graphics) {
        Graphics2D g = (Graphics2D) graphics;

        Color[] colors = {Color.RED, Color.BLUE, Color.MAGENTA, Color.cyan, Color.y
        for (int i = 0; i < 20; i++) {

            int randInt = getRandom(1, 3);
            int x = getRandom(100, 700);
            int y = getRandom(100, 500);
            Color color = colors[getRandom(0, 5)];
            int height, width, radius;

            switch (randInt) {
                case 1:
                    radius = getRandom(50, 200);
                    Circle circle = new Circle(color, x, y, radius);
                    circle.paint(g);
                    break;
                case 2:
                    width = getRandom(50, 200);
                    height = getRandom(50, 200);
                    Rectangle rectangle = new Rectangle(color, x, y, width, height)
                    rectangle.paint(g);
                    break;
                case 3:
                    height = getRandom(50, 200);
                    Triangle triangle = new Triangle(color, x, y, height);
                    triangle.paint(g);
                    break;
            }

        }


    }
}

```

Файл Rectangle.java

```java
package ru.mirea.lab8.task1;

import java.awt.*;

public class Rectangle extends Shape{
    private int width;
    private int height;

    public Rectangle(Color color, int x, int y, int width, int height) {
        super(color, x, y);
        this.width = width;
        this.height = height;
    }

    @Override
    void paint(Graphics graphics) {
        Graphics2D g = (Graphics2D) graphics;
        g.setColor(color);
        g.fillRect(x, y, width, height);
        g.setColor(Color.BLACK);
        g.drawRect(x, y, width, height);

    }
}

```

Файл Triangle.java

```java
package ru.mirea.lab8.task1;

import java.awt.*;

public class Triangle extends Shape{
    private int height;

    public Triangle(Color color, int x, int y, int height) {
        super(color, x, y);
        this.height = height;
    }

    @Override
    void paint(Graphics graphics) {
        Graphics2D g = (Graphics2D) graphics;

        int halfHeight = height / 2;
```

```
        int[] xPoints = {
                x,              // Вершина, расположенная в центре
                x - halfHeight, // Вершина слева
                x + halfHeight  // Вершина справа
        };
        int[] yPoints = {
                y - halfHeight, // Вертикальная координата вершин
                y + halfHeight, // Вертикальная координата вершины снизу
                y + halfHeight  // Вертикальная координата вершины снизу
        };
        int nPoints = 3; // Количество вершин в треугольнике
        g.setColor(color);
        g.fillPolygon(xPoints, yPoints, nPoints); // Рисуем треугольник
        g.setColor(Color.BLACK);
        g.drawPolygon(xPoints, yPoints, nPoints);
    }
}

```

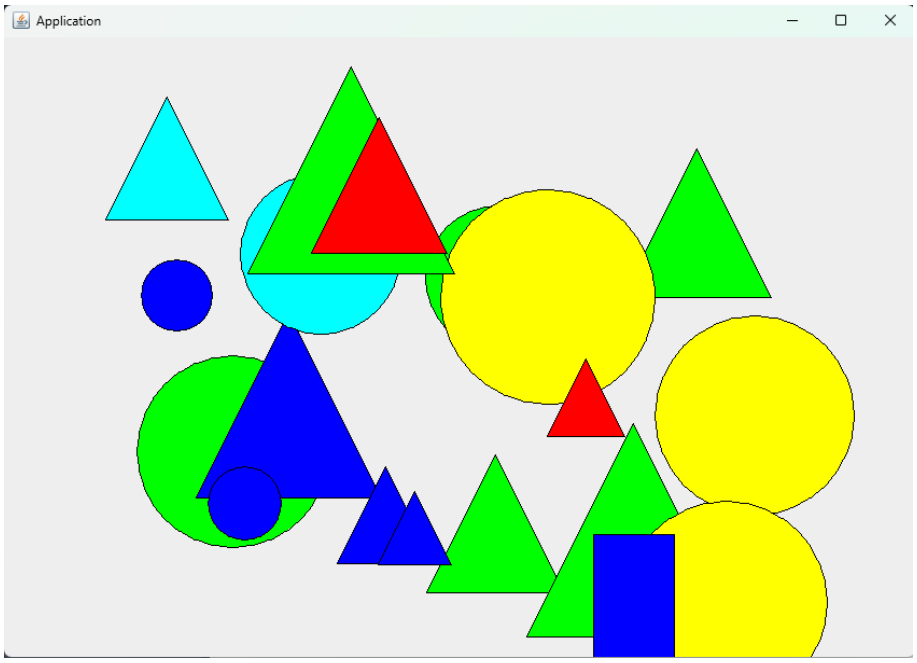
Файл RunApplication.java

```
package ru.mirea.lab8.task1;

public class RunApplication {
    public static void main(String[] args) {
        new Application();
    }
}

```

Результат выполнения программы:



Вывод: во время выполнения практической работы были освоены навыки создания графического интерфейса пользователя, освоиена на практике работа с различными объектами для создания ГИП, менеджерами размещения компонентов.

9. Практическая работа №9. Интерфейсы в Java.

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

Задание: Создать интерфейс Nameable, с методом getName(), возвращающим имя объекта, реализующего интерфейс. Проверить работу для различных объектов (например, можно создать классы, описывающие разные сущности, которые могут иметь имя: планеты, машины, животные и т. д.).

Файл Nameable.java

```
1  package ru.mirea.lab9;
2
3  public interface Nameable {
4      String getName();
5
6  }
7
```

Файл Dog.java

```
1  package ru.mirea.lab9;
2
3  public class Dog implements Nameable{
4      private String name;
5
6      public Dog(String name) {
7          this.name = name;
8      }
9      @Override
10     public String getName() {
11         return name;
12     }
13 }
14
15
```

Файл Planet.java

```java
package ru.mirea.lab9;

public class Planet implements Nameable{
    private String name;

    public Planet(String name) {
        this.name = name;
    }
    @Override
    public String getName() {
        return name;
    }
}


```

Файл Testclass.java

```java
package ru.mirea.lab9;

public class TestClass {
    public static void main(String[] args) {
        Nameable nameableDog = new Dog("Rex");
        Nameable nameablePlanet = new Planet("Earth");
        System.out.println(nameablePlanet.getName());
        System.out.println(nameableDog.getName());
    }
}

```

Результат выполнения программы:

```
Earth
Rex
```

Вывод: во время выполнения практической работы было изучено понятие интерфейса, получен навык создавания интерфейсы в Java и применение их в программах.

10. Практическая работа №10. Программирование рекурсии в Java.

Цель работы: разработка и программирование рекурсивных алгоритмов на языке Java.

Задание 3: Даны два целых числа A и B (каждое в отдельной строке). Выведите все числа от A до B включительно, в порядке возрастания, если A < B, или в порядке убывания в противном случае.

Файл Print.java

```
1  package ru.mirea.lab10.task3;
2
3  public class Print {
4      public static int print(int a, int b){
5          if (a == b){
6              System.out.println(b);
7              return 0;
8          }
9          else if (a < b){
10             System.out.println(a);
11             return print(a + 1, b);
12         }
13         else {
14             System.out.println(a);
15             return print(a - 1, b);
16         }
17     }
18     public static void main(String[] args) {
19         print(1, 5);
20         System.out.println();
21         print(20, 2);
22     }
23 }
24
```

Результат выполнения программы:

```
1
2
3
4
5

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
```

Задание 4: Даны натуральные числа k и s. Определите, сколько существует k- значных натуральных чисел, сумма цифр которых равна s. Запись натурального числа не может начинаться с цифры 0.

В этой задаче можно использовать цикл для перебора всех цифр, стоящих на какой-либо позиции.

Файл CountNumbers.java

```java
package ru.mirea.lab10.task4;

public class CountNumbers {
    public static int count_numbers(int k, int s, int start_digit,
    int current_sum){
        if (k == 0)
            if (current_sum == s)
                return 1;
            else
                return 0;

        int count = 0;

        for (int digit = start_digit; digit < 10; digit++) {
            if (current_sum + digit <= s){
                count += count_numbers(k - 1, s, digit, current_sum
                + digit);
            }
        }
        return count;
    }
    public static void main(String[] args) {
        System.out.println(count_numbers(2, 5, 0, 0));

    }
}

```

Результат выполнения программы:

```
3
```

Задание 5: Дано натуральное число N. Вычислите сумму его цифр. При решении этой задачи нельзя использовать строки, списки, массивы (ну и циклы, разумеется).

Файл CountNumbersSum.java

```java
package ru.mirea.lab10.task5;

public class CountNumbersSum  {
```

```
    public static int countNumbersSum(int n, int current_sum){
        if (n == 0)
            return current_sum;

        return countNumbersSum(n / 10, current_sum + n % 10);
    }
    public static void main(String[] args) {
        System.out.println(countNumbersSum(1242345679, 0));

    }
}

```

Результат выполнения программы:

```
43


Process finished with exit code 0
```

Вывод: во время выполнения практической работы был получен навык разработки и программирования рекурсивных алгоритмов на языке Java.

11. Практическая работа №11. Техники сортировки в Java

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

Задание 1: Написать тестовый класс, который создает массив класса Student и сортирует массив iDNumber и сортирует его вставками.

Файл TestClass.java

```java
package ru.mirea.lab11.task1;
public class TestClass {
    public static void main(String[] args) {
        Student[] array = new Student[]{new Student(102, "Alex"),
        new Student(104, "Max"), new Student(106, "Tom"),
        new Student(101, "Bob")};
        for (Student s: array){
            System.out.println(s);
        }

        System.out.println("------------");

        // Сортировка вставками
        for (int index = 1; index < array.length; index++)
        {
            Student key = array[index];
            int position = index;
            // Shift larger values to the right
            while (position > 0 && key.compareTo(array[position-1]) < 0) {
                array[position] = array[position-1];
                position--;
            }

            array[position] = key;
        }
        for (Student s: array){
            System.out.println(s);
        }
    }
}
```

Файл Student.java

```java
package ru.mirea.lab11.task1;

public class Student implements Comparable<Student>{
    int idNumber;
    String name;

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

    public int getIdNumber() {
        return idNumber;
    }

    public void setIdNumber(int idNumber) {
        this.idNumber = idNumber;
    }

    public String getName() {
        return name;
    }

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

    @Override
    public String toString() {
        return "Student{" +
                "idNumber=" + idNumber +
                ", name='" + name + '\'' +
                '}';
    }

    @Override
    public int compareTo(Student o) {
        if (idNumber == o.getIdNumber())
            return 0;
        if (idNumber < o.getIdNumber())
            return -1;

        return 1;
    }
}
```

Результат выполнения программы:

```
Student{idNumber=102, name='Alex'}
Student{idNumber=104, name='Max'}
Student{idNumber=106, name='Tom'}
Student{idNumber=101, name='Bob'}
------------
Student{idNumber=101, name='Bob'}
Student{idNumber=102, name='Alex'}
Student{idNumber=104, name='Max'}
Student{idNumber=106, name='Tom'}
```

Задание 2: Напишите класс SortingStudentsByGPA который реализует интерфейс Comparator таким образом, чтобы сортировать список студентов по их итоговым баллам в порядке убывания с использованием алгоритма быстрой сортировки

Файл SortingStudentsByGPA.java

```
package ru.mirea.lab11.task2;

import java.util.Comparator;

public class SortingStudentsByGPA implements Comparator<Student> {
    @Override
    public int compare(Student o1, Student o2) {
        return (o1.getScore() - o2.getScore());
    }
}

```

Файл Student.java

```java
package ru.mirea.lab11.task2;

public class Student{
    int idNumber;
    String name;
    int score;

    public void setScore(int score) {
        this.score = score;
    }

    public int getScore() {
        return score;
    }

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

    public int getIdNumber() {
        return idNumber;
    }

    public void setIdNumber(int idNumber) {
        this.idNumber = idNumber;
    }

    public String getName() {
        return name;
    }

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

    @Override
    public String toString() {
        return "Student{" +
                "idNumber=" + idNumber +
                ", name='" + name + '\'' +
                ", score=" + score +
                '}';
    }
}

```

Файл TestClass.java

```java
package ru.mirea.lab11.task2;

public class TestClass {
    public static void main(String[] args) {
        Student[] array = new Student[]{new Student(102, "Alex", 12),
        new Student(104, "Max", 5), new Student(106, "Tom", 42),
        new Student(101, "Bob", 100)};
        for (Student s: array){
            System.out.println(s);
        }
        System.out.println("------------");
        SortingStudentsByGPA comparator = new SortingStudentsByGPA();
        quickSort(array, 0, array.length -1, comparator);

        for (Student s: array){
            System.out.println(s);
        }
    }


    private static void quickSort(Student[] arr, int from, int to,
    SortingStudentsByGPA comparator){
        if (from < to){
            int divideIndex = partition(arr, from, to, comparator);
            quickSort(arr, from, divideIndex - 1, comparator);
            quickSort(arr, divideIndex, to,comparator);
        }
    }
    public static int partition(Student[] arr, int from, int to,
    SortingStudentsByGPA comparator){
        int rightIndex = to;
        int leftIndex = from;

        Student pivot = arr[from];
        while (leftIndex ⩽ rightIndex){
            while (leftIndex ⩽ rightIndex){
                while (comparator.compare(arr[leftIndex], pivot) < 0){
                    leftIndex++;
                }

                while (comparator.compare(arr[rightIndex], pivot) > 0){
                    rightIndex--;
                }
                if (leftIndex ⩽ rightIndex) {
                    swap(arr, rightIndex, leftIndex);
                    leftIndex++;
                    rightIndex--;
```

```
                }
            }
        }
        return leftIndex;
    }

    private static void swap(Student[] arr, int index1, int index2) {
        Student tmp = arr[index1];
        arr[index1] = arr[index2];
        arr[index2] = tmp;
    }


}

```

Результат выполнения программы:

```
Student{idNumber=102, name='Alex', score=12}
Student{idNumber=104, name='Max', score=5}
Student{idNumber=106, name='Tom', score=42}
Student{idNumber=101, name='Bob', score=100}
------------
Student{idNumber=104, name='Max', score=5}
Student{idNumber=102, name='Alex', score=12}
Student{idNumber=106, name='Tom', score=42}
Student{idNumber=101, name='Bob', score=100}
```

Задание 3: Напишите программу, которая объединяет два списка данных о студентах в один отсортированный списках с использованием алгоритма сортировки слиянием.

Файл SortingStudentsByGPA.java

```
package ru.mirea.lab11.task3;

import java.util.Comparator;

```

```java
public class SortingStudentsByGPA implements Comparator<Student> {
    @Override
    public int compare(Student o1, Student o2) {
        return (o1.getScore() - o2.getScore());
    }
}

```

Файл Student.java

```java
package ru.mirea.lab11.task3;

public class Student {
    int idNumber;
    String name;
    int score;

    public void setScore(int score) {
        this.score = score;
    }

    public int getScore() {
        return score;
    }

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

    public int getIdNumber() {
        return idNumber;
    }

    public void setIdNumber(int idNumber) {
        this.idNumber = idNumber;
    }

    public String getName() {
        return name;
    }

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

    @Override
```

```
    public String toString() {
        return "Student{" +
                "idNumber=" + idNumber +
                ", name='" + name + '\'' +
                ", score=" + score +
                '}';
    }
}
```

Файл TestClass.java

```
package ru.mirea.lab11.task3;

import java.util.Arrays;
class TestClass {
    public static void main(String[] args) {
        Student[] array = new Student[]{new Student(102, "Alex", 12),
        new Student(104, "Max", 5), new Student(106, "Tom", 42),
        new Student(101, "Bob", 100)};
        for (Student s: array){
            System.out.println(s);
        }
        System.out.println("------------");
        SortingStudentsByGPA comparator = new SortingStudentsByGPA();

        array = mergeSort(array, comparator);

        for (Student s: array){
            System.out.println(s);
        }
    }


    private static Student[] merge(Student[] left, Student[] right,
    SortingStudentsByGPA comparator){
        Student[] result = new Student[left.length + right.length];

        int resIn = 0, leftIn = 0, rightIn = 0;

        while (leftIn < left.length && rightIn < right.length)
            if (comparator.compare(left[leftIn], right[rightIn]) < 0)
                result[resIn++] = left[leftIn++];
            else result[resIn++] = right[rightIn++];

        while (resIn < result.length)
            if (leftIn != left.length)
                result[resIn++] = left[leftIn++];
```

```
            else result[resIn++] = right[rightIn++];

        return result;
    }
    public static Student[] mergeSort(Student[] src,
    SortingStudentsByGPA comparator) {
        if (src.length <= 1)
            return src;

        Student[] left = Arrays.copyOfRange(src, 0, src.length / 2);
        Student[] right = Arrays.copyOfRange(src, left.length, src.length);

        return merge(mergeSort(left,comparator),
        mergeSort(right,comparator),comparator);
    }
}

```

Результат выполнения программы:

```
Student{idNumber=102, name='Alex', score=12}
Student{idNumber=104, name='Max', score=5}
Student{idNumber=106, name='Tom', score=42}
Student{idNumber=101, name='Bob', score=100}
------------
Student{idNumber=104, name='Max', score=5}
Student{idNumber=102, name='Alex', score=12}
Student{idNumber=106, name='Tom', score=42}
Student{idNumber=101, name='Bob', score=100}
```

Вывод: во время выполнения практической работы на практике были освоены методы сортировки с использованием приемов программирования на объектно-ориентированном языке Java.

12. Практическая работа №12. Использование стандартных контейнерных классов при программировании на Java

Цель работы: изучение на практике приемов работы со стандартными контейнерными классами Java Collection Framework.

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

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

Для простоты будем считать, что все карты различны по номиналу и что самая младшая карта побеждает самую старшую карту (“шестерка берет туз”). Игрок, который забирает себе карты, сначала кладет под низ своей колоды карту первого игрока, затем карту второго игрока (то есть карта второго игрока оказывается внизу колоды).

Файл GameStack.java

```
1  package ru.mirea.lab12;
2
3  import java.util.Stack;
4
5  public class GameStack {
6      Stack<Integer> firstPlayer;
7      Stack <Integer> secondPlayer;
8
9      public GameStack(String firstPlayerStr, String secondPlayerString) {
10         this.firstPlayer = new Stack<>();
11         this.secondPlayer = new Stack<>();
```

```
        for (int i = 0; i < 5; i++){
            this.firstPlayer.push(Integer.parseInt
            (firstPlayerStr.substring(i, i+1)));
            this.secondPlayer.push(Integer.parseInt
            (secondPlayerString.substring(i, i+1)));
        }
    }
    private void pushBack(Stack<Integer> resultStack, Integer lastCard) {
        Stack<Integer> tempStack = new Stack<>();

        Integer prelastCard = resultStack.pop();

        while (!resultStack.empty())
            tempStack.push(resultStack.pop());

        resultStack.push(lastCard);
        resultStack.push(prelastCard);

        while (!tempStack.empty())
            resultStack.push(tempStack.pop());

    }
    public String play(){
        String result;
        int count = 0;

        while (!firstPlayer.empty() && !secondPlayer.empty()
        && (count <= 106) ){
            if (firstPlayer.peek() > secondPlayer.peek()
            && secondPlayer.peek() != 0){
                pushBack(firstPlayer, secondPlayer.pop());
                // первый игрок забирает карты
            } else{
                pushBack(secondPlayer, firstPlayer.pop());
                // второй игрок забирает карты
            }
            count++;
        }

        if (firstPlayer.empty())
        {
            result = "second " + count;
            return result;
        }
        else if (secondPlayer.empty()) {
            result = "first " + count;
            return result;
        }
        else
            return "botva";
```

```
    }
}


```

Файл TestClass.java

```
package ru.mirea.lab12;

import java.util.Scanner;

public class TestClass {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String firstString, secondString;

        String string = scan.nextLine();
        firstString = string.split(" ")[0];
        secondString = string.split(" ")[1];
        System.out.println(new GameStack(firstString, secondString).play());

    }


}

```

Результат выполнения программы:

```
13579 24680
second 5
```

Вывод: во время выполнения практической работы изучены приемы работы со стандартными контейнерными классами Java Collection Framework.

13. Практическая работа №13. Работа с файлами

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

Задание:

1. Реализовать запись в файл введённой с клавиатуры информации
2. Реализовать вывод информации из файла на экран
3. Заменить информацию в файле на информацию, введённую с клавиатуры
4. Добавить в конец исходного файла текст, введённый с клавиатуры

Файл FilesTest.java

```
package ru.mirea.lab13.task1;

import java.io.*;
import java.util.Scanner;

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

        String scan = scanner.nextLine();

        try(FileWriter writer = new FileWriter("notes1.txt", false))
        {
            writer.write(scan);
            writer.flush();
        }
        catch(IOException ex){
            System.out.println(ex.getMessage());
        }
    }
}

```

Файл FileRead.java

```java
package ru.mirea.lab13.task2;

import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class FileRead {
    public static void main(String[] args) {
        try(FileReader reader = new FileReader("notes1.txt"))
        {
            int c;
            while ((c=reader.read()) != -1){
                System.out.print((char)c);
            }
        }
        catch(IOException ex){
            System.out.println(ex.getMessage());
        }
    }
}

```

Файл FileTest.java

```java
package ru.mirea.lab13.task3;

import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

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

        String scan = scanner.nextLine();

        try(FileWriter writer = new FileWriter("notes1.txt", false))
        {
            writer.write(scan);
            writer.flush();
        }
        catch(IOException ex){
            System.out.println(ex.getMessage());
        }
    }
```

```
}

```

Файл FileTesting.java

```
package ru.mirea.lab13.task4;

import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

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

        String scan = scanner.nextLine();

        try(FileWriter writer = new FileWriter("notes1.txt", true))
        {
            writer.write(scan);
            writer.flush();
        }
        catch(IOException ex){
            System.out.println(ex.getMessage());
        }
    }
}

```

Вывод: во время выполнения практической работы была освоена на практике работа с файлами на языке Java. Получены практические навыки по чтению и записи данных в файл.

14. Практическая работа №14. Различные виды списков ожидания

Цель работы: освоить на практике работу с различными видами списков ожидания

Задание:

- Исследуйте UML диаграмму классов на рисунке 1 и понаблюдайте, как она выражает то, что мы говорили выше в словах. Убедитесь, что вы понимаете все аспекты диаграммы.
- Расширить и модифицировать исходный код WaitList, как необходимо, чтобы полностью реализовать всю схему UML. Включить комментарии Javadoc. Обратите внимание на переключение ролей после реализации каждого интерфейса / класса!
- Изучение работу метода main(), которая использует ваши новые классы и интерфейс.

Файл IWaitList.java

```
package ru.mirea.lab14;

import java.util.Collection;

public interface IWaitList<E> {

    void add(E element);

    E remove();

    boolean contains(E element);

    boolean containsAll(Collection<E> c);

    boolean isEmpty();

```

```
}

```

Файл BoundedWaitList.java

```
package ru.mirea.lab14;

import java.util.Collection;


public class BoundedWaitList<E> extends WaitList<E> {
    private int capacity;

    public BoundedWaitList(int capacity) {
        super();
        this.capacity = capacity;
    }

    public int getCapacity() {
        return capacity;
    }


    @Override
    public void add(E element) {
        if (content.size() == capacity) throw new
        IllegalStateException("Очередь заполнена");
        content.add(element);
    }

    @Override
    public String toString() {
        return "BoundedWaitList{" +
                "capacity=" + capacity +
                ", content=" + content +
                '}';
    }
}

```

Файл UnfairList.java

```
package ru.mirea.lab14;

import java.util.ArrayList;
```

```
4
5  public class UnfairWaitList<E> extends WaitList<E>{
6      ArrayList<E> removedItems = new ArrayList<>();
7
8      public UnfairWaitList() {
9          super();
10     }
11
12     @Override
13     public void add(E element) {
14         if (!removedItems.contains(element))
15             super.add(element);
16         else
17             throw new IllegalStateException
18             ("Удаленный элемент не может быть добавлен снова");
19     }
20
21     public void remove(E element) {
22         boolean removed = false;
23         for (int i = 0; i < content.size(); i++) {
24
25             E el = content.remove();
26             if (!removed && el.equals(element)) {
27                 if (i == 0){
28                     throw new IllegalStateException
29                     ("Первый элемент не может быть удален");
30                 }
31                 removed = true;
32                 removedItems.add(element);
33             }
34             else
35                 content.add(el);
36         }
37         if (removed) {
38             content.add(content.remove());
39         }
40     }
41
42     public void MoveToBack(E element){
43         remove(element);
44         content.add(element);
45     }
46 }
47
```

Файл WaitList.java

```java
package ru.mirea.lab14;

import java.util.ArrayList;
import java.util.Collection;
import java.util.concurrent.ConcurrentLinkedQueue;

public class WaitList<E> implements IWaitList<E>{

    protected ConcurrentLinkedQueue<E> content;

    public WaitList(){
        content = new ConcurrentLinkedQueue<>();
    }

    public WaitList(Collection<E> c) {
        content = new ConcurrentLinkedQueue<>(c);
    }

    @Override
    public String toString() {
        return "WaitList{" +
                "content=" + content +
                '}';
    }

    @Override
    public void add(E element) {

        content.add(element);
    }


    @Override
    public E remove() {
        if (content.isEmpty()) throw new IllegalStateException("Пустая очередь");
        return content.remove();
    }


    @Override
    public boolean contains(E element) {
        boolean res = false;
        for (int i = 0; i < content.size(); i++) {
            E el = content.remove();
            if (el.equals(element))
                res = true;
            content.add(el);
```

```
        }
        return res;
    }

    @Override
    public boolean containsAll(Collection<E> c) {
        ArrayList<E> al = new ArrayList<>(c);
        for (int i = 0; i < c.size(); i++) {
            E el = al.get(i);
            boolean res = contains(el);
            if (res == false)
                return false;
        }
        return true;
    }

    @Override
    public boolean isEmpty() {
        return content.isEmpty();
    }
}

```

Файл lab14.java

```
package ru.mirea.lab14;

import java.util.ArrayList;

public class lab14 {
    public static void main(String[] args) {

/*        ArrayList<String> arrayList = new ArrayList<>();
        arrayList.add("item1");
        arrayList.add("item2");
        arrayList.add("item3");

        WaitList<String> waitList = new WaitList<>(arrayList);
        waitList.add("item4");
        System.out.println(waitList);
        waitList.remove();
        System.out.println(waitList);
        System.out.println(waitList.isEmpty());
        System.out.println(waitList.contains("item2"));

        ArrayList<String> alTest = new ArrayList<>();
        alTest.add("item2");
        alTest.add("item3");
```

```
24          System.out.println(waitList.containsAll(alTest));
25
26          System.out.println();
27
28          BoundedWaitList<String> boundedWaitList = new BoundedWaitList<>(3);
29          boundedWaitList.add("bitem1");
30          boundedWaitList.add("bitem2");
31          boundedWaitList.add("bitem3");*/
32  //          try {
33  //              boundedWaitList.add("bitem4");
34  //          } catch (Exception e) {
35  //              e.printStackTrace();
36  //          }
37  //          System.out.println(boundedWaitList);
38
39          UnfairWaitList<Integer> unfairWaitList = new UnfairWaitList<>();
40          unfairWaitList.add(1);
41          unfairWaitList.add(2);
42          unfairWaitList.add(3);
43          unfairWaitList.add(4);
44          System.out.println(unfairWaitList);
45
46          unfairWaitList.remove(2);
47          System.out.println(unfairWaitList);
48  //          unfairWaitList.add(2);
49          unfairWaitList.remove(1);
50
51          System.out.println(unfairWaitList);
52
53          System.out.println(unfairWaitList);
54          unfairWaitList.MoveToBack(3);
55          System.out.println(unfairWaitList);
56      }
57  }
58
```

Результат выполнения программы:

```
WaitList{content=[1, 2, 3, 4]}
WaitList{content=[1, 3, 4]}
Exception in thread "main" java.lang.IllegalStateException Create breakpoint : Первый элемент не может быть удален
    at ru.mirea.lab14.UnfairWaitList.remove(UnfairWaitList.java:27)
    at ru.mirea.lab14.lab14.main(lab14.java:49)
```

Вывод: во время выполнения практической работы была освоена на практике работа с различными видами списков ожидания.

15. Практическая работа №15. Разработка интерактивных программ на языке Java с использованием паттерна MVC

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

Задание: Напишите реализацию программного кода по UML диаграмме, представленной на рисунке 15.2. Программа должна демонстрировать использование паттерна MVC.

Файл Student.java

```
package ru.mirea.lab15;

public class Student {
    private String rollNo;
    private String name;

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

    public String getRollNo() {
        return rollNo;
    }
    public void setRollNo(String rollNo) {
        this.rollNo = rollNo;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}

```

Файл StudentController.java

```

public class StudentController{
    private Student model;
    private StudentView view;

    public StudentController(Student model, StudentView view) {
        this.model = model;
        this.view = view;
    }

    public void setStudentName(String name){
        model.setName(name);
    };
    public String getStudentName(){
        return model.getName();
    };
    public void setStudentRollNo(String RollNo){
        model.setRollNo(RollNo);
    };
    public String getStudentRollNo(){
        return model.getRollNo();
    };
    public void updateView(){
        view.printStudentDetails(getStudentName(), getStudentRollNo());
    };

}

```

Файл StudentView.java

```
package ru.mirea.lab15;

public class StudentView{
    public void printStudentDetails(String studentName, String studentRollNo)
    {
        System.out.println("Student: ");
        System.out.println("Name: " + studentName);
        System.out.println("RollNo: " + studentRollNo);
    }
}

```

Файл MVCPaternDemo.java

```java
package ru.mirea.lab15;

import java.util.ArrayList;
import java.util.Objects;

public class MVCPatternDemo {
    public static void main(String[] args) {
        ArrayList<Student> students =new ArrayList<>();

        students.add(new Student("Alex", "#FAS1314"));
        students.add(new Student("Ivan", "#FAKO214"));
        students.add(new Student("Alexey", "#ASA9984"));


        Student model = retriveStudentFromDataBase(students, "Alex");
        StudentView view = new StudentView();
        StudentController controller = new StudentController(model, view);
        controller.updateView();
        System.out.println();

        model = retriveStudentFromDataBase(students, "Ivan");
        view = new StudentView();
        controller = new StudentController(model, view);
        controller.updateView();
        System.out.println();
        model = retriveStudentFromDataBase(students, "Alex");
        view = new StudentView();
        controller = new StudentController(model, view);
        System.out.println();
        controller.setStudentName("Alexander");
        controller.setStudentRollNo("#FSUBI131");
        controller.updateView();
    }

    public static Student retriveStudentFromDataBase(ArrayList<Student>
    array, String name){
        for(Student student: array){
            if (Objects.equals(student.getName(), name)){
                return student;
            }
        }
       return null;
    }
}
```

Результат выполнения программы:

```
Student:
Name: Alex
RollNo: #FAS1314

Student:
Name: Ivan
RollNo: #FAKO214


Student:
Name: Alexander
RollNo: #FSUBI131
```

Вывод: во время выполнения практической работы была освоена разработка программ с использованием событийного программирования на языке программирования Джава с использованием паттерна MVC.

16. Практическая работа №16. Исключения и работа с ними в Java

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.

Задание:

- Измените код из предыдущего примера следующим образом: Удалите throws Exception из метода getKey().
- Измените метод getKey(), добавив try-catch блок для обработки исключений.
- Добавьте цикл к getKey() таким образом, чтобы пользователь получил еще один шанс на ввод значение ключа

Замечания: Инструкция throw очень аналогична инструкции return – она прекращает выполнение метода, дальше мы не идем. Если мы нигде не ставим catch, то у нас выброс Exception очень похож на System.exit() – система завершает процесс. Но мы в любом месте можем поставить catch и, таким образом, предотвратить поломку кода.

Файл Exception1.java

```
1 package ru.mirea.lab16.task1;
2
3 public class Exception1 {
4     public void exceptionDemo() {
5         System.out.println( 2 / 0 );
6     }
7
8     public static void main(String[] args) {
```

```
        new Exception1().exceptionDemo();

    }
}

```

Файл Exception12.java

```
package ru.mirea.lab16.task1;

public class Exception1 {
    public void exceptionDemo() {
        System.out.println( 2 / 0 );
    }

    public static void main(String[] args) {
        new Exception1().exceptionDemo();

    }
}

```

Файл Exception2.java

```
package ru.mirea.lab16.task1;

public class Exception2 {
}

```

Файл ThrowsDemo.java

```
package ru.mirea.lab16.task8;

import java.util.Scanner;

public class ThrowsDemo {
    public void getKey() throws Exception {
        Scanner myScanner = new Scanner( System.in );
        boolean f = false;
        do {
            f = false;
```

```
            String key = myScanner.nextLine();
//            try {
                printDetails( key );
//            } catch (Exception e) {
//                System.out.println("Exception.. Try again");
//                f = true;
//            }
        } while (f);

    }

    public void printDetails(String key) throws Exception {
        String message = getDetails(key);
        System.out.println(message);
    }

    private String getDetails(String key) throws Exception {
        if (key.isEmpty()) {
            throw new Exception("Key set to empty string");
        }
        return "data for " + key;
    }

    public static void main(String[] args) throws Exception {
        ThrowsDemo td = new ThrowsDemo();
//        try {
            td.getKey();
//        } catch (Exception e) {
//            System.out.println("Exception");
//        }
        System.out.println("test");
    }
}
```

Файл ThrowsDemo2.java

```
package ru.mirea.lab16.task5;
public class TrowsDemo2 {
    public void getDetails(String key) {
        if(key == null) {
            throw new NullPointerException("null key in getDetails" );
        }
        System.out.println(key);
    }

    public static void main (String[]args){
        ThrowsDemo td = new ThrowsDemo();
        try {
```

```
            td.getDetails(null);
        } catch (NullPointerException e) {
            System.out.println(e.getMessage());
        }
    }
}

```

Файл ThrowsDemo.java

```
package ru.mirea.lab16.task6;

public class ThrowsDemo {
    public void printMessage(String key) {
        String message = getDetails(key);
        System.out.println(message);
    }
    public String getDetails(String key) {
        if(key == null) {
            throw new NullPointerException("null key in getDetails");
        }
        return "data for " + key;
    }
    public static void main (String[]args){
        ThrowsDemo td = new ThrowsDemo();

        try {
            td.printMessage("me!");
            td.printMessage(null);
        } catch (NullPointerException e) {
            System.out.println("NullPointerException");
        }
    }
}



```

Вывод: во время выполнения практической работы были получены навыки разработки программ, изучен синтаксис языка Java, освоены основные конструкции языка Java (циклы, условия, создание переменных и массивов, создание методов, вызов методов), а также изучены осуществление стандартного ввод/вывод данных.

17. Практическая работа №17. Создание пользовательских исключений

Цель работы: научиться создавать собственные исключения.

Задание: 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.

Файл NotValidINNException.java

```
package ru.mirea.lab17;

public class NotValidINNException extends Exception{
    public NotValidINNException(String errorMessage) {
        super(errorMessage);
    }
}

```

Файл Test.java

```
package ru.mirea.lab17;

import java.util.Scanner;

public class Test {
    public static void main(String[] args) throws NotValidINNException {
        System.out.println("ФИО: ");
        Scanner scanner = new Scanner(System.in );
        String fio = scanner.nextLine();

        System.out.println("ИНН: ");
        String inn = scanner.nextLine();

        if (inn.length() == 10 || inn.length() == 12) {
            System.out.println("ИНН действителен");
        }else {
            throw new NotValidINNException("ИНН недействителен");
```

```
18          }
19          System.out.println("test");
20      }
21  }
22
```

Результат выполнения программы:

```
ФИО:
Воробьев Александр Олегович
ИНН:
235235235
Exception in thread "main" ru.mirea.lab17.NotValidINNException: ИНН недействителен
    at ru.mirea.lab17.Test.main(Test.java:17)
```

Вывод: во время выполнения данной работы было создание собственных исключений.

18. Практическая работа №18. Работа с дженериками.

Цель работы: научиться работать с обобщенными типами в Java и применять их в программах.

Задание:

- Создать обобщенный класс с тремя параметрами (T, V, K).
- Класс содержит три переменные типа (T, V, K), конструктор, принимающий на вход параметры типа (T, V, K), методы возвращающие значения трех переменных. Создать метод, выводящий на консоль имена классов для трех переменных класса.
- Наложить ограничения на параметры типа: T должен реализовать интерфейс Comparable (классы оболочки, String), V должен реализовать интерфейс Serializable и расширять класс Animal, K

Файл Animal.java

```
package ru.mirea.lab18;

public class Animal {
    String name;

    public Animal(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

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

```

Файл Calculator.java

```java
package ru.mirea.lab18;

public class Calculator {
    public static <T extends Number, V extends Number>
    double sum(T num1, V num2) {
        return num1.doubleValue() + num2.doubleValue();
    }

    public static <T extends Number, V extends Number>
    double multiply(T num1, V num2) {
        return num1.doubleValue() * num2.doubleValue();
    }

    public static <T extends Number, V extends Number>
    double divide(T num1, V num2) {
        if (num2.doubleValue() == 0) {
            throw new ArithmeticException("Division by zero");
        }
        return num1.doubleValue() / num2.doubleValue();
    }

    public static <T extends Number, V extends Number>
    double subtraction(T num1, V num2) {
        return num1.doubleValue() - num2.doubleValue();
    }
}

```

Файл GenericClass.java

```java
package ru.mirea.lab18;

import java.io.Serializable;

public class GenericClass <T extends Comparable<T>,
V extends Animal & Serializable, K> {
    T t;
    V v;
    K k;

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

```
16
17      public T getT() {
18          return t;
19      }
20
21      public V getV() {
22          return v;
23      }
24
25      public K getK() {
26          return k;
27      }
28
29      public void printClassNames() {
30          System.out.println("Переменная 1 класса: "
31          + t.getClass().getName());
32          System.out.println("Переменная 2 класса: "
33          + v.getClass().getName());
34          System.out.println("Переменная 3 класса: "
35          + k.getClass().getName());
36      }
37  }
38
39
```

Файл MinMax.java

```
1   package ru.mirea.lab18;
2
3   public class MinMax <T extends Comparable<T>>{
4       T[] array;
5
6       public MinMax(T[] array){
7           this.array = array;
8       }
9
10      public T findMin(){
11          T min = array[0];
12          for (int i = 1; i < array.length; i++){
13              if (array[i].compareTo(min) < 0){
14                  min = array[i];
15              }
16          }
17          return min;
18      }
19
20      public T findMax(){
21          T max = array[0];
```

```
        for (int i = 1; i < array.length; i++){
            if (array[i].compareTo(max) > 0){
                max = array[i];
            }
        }
        return max;
    }
}
```

Файл Monkey.java

```
package ru.mirea.lab18;

import java.io.Serializable;

public class Monkey extends Animal implements Serializable {
    public Monkey(String name) {
        super(name);
    }
}
```

Файл Test.java

```
package ru.mirea.lab18;

public class Test {
    public static void main(String[] args) {
        GenericClass <Integer, Monkey, String> genericClass =
        new GenericClass<>(10, new Monkey("ЧиЧиЧи"), "Hello");
        genericClass.printClassNames();

        System.out.println();

        Integer[] intArray = {3, 7, 2, 9, 1};
        MinMax<Integer> minMax1 = new MinMax<>(intArray);
        System.out.println("Min: " + minMax1.findMin());
        System.out.println("Max: " + minMax1.findMax());

        System.out.println();

        Double [] doubleArray = {3.2, 7.1, 2.5, 9.2, 1.5};
```

```
19          MinMax<Double> minMax2 = new MinMax<>(doubleArray);
20          System.out.println("Min: " + minMax2.findMin());
21          System.out.println("Max: " + minMax2.findMax());
22
23          System.out.println();
24
25          System.out.println("Sum: " + Calculator.sum(5, 3.5));
26          System.out.println("Multiply: " + Calculator.multiply(5, 1.5));
27          System.out.println("Divide: " + Calculator.divide(5, 6.5));
28          System.out.println("Subtraction: " + Calculator.subtraction(5, 3.5));
29      }
30  }
31
32
```

Результат выполнения программы:

```
Переменная 1 класса: java.lang.Integer
Переменная 2 класса: ru.mirea.lab18.Monkey
Переменная 3 класса: java.lang.String

Min: 1
Max: 9

Min: 1.5
Max: 9.2

Sum: 8.5
Multiply: 7.5
Divide: 0.7692307692307693
Subtraction: 1.5
```

Вывод: во время выполнения практической работы была изучена работа с обобщенными типами в Java и применение их в программах.

19. Практическая работа №19. Стирание типов в Джава

Цель работы: научиться работать с обобщенными типами в Java и применять прием стирание типов разработке.

Задание 1: Написать метод для конвертации массива строк/чисел в список.

Файл Test.java

```java
package ru.mirea.lab19.task1;

import java.util.ArrayList;

public class Test {
    public static <T> ArrayList<T> arrayToList(T[] array){
        ArrayList<T> list = new ArrayList<>();
        for(T element : array){
            list.add(element);
        }
        return list;
    }
    public static void main(String[] args) {
        Integer[] array = {1, 2, 3, 4};
        ArrayList<Integer> list = arrayToList(array);
        System.out.println(list);


        String[] stringArray = {"Hello", "world", "its", "me"};
        ArrayList<String> stringList = arrayToList(stringArray);
        System.out.println(stringList);
    }
}

```

Результат выполнения программы:

```
[1, 2, 3, 4]
[Hello, world, its, me]
```

Задание 2: Написать класс, который умеет хранить в себе массив любых типов данных (int, long etc.).

Файл UniversalClass.java

```java
package ru.mirea.lab19.task2;

import java.util.ArrayList;

public class UniversalClass <T> {
    private ArrayList<T> array;

    public UniversalClass() {
        this.array = new ArrayList<>();
    }

    public void addElement(T element) {
        array.add(element);
    }
    public T getElement(int a) {
        return array.get(a);
    }

    public ArrayList<T> getElements() {
        return array;
    }

}

```

Файл TestUniversalClass.java

```java
package ru.mirea.lab19.task2;

public class TestUniversalClass {
    public static void main(String[] args) {
        UniversalClass<Integer> intMass = new UniversalClass<>();
        intMass.addElement(2);
        intMass.addElement(4);
        intMass.addElement(2);
        System.out.println(intMass.getElements());
        System.out.println(intMass.getElement(2));

        UniversalClass<String> strMass = new UniversalClass<>();
        strMass.addElement("Dwqwd");
```

```
        strMass.addElement("Hello");
        strMass.addElement("WOrekd");
        System.out.println(strMass.getElements());
        System.out.println(strMass.getElement(1));
    }
}
```

Результат выполнения программы:

```
[2, 4, 2]
2
[Dwqwd, Hello, WOrekd]
Hello
```

Задание 4: Написать функцию, которая сохранит содержимое каталога в список и выведет первые 5 элементов на экран.

Файл CatalogReader.java

```
package ru.mirea.lab19.task4;

import java.util.ArrayList;
import java.io.File;

public class CatalogReader {
    public static void main(String[] args) {
        ArrayList<String> catalogList = listDirectory("C:\\Program Files");
        for(int i = 0; i < 5; i++){
            System.out.println(catalogList.get(i));
        }
    }
    public static ArrayList<String> listDirectory(String path){
        ArrayList<String> list = new ArrayList<>();

        File directory = new File(path);
        if (directory.isDirectory()){
            File[] files = directory.listFiles();
            if (files != null) {
                for (File file : files) {
                    list.add(file.getName());
                }
            }
        }
        return list;
```

```
26        }
27    }
28
```

Результат выполнения программы:

```
7-Zip
Adobe
Android
Application Verifier
Blackmagic Design
```

Вывод: во время выполнения практической работы были освоены навыки работы с обобщенными типами в Java и применение приема стирания типов разработке программ на Джаве.

20. Практическая работа №20. Абстрактные типы данных. Стек

Цель работы: научиться разрабатывать программы с абстрактными типами данных на языке Джава и применять паттерн MVC при разработке программ.

Задание: Написать калькулятор для чисел с использованием RPN (Reverse Polish Notation в пер. на русск. яз. - обратной польской записи)

Файл Calculator.java

```
package ru.mirea.lab20;
import java.util.Stack;
import static java.lang.Double.parseDouble;

public class Calculator {
    Stack<Double> operStack = new Stack<>();

    public double processing(String input){
        String[] operands = input.split(" ");

        for (String element: operands){
            if (isNumeric(element)){
                operStack.push(parseDouble(element));
            }
            else{
                double y = operStack.pop();
                double x = operStack.pop();
                switch (element){
                    case "+":
                        operStack.push(x + y);
                        break;
                    case "-":
                        operStack.push(x - y);
                        break;
                    case "*":
                        operStack.push(x * y);
                        break;
                    case "/":
                        operStack.push(x / y);
                        break;
                }
```

```
            }
        }
        return operStack.pop();
    }

    private boolean isNumeric(String element) {
        try {
            parseDouble(element);
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }
}

```

Файл TestCalc.java

```
package ru.mirea.lab20;

public class TestCalc {
    public static void main(String[] args) {
        String input = "1 20 +";
        String input1 = "2 3 * 4 5 * + ";
        String input2 = "2 3 4 5 6 * + - / ";
        Calculator calc = new Calculator();
        System.out.println(calc.processing(input));
        System.out.println(calc.processing(input1));
        System.out.println(calc.processing(input2));
    }
}

```

Результат выполнения программы:

```
21.0
26.0
-0.06451612903225806
```

Вывод: во время выполнения практической работы изучена разработка программы с абстрактными типами данных на языке Джава и применение паттерна MVC при разработке программ.

21. Практическая работа №21. Абстрактные типы данных. Очередь

Цель работы: научиться разрабатывать программы с абстрактными типами данных на языке Джава.

Задание: 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).

Определите интерфейс очереди Queue и опишите его контракт. Реализуйте класс LinkedQueue — очередь на связном списке.

Выделите общие части классов LinkedQueue и ArrayQueue в базовый класс AbstractQueue.

Файл AbstractQueue.java

```
package ru.mirea.lab21;

public class AbstractQueue {
    protected int currentSize;
    public boolean isEmpty() {
        return currentSize == 0;
    }

    public int size() {
        return currentSize;
    }
}

```

Файл ArrayQueue.java

```java
package ru.mirea.lab21;

public class ArrayQueue extends AbstractQueue{
    private int[] queueArray;
    private int front;
    private int rear;
    private int maxSize;

    public ArrayQueue(int maxSize) {
        this.maxSize = maxSize;
        this.queueArray = new int[maxSize];
        this.front = 0;
        this.rear = -1;
        this.currentSize = 0;
    }

    public int element(){
        if (currentSize == 0){
            System.out.println("Очередь пуста");
            return -1;
        }
        return queueArray[front];
    }

    public void enqueue(int item){
        if (currentSize == maxSize) {
            System.out.println("Очередь переполнена");
            return;
        }
        rear = (rear + 1) % maxSize;
        queueArray[rear] = item;
        currentSize++;
    }

    public int dequeue(){
        if (currentSize == 0){
            System.out.println("Очередь пуста");
            return -1;
        }
        int item = queueArray[front];
        front = (front + 1) % maxSize;
        currentSize--;
        return item;
    }


    public void clear(){
```

```
48          front = 0;
49          rear = -1;
50          currentSize = 0;
51      }
52  }
53
```

Файл ArrayQueueATD.java

```
1   package ru.mirea.lab21;
2
3   public class ArrayQueueATD {
4       private int[] queueArray;
5       private int front;
6       private int rear;
7       private int maxSize;
8       private int currentSize;
9
10      public ArrayQueueATD(int maxSize) {
11          this.maxSize = maxSize;
12          this.queueArray = new int[maxSize];
13          this.front = 0;
14          this.rear = -1;
15          this.currentSize = 0;
16      }
17
18      public static int element(ArrayQueueATD queue){
19          if (queue.currentSize == 0){
20              System.out.println("Очередь пуста");
21              return -1;
22          }
23          return queue.queueArray[queue.front];
24      }
25
26      public static void enqueue(ArrayQueueATD queue, int item){
27          if (queue.currentSize == queue.maxSize) {
28              System.out.println("Очередь переполнена");
29              return;
30          }
31          queue.rear = (queue.rear + 1) % queue.maxSize;
32          queue.queueArray[queue.rear] = item;
33          queue.currentSize++;
34      }
35
36      public static int dequeue(ArrayQueueATD queue){
37          if (queue.currentSize == 0){
38              System.out.println("Очередь пуста");
39              return -1;
```

```
        }
        int item = queue.queueArray[queue.front];
        queue.front = (queue.front + 1) % queue.maxSize;
        queue.currentSize--;
        return item;
    }

    public static int size(ArrayQueueATD queue) {
        return queue.currentSize;
    }
    public static boolean isEmpty(ArrayQueueATD queue) {
        return queue.currentSize == 0;
    }

    public static void clear(ArrayQueueATD queue){
        queue.front = 0;
        queue.rear = -1;
        queue.currentSize = 0;
    }
}

```

Файл ArrayQueueModule.java

```
package ru.mirea.lab21;

import java.util.ArrayList;
import java.util.Objects;

public class ArrayQueueModule{
    private int[] queueArray;
    private int front;
    private int rear;
    private int maxSize;
    private int currentSize;

    public ArrayQueueModule(int maxSize) {
        this.maxSize = maxSize;
        this.queueArray = new int[maxSize];
        this.front = 0;
        this.rear = -1;
        this.currentSize = 0;
    }

    public int element(){
        if (currentSize == 0){
            System.out.println("Очередь пуста");
            return -1;
```

```
        }
        return queueArray[front];
    }

    public void enqueue(int item){
        if (currentSize == maxSize) {
            System.out.println("Очередь переполнена");
            return;
        }
        rear = (rear + 1) % maxSize;
        queueArray[rear] = item;
        currentSize++;
    }

    public int dequeue(){
        if (currentSize == 0){
            System.out.println("Очередь пуста");
            return -1;
        }
        int item = queueArray[front];
        front = (front + 1) % maxSize;
        currentSize--;
        return item;
    }

    public int size() {
        return currentSize;
    }
    public boolean isEmpty() {
        return currentSize == 0;
    }

    public void clear(){
        front = 0;
        rear = -1;
        currentSize = 0;
    }
}
```

Файл LinkedQueue.java

```
package ru.mirea.lab21;

public class LinkedQueue extends AbstractQueue{
    private Node<Integer> front;
    private Node<Integer> rear;

```

```
7
8       public LinkedQueue() {
9           front = null;
10          rear = null;
11          currentSize = 0;
12      }
13
14
15
16      public void enqueue(Integer data) {
17          Node<Integer> newNode = new Node<>(data);
18
19          if (currentSize == 0) {
20              front = newNode;
21              rear = newNode;
22          } else {
23              rear.next = newNode;
24              rear = newNode;
25          }
26
27          currentSize++;
28      }
29
30      public Integer dequeue() {
31          if (currentSize == 0){
32              System.out.println("Очередь пуста");
33              return null;
34          }
35
36          Integer data = front.data;
37          front = front.next;
38          currentSize--;
39
40          if (currentSize == 0) {
41              rear = null;
42          }
43
44          return data;
45      }
46
47      private static class Node<Integer> {
48          private Integer data;
49          private Node<Integer> next;
50
51          public Node(Integer data) {
52              this.data = data;
53              this.next = null;
54          }
55      }
56  }
```

Файл TestArrayQueue.java

```java
package ru.mirea.lab21;

public class TestArrayQueue {
    public static void main(String[] args) {
        ArrayQueue queue = new ArrayQueue(5);

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

        System.out.println(queue.dequeue() == 10);
        System.out.println(queue.dequeue() == 20);

        queue.enqueue(40);
        queue.enqueue(50);

        System.out.println(queue.dequeue() == 30);
        System.out.println(queue.dequeue() == 40);
        System.out.println(queue.dequeue() == 50);

    }
}
```

Файл TestLinkedQueue.java

```java
package ru.mirea.lab21;

public class TestLinkedQueue {
    public static void main(String[] args) {
        LinkedQueue queue = new LinkedQueue();

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

        System.out.println(queue.dequeue() == 10);
        System.out.println(queue.dequeue() == 20);

        queue.enqueue(40);
        queue.enqueue(50);
```

```
16
17          System.out.println(queue.dequeue() == 30);
18          System.out.println(queue.dequeue() == 40);
19          System.out.println(queue.dequeue() == 50);
20
21      }
22  }
23
```

Файл TestQueueATD.java

```
1   package ru.mirea.lab21;
2
3   import static ru.mirea.lab21.ArrayQueueATD.*;
4
5   public class TestQueueATD {
6       public static void main(String[] args) {
7           ArrayQueueATD queue = new ArrayQueueATD(5);
8
9           enqueue(queue,10);
10          enqueue(queue,20);
11          enqueue(queue,30);
12
13          System.out.println(dequeue(queue) == 10);
14          System.out.println(dequeue(queue) == 20);
15
16          enqueue(queue, 40);
17          enqueue(queue, 50);
18
19          System.out.println(dequeue(queue) == 30);
20          System.out.println(dequeue(queue) == 40);
21          System.out.println(dequeue(queue));
22          System.out.println(isEmpty(queue));
23      }
24  }
25
```

Файл testQueueModule.java

```
1   package ru.mirea.lab21;
2
3   public class testQueueModule {
4       public static void main(String[] args) {
5           ArrayQueueModule queue = new ArrayQueueModule(5);
6
```

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

        System.out.println(queue.dequeue() == 10);
        System.out.println(queue.dequeue() == 20);

        queue.enqueue(40);
        queue.enqueue(50);

        System.out.println(queue.dequeue() == 30);
        System.out.println(queue.dequeue() == 40);
        System.out.println(queue.dequeue() == 50);

    }
}
```

Результат выполнения программы:

```
true
true
true
true
true
```

Вывод: во время выполнения практической работы был получен навык разработки программы с абстрактными типами данных на языке Джава

22. Практическая работа №22. Паттерны проектирования, порождающие паттерны: абстрактная фабрика, фабричный метод

Цель работы: научиться применять порождающие паттерны при разработке программ на Java.

Задание: 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).

Файл AbstractChairFactory.java

```
package ru.mirea.lab22;

public interface AbstractChairFactory {
    public abstract Chair createVictorianChair();
    public abstract Chair createMagicanChair();
    public abstract Chair createFunctionalChair();
}

```

Файл Chair.java

```
package ru.mirea.lab22;

public interface Chair {
}

```

Файл ChairFactory.java

```java
package ru.mirea.lab22;

public class ChairFactory implements AbstractChairFactory{

    @Override
    public Chair createVictorianChair() {
        return new VictorianChair(0);
    }

    @Override
    public Chair createMagicanChair() {
        return new MagicChair();
    }

    @Override
    public Chair createFunctionalChair() {
        return new FunctionalChair();
    }
}

```

Файл TestFactory.java

```java
package ru.mirea.lab22;

public class TestFactory {
    public static void main(String[] args) {
        ChairFactory chairFactory = new ChairFactory();

        Chair victorianChair = chairFactory.createVictorianChair();
        Chair magicChair = chairFactory.createMagicanChair();
        Chair multifunctionalChair = chairFactory.createFunctionalChair();

        Client client = new Client();

        client.setChair(victorianChair);
        client.sit();

        client.setChair(multifunctionalChair);
        client.sit();

        client.setChair(magicChair);
        client.sit();
    }

```

```
}

```

Файл Client.java

```
package ru.mirea.lab22;

public class Client {
    Chair chair;
    public void sit(){
        System.out.println("Сижу ... " + chair);
    }
    public void setChair(Chair chair){
        this.chair = chair;
    }
}

```

Файл FunctionalChair.java

```
package ru.mirea.lab22;

public class FunctionalChair  implements Chair{
    public int sum(int a, int b){
        return (a+b);
    }

    @Override
    public String toString() {
        return "Функциональный стул";
    }
}

```

Файл VictorianChair.java

```
package ru.mirea.lab22;

public class VictorianChair implements Chair{
    private int age;

    public VictorianChair(int age) {
```

```
        this.age = age;
    }

    public int getAge() {
        return age;
    }

    @Override
    public String toString() {
        return "Викторианский стул";
    }
}
```

Файл MagicChair.java

```
package ru.mirea.lab22;

public class MagicChair  implements Chair{
    public void doMagic(){
        System.out.println("Yyy магия!");
    }
    @Override
    public String toString() {
        return "Волшебный стул";
    }
}
```

Результат выполнения программы:

```
Сижу...Викторианский стул
Сижу...Функциональный стул
Сижу...Волшебный стул
```

Вывод: во время выполнения практической работы было изучено применение порождающего паттерна при разработке программ на Java.

23. Практическая работа №23. Реализация функционала ресторана

Цель работы: ознакомиться с принципами создания динамических структур в Java, механизмом исключений и концепцией интерфейсов.

Задание: Создайте класс Drink – напитка. Класс описывает сущность – напиток и характеризуется следующими свойствами - стоимостью, названием и описанием. Класс должен быть определен как неизменяемый (Immutable class)

Создайте интерфейс Item для работы с позициями заказа. Интерфейс определяет 3 метода:

- возвращает стоимость;
- возвращает название;
- возвращает описание позиции.

Класс Drink и Dish должны реализовывать этот интерфейс. Класс Dish сделайте неизменяемым (аналогично Drink). Order должен хранить (удалять и добавлять) не только экземпляры класса Dish, но и Drink (Для этого разработайте классы Order и TablesOrderManager).

Создайте класс InternetOrder, который моделирует сущность интернет-заказ в ресторане или кафе. Класс основан на циклическом двусвязном списке с выделенной головой и может хранить как блюда, так и напитки. Внимание: список реализуется самостоятельно.

Файл MenuItem.java

```
1 package ru.mirea.lab23;
2
3 public interface MenuItem extends Comparable<MenuItem>{
```

```
    public double getCost();
    public String getName();
    public String getDescription();

    @Override
    int compareTo(MenuItem o);
}

```

Файл Dish.java

```
package ru.mirea.lab23;

public class Dish implements MenuItem {
    private double cost;
    private String name;
    private String description;

    public Dish(double cost, String name, String description) {
        this.cost = cost;
        this.name = name;
        this.description = description;
    }

    public double getCost(){
        return this.cost;
    }
    public String getName(){
        return this.name;
    }
    public String getDescription(){
        return this.description;
    }

    public void setCost(double cost) {
        this.cost = cost;
    }

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

    public void setDescription(String description) {
        this.description = description;
    }

    @Override
    public int compareTo(MenuItem o) {
```

```
38          return (int) (o.getCost() - cost);
39      }
40
41      @Override
42      public String toString() {
43          return "Dish{" +
44                  "cost=" + cost +
45                  ", name='" + name + '\'' +
46                  ", description='" + description + '\'' +
47                  '}';
48      }
49  }
50
```

Файл Drink.java

```
1   package ru.mirea.lab23;
2
3   import ru.mirea.lab24.Alcoholable;
4   import ru.mirea.lab24.DrinkTypeEnum;
5
6   public final class Drink  implements MenuItem {
7       private double cost;
8       private String name;
9       private String description;
10      public Drink(double cost, String name, String description) {
11          this.cost = cost;
12          this.name = name;
13          this.description = description;
14      }
15      public Drink(String name, String description) {
16          this.cost = 0;
17          this.name = name;
18          this.description = description;
19      }
20
21      public Drink(int cost, String name, String description) {
22          this.cost = cost;
23          this.name = name;
24          this.description = description;
25      }
26      public double getCost(){
27          return this.cost ;
28      }
29      public String getName(){
30          return this.name;
31      }
32      public String getDescription(){
```

```
        return this.description;
    }
    @Override
    public int compareTo(MenuItem o) {
        return (int) (o.getCost() - cost);
    }

    @Override
    public String toString() {
        return "Drink{" +
                "cost=" + cost +
                ", name='" + name + '\'' +
                ", description='" + description + '\'' +
                '}';
    }
}
```

Файл InternetOrder.java

```
package ru.mirea.lab23;


import java.util.Arrays;
import java.util.Objects;

public class InternetOrder {
    private int size;
    private ListNode head = null;
    private ListNode tail = null;

    public InternetOrder() {
        head = new ListNode();
    }
    public InternetOrder(MenuItem[] arr) {
        if (arr.length > 0){
            head = new ListNode();
            head.setValue(arr[0]);

            ListNode current = head;

            for(int i = 1; i < arr.length; i++){
                ListNode node = new ListNode();
                node.setValue(arr[i]);

                current.setNext(node);
                node.setPrev(current);

```

```
                current = node;
            }

            tail = current;
            tail.setNext(head);
            head.setPrev(tail);
        }
    }

//добавляющий позицию в заказ
    public boolean add(MenuItem item) {
        ListNode node = new ListNode();
        node.setValue(item);

        if (head == null) { // Если список пустой
            head = node;
            tail = node;
            head.setNext(tail);
            head.setPrev(tail);
            tail.setNext(head);
            tail.setPrev(head);
        } else {
            node.setPrev(tail);
            node.setNext(head);
            tail.setNext(node);
            head.setPrev(node);
            tail = node;
        }

        return true;
    }

//удаляющий позицию из заказа по его названию
public boolean remove(String itemName) {
    if (head == null) {
        return false; // Если список пуст, невозможно удалить элемент
    }

    ListNode current = tail;

    do {
        if (current.getValue().getName().equals(itemName)) {
            if (head == tail) {
                head = null;
                tail = null;
            } else {
                ListNode prev = current.getPrev();
                ListNode next = current.getNext();

                prev.setNext(next);
```

```
                next.setPrev(prev);

                if (current == head) {
                    head = next;
                }
                if (current == tail) {
                    tail = prev;
                }
            }

            return true; // Элемент удален успешно
        }
        current = current.getPrev();
    } while (current != tail);

    return false; // Элемент с указанным именем не найден
}


    public boolean removeAll(String itemName) {
        if (head == null) {
            return false; // Если список пуст, невозможно удалить элементы
        }

        boolean removed = false;
        ListNode current = head;

        do {
            if (current.getValue().getName().equals(itemName)) {
                if (head == tail) {
                    head = null;
                    tail = null;
                    return true; // Если в списке всего
                    один элемент и он совпадает с удаляемым
                } else {
                    ListNode prev = current.getPrev();
                    ListNode next = current.getNext();

                    prev.setNext(next);
                    next.setPrev(prev);

                    if (current == head) {
                        head = next;
                    }
                    if (current == tail) {
                        tail = prev;
                    }

                    removed = true; // Успешно удалили
                    хотя бы один элемент
```

```
                Java;
            }
            current = current.getNext();
        } while (current != head);

        return removed; // Возвращаем результат удаления
    }

    //возвращающий массив названий заказанных блюд и напитков
    public String[] itemsNames() {
        if (head == null) {
            return null; // Возвращаем пустой массив, если список пустой
        }
        ListNode current = head.getNext();
        int c = 1;

        while (current != head){
            Java;
            current = current.getNext();
        }

        String[] arr = new String[c];

        for(int j = 0; j < c; j++){
            arr[j] = current.getValue().getName();
            current = current.getNext();
        }

        return arr;
    }


    //возвращающий число заказанных блюд или напитков (принимает
    //название блюда или напитка в качестве параметра).
    public int itemQuantity(String itemName) {
        if (head == null) {
            return 0;
        }
        ListNode current = head;
        int c = 0;

        do {
            if (Objects.equals(current.getValue().getName(), itemName)){
                Java;
            }
            current = current.getNext();
        } while (current != head);

        return c;
    }
```

```
179
180 //    Возвращающий массив заказанных блюд и напитков
181     public MenuItem[] getItems() {
182         if (head == null) {
183             return new MenuItem[0]; // Возвращаем пустой массив,
184             если список пустой
185         }
186         ListNode current = head.getNext();
187         int c = 1;
188
189         while (current ≠ head){
190             Java;
191             current = current.getNext();
192         }
193
194         MenuItem[] arr = new MenuItem[c];
195
196         for(int j = 0; j < c; j++){
197             arr[j] = current.getValue();
198             current = current.getNext();
199         }
200
201         return arr;
202     }
203
204
205
206     //возвращающий массив позиций заказа, отсортированный по
207     //убыванию цены.
208     public MenuItem[] sortedItemsByCost() {
209         if (head == null) {
210             return new MenuItem[0]; // Возвращаем пустой массив,
211             если список пустой
212         }
213         ListNode current = head.getNext();
214         int c = 1;
215
216         while (current ≠ head){
217             Java;
218             current = current.getNext();
219         }
220
221         MenuItem[] arr = new MenuItem[c];
222
223         for(int j = 0; j < c; j++){
224             arr[j] = current.getValue();
225             current = current.getNext();
226         }
227         Arrays.sort(arr);
228         return arr;
```

```
    }

    public double costTotal() {
        if (head == null) {
            return 0;
        }
        double cost = head.getValue().getCost();

        ListNode current = head.getNext();
        int c = 1;

        while (current ≠ head){
            cost += current.getValue().getCost();
            current = current.getNext();
        }

        return cost;
    }

}
```

Файл ListNode.java

```
package ru.mirea.lab23;

public class ListNode {
    private ListNode next;
    private ListNode prev;
    private MenuItem value;

    public ListNode getNext() {
        return next;
    }

    public ListNode getPrev() {
        return prev;
    }

    public MenuItem getValue() {
        return value;
    }

    public void setNext(ListNode next) {
        this.next = next;
    }

    public void setPrev(ListNode prev) {
```

```
        this.prev = prev;
    }

    public void setValue(MenuItem value) {
        this.value = value;
    }
}

```

Файл TestClass.java

```
package ru.mirea.lab23;

import java.util.Arrays;

public class TestClass {
    public static void main(String[] args) {
        InternetOrder order = new InternetOrder(new MenuItem[]
        {new Dish(10, "Пицца", "pizza")});
        order.add(new Drink(90, "Пицца", "pizza"));
        System.out.println(order.itemQuantity("Пицца"));
        System.out.println(Arrays.toString(order.getItems()));
        System.out.println(Arrays.toString(order.itemsNames()));
        System.out.println(Arrays.toString(order.sortedItemsByCost()));
        order.remove("Пицца");
        System.out.println(Arrays.toString(order.itemsNames()));


    }
}

```

Результат выполнения программы:

```
2
[Dish{cost=10.0, name='Пицца', description='pizza'}, Drink{cost=90.0, name='Пицца', description='pizza'}]
[Пицца, Пицца]
[Drink{cost=90.0, name='Пицца', description='pizza'}, Dish{cost=10.0, name='Пицца', description='pizza'}]
[Пицца]
```

Вывод: во время выполнения практической работы были изучны принципы создания динамических структур в Java, механизмы исключений и концепции интерфейсов.

24. Практическая работа №24. Реализация функционала ресторана. Продолжение

Цель работы: ознакомиться с принципами создания динамических структур в Java, механизмом исключений и концепцией интерфейсов.

Задание: Переименуйте класс Order из предыдущего задания в RestaurantOrder. Создайте интерфейс Order – позиции заказа.

Замечание: Классы InternetOrder и RestaurantOrder должны реализовывать интерфейс Order. Создайте объявляемое исключение OrderAlreadyAddedExcepti выбрасываемое при попытке добавить заказ столику или по адресу, если со столиком или адресатом уже связан заказ.

Конструктор классов Drink и Dish должен выбрасывать исключение java.lang.IllegalArgumentException при попытке создать блюдо или напиток со стоимостью меньше 0, без имени или описания (если параметры имя и описание - пустые строки).

Создайте не объявляемое исключение IllegalTableNumber, выбрасываемое в методах, принимающих номер столика в качестве параметра, если столика с таким номером не существует.

Файл Alcoholable.java

```
1  package ru.mirea.lab24;
2
3  public interface Alcoholable {
4      public boolean isAlcoholicDrink();
5      public double getAlcoholVol();
6  }
7
```

Файл MenuItem.java

```java
package ru.mirea.lab24;

public interface MenuItem extends Comparable<MenuItem>{
    public double getCost();
    public String getName();
    public String getDescription();

    @Override
    int compareTo(MenuItem o);
}

```

Файл Order.java

```java
package ru.mirea.lab24;

public interface Order {
    public boolean add(MenuItem item);
    public String[] itemsNames();
    public int itemsQuantity();
    public int itemQuantity(String itemName);
    public int itemQuantity(MenuItem itemName);
    public MenuItem [] getItems();
    public boolean remove(String itemName);
    public boolean remove(MenuItem item);
    public int removeAll(String itemName);
    public int removeAll(MenuItem item);
    public MenuItem[] sortedItemsByCost();
    public double costTotal();
    public Customer getCustomer();
    public void setCustomer(Customer customer);
}

```

Файл OrdersManager.java

```java
package ru.mirea.lab24;

public interface OrdersManager {
    public int itemsQuantity(String itemName);
    public int itemsQuantity(MenuItem item);
    public Order[] getOrders();
```

```
    public double ordersCostSummary();
    public int ordersQuantity();
}

```

Файл Address.java

```
package ru.mirea.lab24;

public class Address {
    private String cityName;
    private int zipCode;
    private String streetName;
    private int buildingNumber;
    private char buildingLetter;
    private int apartmentNumber;
    public Address EMPTY_ADDRESS;

    public Address() {
        this.EMPTY_ADDRESS = new Address("", 0, "", 0, ' ', 0);
        this.cityName = EMPTY_ADDRESS.cityName;
        this.zipCode = EMPTY_ADDRESS.zipCode;
        this.streetName = EMPTY_ADDRESS.streetName;
        this.buildingNumber = EMPTY_ADDRESS.buildingNumber;
        this.buildingLetter = EMPTY_ADDRESS.buildingLetter;
        this.apartmentNumber = EMPTY_ADDRESS.apartmentNumber;
    }

    public Address(String cityName, int zipCode, String streetName,
    int buildingNumber, char buildingLetter, int apartmentNumber) {
        this.cityName = cityName;
        this.zipCode = zipCode;
        this.streetName = streetName;
        this.buildingNumber = buildingNumber;
        this.buildingLetter = buildingLetter;
        this.apartmentNumber = apartmentNumber;
    }

    public String getCityName() {
        return cityName;
    }

    public int getZipCode() {
        return zipCode;
    }

    public String getStreetName() {
        return streetName;
```

```
    }

    public int getBuildingNumber() {
        return buildingNumber;
    }

    public char getBuildingLetter() {
        return buildingLetter;
    }

    public int getApartmentNumber() {
        return apartmentNumber;
    }
}
```

Файл Customer.java

```
package ru.mirea.lab24;

public class Customer {
    private String firstName;
    private String secondName;
    private int age;
    private Address address;

    private Customer MATURE_UNKNOWN_CUSTOMER;
    private Customer NOT_MATURE_UNKNOWN_CUSTOMER;

    public Customer(int age) {
        if (age < 18){
            NOT_MATURE_UNKNOWN_CUSTOMER = new Customer("", "", age,
            new Address());
            this.firstName = NOT_MATURE_UNKNOWN_CUSTOMER.firstName;
            this.secondName = NOT_MATURE_UNKNOWN_CUSTOMER.secondName;
            this.address = NOT_MATURE_UNKNOWN_CUSTOMER.address;
        } else {
            MATURE_UNKNOWN_CUSTOMER = new Customer("", "", 18,
            new Address());
            this.firstName = MATURE_UNKNOWN_CUSTOMER.firstName;
            this.secondName = MATURE_UNKNOWN_CUSTOMER.secondName;
            this.address = MATURE_UNKNOWN_CUSTOMER.address;
        }
    }

    public Customer(String firstName, String secondName, int age,
    Address address) {
        this.firstName = firstName;
```

```
        this.secondName = secondName;
        this.age = age;
        this.address = address;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getSecondName() {
        return secondName;
    }

    public int getAge() {
        return age;
    }

    public Address getAddress() {
        return address;
    }
}

```

Файл Dish.java

```
package ru.mirea.lab24;

import java.util.Objects;

public class Dish implements MenuItem {
    private double cost;
    private String name;
    private String description;

    public Dish(double cost, String name, String description) {
        this.cost = cost;
        this.name = name;
        this.description = description;
        if (cost < 0 || Objects.equals(description, "")
        || Objects.equals(name, ""))
            throw new IllegalArgumentException();
    }

    public double getCost(){
        return this.cost;
    }
    public String getName(){
        return this.name;
```

```
    }
    public String getDescription(){
        return this.description;
    }

    public void setCost(double cost) {
        this.cost = cost;
    }

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

    public void setDescription(String description) {
        this.description = description;
    }

    @Override
    public int compareTo(MenuItem o) {
        return (int) (o.getCost() - cost);
    }

    @Override
    public String toString() {
        return "Dish{" +
                "cost=" + cost +
                ", name='" + name + '\'' +
                ", description='" + description + '\'' +
                '}';
    }
}
```

Файл Drink.java

```
package ru.mirea.lab24;

import java.util.Objects;

public final class Drink  implements MenuItem, Alcoholable{
    private double cost;
    private String name;
    private String description;
    private double alcoholVol;
    private DrinkTypeEnum type;

    public  Drink(int  cost,  String  name,  String  description,
    double alcoholVol, DrinkTypeEnum type)
```

```
14      {
15          this.cost = cost;
16          this.name = name;
17          this.description = description;
18          this.alcoholVol = alcoholVol;
19          this.type = type;
20
21          if (cost < 0 || Objects.equals(description, "") ||
22          Objects.equals(name, ""))
23              throw new IllegalArgumentException();
24      }
25
26      public Drink(double cost, String name, String description) {
27          this.cost = cost;
28          this.name = name;
29          this.description = description;
30      }
31
32      public Drink(String name, String description) {
33          this.cost = 0;
34          this.name = name;
35          this.description = description;
36      }
37
38      public Drink(int cost, String name, String description) {
39          this.cost = cost;
40          this.name = name;
41          this.description = description;
42      }
43
44      public DrinkTypeEnum getType() {
45          return type;
46      }
47
48      public double getCost(){
49          return this.cost ;
50      }
51      public String getName(){
52          return this.name;
53      }
54      public String getDescription(){
55          return this.description;
56      }
57
58      @Override
59      public boolean isAlcoholicDrink() {
60          return alcoholVol > 0.0;
61      }
62
63      @Override
```

```
64      public double getAlcoholVol() {
65          return alcoholVol;
66      }
67
68      @Override
69      public int compareTo(MenuItem o) {
70          return (int) (o.getCost() - cost);
71      }
72
73      @Override
74      public String toString() {
75          return "Drink{" +
76                  "cost=" + cost +
77                  ", name='" + name + '\'' +
78                  ", description='" + description + '\'' +
79                  '}';
80      }
81  }
82
```

Файл InternetOrder.java

```
1   package ru.mirea.lab24;
2
3   import java.util.Arrays;
4   import java.util.Objects;
5
6   public class InternetOrder implements Order{
7       private Customer customer;
8       private int size;
9       private ListNode head = null;
10      private ListNode tail = null;
11
12      public InternetOrder()
13      {
14          customer = new Customer(18);
15          size = 0;
16          head = null;
17          tail = null;
18      }
19
20      public InternetOrder(MenuItem[] arr) {
21          size = arr.length;
22          if (arr.length > 0) {
23              head = new ListNode();
24              head.setValue(arr[0]);
25
26              ListNode current = head;
```

```
27
28              for (int i = 1; i < arr.length; i++) {
29                  ListNode node = new ListNode();
30                  node.setValue(arr[i]);
31
32                  current.setNext(node);
33                  node.setPrev(current);
34
35                  current = node;
36              }
37
38              tail = current;
39              tail.setNext(head);
40              head.setPrev(tail);
41          }
42      }
43
44      //добавляющий позицию в заказ
45      @Override
46      public boolean add(MenuItem item) {
47          ListNode node = new ListNode();
48          node.setValue(item);
49
50          if (head == null) {
51              head = node;
52              tail = node;
53              head.setNext(tail);
54              head.setPrev(tail);
55              tail.setNext(head);
56              tail.setPrev(head);
57          } else {
58              node.setPrev(tail);
59              node.setNext(head);
60              tail.setNext(node);
61              head.setPrev(node);
62              tail = node;
63          }
64          size++;
65
66          return true;
67      }
68
69      //удаляющий позицию из заказа по его названию
70      @Override
71      public boolean remove(String itemName) {
72          if (head == null) {
73              return false; // Если список
74              пуст, невозможно удалить элемент
75          }
76
```

```
        ListNode current = tail;

        do {
            if (current.getValue().getName().equals(itemName)) {
                if (head == tail) {
                    head = null;
                    tail = null;
                } else {
                    ListNode prev = current.getPrev();
                    ListNode next = current.getNext();

                    prev.setNext(next);
                    next.setPrev(prev);

                    if (current == head) {
                        head = next;
                    }
                    if (current == tail) {
                        tail = prev;
                    }
                }

                return true; // Элемент удален успешно
            }
            current = current.getPrev();
        } while (current ≠ tail);

        return false; // Элемент с указанным именем не найден
    }

    @Override
    public boolean remove(MenuItem item) {
        return false;
    }

    @Override
    public int removeAll(String itemName) {
        if (head == null) {
            return 0;
        }

        int removed = 0;
        ListNode current = head;

        do {
            if (current.getValue().getName().equals(itemName)) {
                if (head == tail) {
                    head = null;
                    tail = null;
                    return 1; // Если в списке
```

```
                    всего один элемент и он совпадает с удаляемым
                } else {
                    ListNode prev = current.getPrev();
                    ListNode next = current.getNext();

                    prev.setNext(next);
                    next.setPrev(prev);

                    if (current == head) {
                        head = next;
                    }
                    if (current == tail) {
                        tail = prev;
                    }

                    removed++; // Успешно удалили хотя бы один элемент
                }
            }
            current = current.getNext();
        } while (current != head);

        return removed;
    }

    @Override
    public int removeAll(MenuItem itemName) {
        if (head == null) {
            return 0;
        }

        boolean removed = false;
        ListNode current = head;

        do {
            if (current.getValue().equals(itemName)) {
                if (head == tail) {
                    head = null;
                    tail = null;
                    return 1;
                } else {
                    ListNode prev = current.getPrev();
                    ListNode next = current.getNext();

                    prev.setNext(next);
                    next.setPrev(prev);

                    if (current == head) {
                        head = next;
                    }
                    if (current == tail) {
```

```
                    tail = prev;
                }

                removed = true;
            }
        }
        current = current.getNext();
    } while (current != head);

    return 0;
}

//возвращающий массив названий заказанных блюд и напитков
@Override
public String[] itemsNames() {
    if (head == null) {
        return null;
    }
    ListNode current = head.getNext();
    int c = 1;

    while (current != head) {
        Java;
        current = current.getNext();
    }

    String[] arr = new String[c];

    for (int j = 0; j < c; j++) {
        arr[j] = current.getValue().getName();
        current = current.getNext();
    }

    return arr;
}

@Override
public int itemsQuantity() {
    return size;
}


//возвращающий число заказанных блюд или напитков (принимает
//название блюда или напитка в качестве параметра).
@Override
public int itemQuantity(String itemName) {
    if (head == null) {
        return 0;
    }
    ListNode current = head;
```

```
        int c = 0;

        do {
            if (Objects.equals(current.getValue().getName(), itemName)) {
                Java;
            }
            current = current.getNext();
        } while (current ≠ head);

        return c;
    }



    @Override
    public int itemQuantity(MenuItem itemName) {
        if (head == null) {
            return 0;
        }
        ListNode current = head;
        int c = 0;

        do {
            if (Objects.equals(current.getValue(), itemName)) {
                Java;
            }
            current = current.getNext();
        } while (current ≠ head);

        return c;
    }

    //    Возвращающий массив заказанных блюд и напитков
    @Override
    public MenuItem[] getItems() {
        if (head == null) {
            return new MenuItem[0]; // Возвращаем
            пустой массив, если список пустой
        }
        ListNode current = head.getNext();
        int c = 1;

        while (current ≠ head) {
            Java;
            current = current.getNext();
        }

        MenuItem[] arr = new MenuItem[c];

        for (int j = 0; j < c; j++) {
```

```
277             arr[j] = current.getValue();
278             current = current.getNext();
279         }
280 
281         return arr;
282     }
283 
284 
285     //возвращающий массив позиций заказа,
286     отсортированный по убыванию цены.
287 
288     @Override
289     public MenuItem[] sortedItemsByCost() {
290         if (head == null) {
291             return new MenuItem[0]; // Возвращаем
292             пустой массив, если список пустой
293         }
294         ListNode current = head.getNext();
295         int c = 1;
296 
297         while (current ≠ head) {
298             Java;
299             current = current.getNext();
300         }
301 
302         MenuItem[] arr = new MenuItem[c];
303 
304         for (int j = 0; j < c; j++) {
305             arr[j] = current.getValue();
306             current = current.getNext();
307         }
308         Arrays.sort(arr);
309         return arr;
310     }
311 
312     @Override
313     public double costTotal() {
314         if (head == null) {
315             return 0;
316         }
317         double cost = head.getValue().getCost();
318 
319         ListNode current = head.getNext();
320         int c = 1;
321 
322         while (current ≠ head) {
323             cost += current.getValue().getCost();
324             current = current.getNext();
325         }
326 
```

```
327         return cost;
328     }
329 
330     @Override
331     public Customer getCustomer() {
332         return customer;
333     }
334 
335     @Override
336     public void setCustomer(Customer customer) {
337         this.customer = customer;
338     }
339 
340 }
341 
```

Файл IllegalTableNumber.java

```
1  package ru.mirea.lab24;
2  
3  public class IllegalTableNumber extends Exception{
4      public IllegalTableNumber(String message) {
5          super(message);
6      }
7  }
8  
```

Файл InternetOrdersManager.java

```
1  package ru.mirea.lab24;
2  
3  public class InternetOrdersManager implements OrdersManager{
4      private QueueNode head;
5      private QueueNode tail;
6      private int size;
7  
8      @Override
9      public int itemsQuantity(String itemName) {
10         int count = 0;
11         QueueNode current = head;
12 
13         while (current ≠ null)
14         {
15             count += current.getValue().itemQuantity(itemName);
16             current = current.getNext();
```

```
        }

        return count;
    }

    @Override
    public int itemsQuantity(MenuItem item) {
        int count = 0;
        QueueNode current = head;

        while (current != null)
        {
            count += current.getValue().itemQuantity(item);
            current = current.getNext();
        }

        return count;
    }

    @Override
    public Order[] getOrders() {
        Order[] orders = new Order[size];

        QueueNode current = head;
        int count = 0;

        while (current != null)
        {
            orders[count] = current.getValue();
            current = current.getNext();
            count++;
        }

        return orders;
    }

    @Override
    public double ordersCostSummary() {
        double total = 0;
        QueueNode current = head;

        while (current != null)
        {
            total += current.getValue().costTotal();
            current = current.getNext();
        }

        return total;
    }

```

```
67      @Override
68      public int ordersQuantity() {
69          int sum = 0;
70          QueueNode current = head;
71
72          while (current != null)
73          {
74              sum += 1;
75              current = current.getNext();
76          }
77
78          return sum;
79      }
80
81      public boolean add(Order newOrder) {
82          QueueNode newNode = new QueueNode(null, null, newOrder);
83          try {
84              for (Order order: this.getOrders()){
85                  if (order.getCustomer().getAddress()
86                  == newOrder.getCustomer().getAddress()){
87                      throw new OrderAlreadyAddedException("");
88                  }
89              }
90          } catch (OrderAlreadyAddedException e){
91              System.out.println("На этот адресс уже оформлен заказ");
92          }
93
94          if (head == null)
95          {
96              head = newNode;
97          }
98          else
99          {
100             tail.setNext(newNode);
101             newNode.setPrev(tail);
102         }
103         tail = newNode;
104
105         size++;
106
107         return true;
108     }
109
110     public Order remove()
111     {
112         QueueNode del = head;
113         head = head.getNext();
114         return del.getValue();
115     }
116
```

```
117     public Order order()
118     {
119         return head.getValue();
120     }
121 }
122
```

Файл ListNode.java

```
1  package ru.mirea.lab24;
2
3  public class ListNode {
4      private ListNode next;
5      private ListNode prev;
6      private MenuItem value;
7
8      public ListNode getNext() {
9          return next;
10     }
11
12     public ListNode getPrev() {
13         return prev;
14     }
15
16     public MenuItem getValue() {
17         return value;
18     }
19
20     public void setNext(ListNode next) {
21         this.next = next;
22     }
23
24     public void setPrev(ListNode prev) {
25         this.prev = prev;
26     }
27
28     public void setValue(MenuItem value) {
29         this.value = value;
30     }
31 }
32
```

Файл QueueNode.java

```java
package ru.mirea.lab24;

public class QueueNode {
    private QueueNode next;
    private QueueNode prev;
    private Order value;

    public QueueNode(QueueNode next, QueueNode prev, Order value) {
        this.next = next;
        this.prev = prev;
        this.value = value;
    }

    public QueueNode getNext() {
        return next;
    }

    public void setNext(QueueNode next) {
        this.next = next;
    }

    public QueueNode getPrev() {
        return prev;
    }

    public void setPrev(QueueNode prev) {
        this.prev = prev;
    }

    public Order getValue() {
        return value;
    }

    public void setValue(Order value) {
        this.value = value;
    }
}

```

Файл TableOrder.java

```java
package ru.mirea.lab24;

public class TableOrder implements Order{
    private Customer customer;
```

```
    private int size;
    private MenuItem[] items;

    public TableOrder()
    {
        customer = new Customer(18);
        items = new MenuItem[10];
    }

    @Override
    public boolean add(MenuItem item) {
        if (size == items.length)
        {
            MenuItem[] newItems = new MenuItem[size * 2];
            System.arraycopy(items, 0, newItems, 0, size);
            items = newItems;
        }

        if (item instanceof Drink && ((Drink)item).isAlcoholicDrink())
        {
            if (customer.getAge() >= 18)
            {
                items[size] = item;
                size++;
            }
        }
        else
        {
            items[size] = item;
            size++;
        }

        return true;
    }

    @Override
    public String[] itemsNames() {
        if (size == 0) {
            return null;
        }

        String[] arr = new String[size];

        for (int j = 0; j < size; j++) {
            arr[j] = items[j].getName();
        }

        return arr;
    }

```

```
55      @Override
56      public int itemsQuantity() {
57          return size;
58      }
59
60      @Override
61      public int itemQuantity(String itemName) {
62          int quantity = 0;
63
64          for (int i = 0; i < size; i++)
65          {
66              if (items[i].getName().equals(itemName))
67              {
68                  quantity++;
69              }
70          }
71
72          return quantity;
73      }
74
75      @Override
76      public int itemQuantity(MenuItem itemName) {
77          int quantity = 0;
78
79          for (MenuItem item : items)
80          {
81              if (item == itemName)
82              {
83                  quantity++;
84              }
85          }
86
87          return quantity;
88      }
89
90      @Override
91      public MenuItem[] getItems() {
92          return items;
93      }
94
95      @Override
96      public boolean remove(String itemName) {
97          for (int i = size - 1; i >= 0; i--)
98          {
99              if (items[i].getName().equals(itemName))
100             {
101                 items[i] = null;
102                 System.arraycopy(items, i + 1, items, i, size - i);
103                 size--;
104
```

```
105                 return true;
106             }
107         }
108 
109         return false;
110     }
111 
112     @Override
113     public boolean remove(MenuItem item) {
114         for (int i = size - 1; i >= 0; i--)
115         {
116             if (items[i].equals(item))
117             {
118                 items[i] = null;
119                 System.arraycopy(items, i + 1, items, i, size - i);
120                 size--;
121 
122                 return true;
123             }
124         }
125 
126         return false;
127     }
128 
129     @Override
130     public int removeAll(String itemName) {
131         int count = 0;
132 
133         for (int i = size - 1; i >= 0; i--)
134         {
135             if (items[i].getName().equals(itemName))
136             {
137                 items[i] = null;
138                 System.arraycopy(items, i + 1, items, i, size - i);
139                 count++;
140             }
141         }
142 
143         size -= count;
144         return count;
145     }
146 
147     @Override
148     public int removeAll(MenuItem item) {
149         int count = 0;
150 
151         for (int i = size - 1; i >= 0; i--)
152         {
153             if (items[i].equals(item))
154             {
```

```
                items[i] = null;
                System.arraycopy(items, i + 1, items, i, size - i);
                count++;
            }
        }

        size -= count;
        return count;
    }

    @Override
    public MenuItem[] sortedItemsByCost() {
        MenuItem[] sortedItems = new MenuItem[size];
        System.arraycopy(items, 0, sortedItems, 0, size);

        for (int i = 0; i < size - 1; i++)
        {
            for (int j = i + 1; j < size; j++)
            {
                if (sortedItems[j].getCost() > sortedItems[i].getCost())
                {
                    MenuItem temp = sortedItems[i];
                    sortedItems[i] = sortedItems[j];
                    sortedItems[j] = temp;
                }
            }
        }

        return sortedItems;
    }

    @Override
    public double costTotal() {
        double totalCost = 0;

        for (int i = 0; i < size; i++) {
            totalCost += items[i].getCost();
        }

        return totalCost;
    }

    @Override
    public Customer getCustomer() {
        return customer;
    }

    @Override
    public void setCustomer(Customer customer) {
        this.customer = customer;
```

```
    }
}

```

Файл OrderAlreadyAddedException.java

```
package ru.mirea.lab24;

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

Файл TableOrdersManager.java

```
package ru.mirea.lab24;
import java.util.HashMap;
public class TableOrdersManager implements OrdersManager{
    private Order[] orders;

    private final int NUMBER_OF_TABLES = 20;

    public TableOrdersManager()
    {
        orders = new Order[NUMBER_OF_TABLES];
    }

    public void add(Order order, int tableNumber) {
        try {
            if (tableNumber ⩾ 0 && tableNumber < NUMBER_OF_TABLES) {
                if (orders[tableNumber] == null)
                    orders[tableNumber] = order;
                else throw new OrderAlreadyAddedException("");
            } else if (tableNumber < 0 || tableNumber > NUMBER_OF_TABLES)
                throw new IllegalTableNumber("");
        } catch (OrderAlreadyAddedException e) {
            System.out.println("Столик с таким номером уже заказан");
        } catch (IllegalTableNumber e) {
            System.out.println("Столика с таким номером нет");
        }
    }

    public void addItem(MenuItem item, int tableNumber){
        if (tableNumber ⩾ 0 && tableNumber <
```

```
30          NUMBER_OF_TABLES && orders[tableNumber] != null) {
31              orders[tableNumber].add(item);
32          }
33      }
34      public int freeTableNumber(){
35
36          for (int i = 0; i < orders.length; i++)
37          {
38              if (orders[i] == null)
39              {
40                  return i;
41              }
42          }
43          return -1;
44      }
45
46      public int[] freeTableNumbers(){
47          HashMap<Integer, Boolean> map = new HashMap<>();
48
49          for (int i = 0; i < orders.length; i++)
50          {
51              if (orders[i] == null)
52              {
53                  map.put(i, true);
54              }
55          }
56
57          return map.keySet().stream().mapToInt(Integer::intValue).toArray();
58      }
59
60      public Order getOrder(int tableNumber){
61
62          return orders[tableNumber];
63      }
64
65      public void remove(int tableNumber){
66          if (tableNumber >= 0 && tableNumber < NUMBER_OF_TABLES)
67          {
68              orders[tableNumber] = null;
69          }
70      }
71
72      public void remove(Order order){
73          for (int i = 0; i < orders.length; i++)
74          {
75              if (orders[i] == order) {
76                  orders[i] = null;
77                  break;
78              }
79          }
```

```
80      }
81
82      public void removeAll(Order order){
83          for (int i = 0; i < orders.length; i++)
84          {
85              if (orders[i] == order)
86              {
87                  orders[i] = null;
88                  System.arraycopy(orders, i + 1, orders, i, orders.length - 1);
89              }
90          }
91
92      }
93
94      @Override
95      public int itemsQuantity(String itemName)
96      {
97          int quantity = 0;
98
99          for (Order order : orders)
100         {
101             if (order != null)
102             {
103                 quantity += order.itemQuantity(itemName);
104             }
105         }
106
107         return quantity;
108     }
109
110     @Override
111     public int itemsQuantity(MenuItem item)
112     {
113         int quantity = 0;
114
115         for (Order order : orders)
116         {
117             if (order != null)
118             {
119                 quantity += order.itemQuantity(item);
120             }
121         }
122
123         return quantity;
124     }
125
126     @Override
127     public Order[] getOrders() {
128         return orders;
129     }
```

```
130
131     @Override
132     public double ordersCostSummary()
133     {
134         double cost = 0;
135
136         for (Order order : orders)
137         {
138             if (order =≠ null)
139             {
140                 cost += order.costTotal();
141             }
142         }
143
144         return cost;
145     }
146
147     @Override
148     public int ordersQuantity() {
149         int quantity = 0;
150
151         for (Order order : orders)
152         {
153             if (order =≠ null)
154             {
155                 quantity++;
156             }
157         }
158
159         return quantity;
160     }
161 }
162
```

Файл DrinkTypeEnum.java

```
1  package ru.mirea.lab24;
2
3  public enum DrinkTypeEnum {
4      BEER,
5      WINE,
6      VODKA,
7      BRANDY,
8      CHAMPAGNE,
9      WHISKEY,
10     TEQUILA,
11     RUM,
12     VERMUTH,
```

```
    LIQUOR,
    JAGERMEISTER,
    JUICE,
    COFEE,
    GREEN_TEA,
    BLACK_TEA,
    MILK,
    WATER,
    SODA;
}

```

Файл TestBook.java

```
package ru.mirea.lab24;

import java.util.Arrays;

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

        TableOrdersManager ordersManager = new TableOrdersManager();

        Dish dish1 = new Dish(90, "Пицца", "Пицца");
        Dish dish2 = new Dish(60, "Хачапури" , "Хачапури по-аджарски");
        Dish dish3 = new Dish(160, "Сендвич" , "Сендвич с ветчиной и сыром");

        Drink drink1 = new Drink(90, "Сок", "Виноградный сок",
        0.0, DrinkTypeEnum.JUICE);
        Drink drink2 = new Drink(160, "Пиво", "Пиво",
        0.2, DrinkTypeEnum.BEER);
        Drink drink3 = new Drink(150, "Водка", "Водка",
        0.35, DrinkTypeEnum.VODKA);

        Order order1 = new TableOrder();
        order1.add(dish1);
        order1.add(drink2);
        order1.setCustomer(new Customer("Артуро", "Галлиярди", 31,
        new Address()));
        ((TableOrdersManager)ordersManager).add(order1, 1);

        Order order2 = new TableOrder();
        order2.add(dish2);
        order2.add(dish2);
        order2.add(dish1);
        order2.add(drink3);
        order2.setCustomer(new Customer("Элиза", "Маркс", 15,
        new Address()));
```

```
35          ordersManager.add(order2, 2);
36
37          Order order3 = new TableOrder();
38          order3.add(dish1);
39          order3.add(dish3);
40          order3.setCustomer(new  Customer("Петя",  "Петров",  25,
41          new Address()));
42          ordersManager.add(order3, 3);
43
44
45          ordersManager.addItem(drink1, 1);
46          ordersManager.addItem(dish3, 1);
47
48          ordersManager.addItem(dish3, 2);
49          ordersManager.addItem(drink3, 2); // элиза заказывает водку ...
50
51          ordersManager.addItem(dish2, 3);
52          ordersManager.addItem(drink3, 3);
53
54
55          InternetOrdersManager internetOrdersManager =
56          new InternetOrdersManager();
57
58          Order internetOrder1 = new InternetOrder();
59          internetOrder1.add(drink3);
60          internetOrder1.add(dish3);
61          internetOrder1.setCustomer(new  Customer("Вася",  "Петров",
62          29,  new Address()));
63          internetOrdersManager.add(internetOrder1);
64
65
66          for (Order order : ordersManager.getOrders())
67          {
68              if (order =≠ null)
69              {
70                  System.out.println("Заказ  "  +
71                  order.getCustomer().getFirstName()  +  "  "  + order.getCustomer().
72                          + order.getCustomer().getAge() + ": ");
73                  System.out.print("{ ");
74                  for (String names : order.itemsNames()) {
75                      System.out.print(names + " ");
76                  }
77                  System.out.println("}");
78              }
79          }
80
81          for (Order order : internetOrdersManager.getOrders())
82          {
83              if (order =≠ null)
84              {
```

```
85                  System.out.println("Заказ  "  +
86                  order.getCustomer().getFirstName()  +  "  "  + order.getCustomer().
87                          + order.getCustomer().getAge() + ": ");
88                  System.out.print("{ ");
89                  for (String names : order.itemsNames()) {
90                      System.out.print(names + " ");
91                  }
92                  System.out.println("}");
93              }
94          }
95          System.out.println();
96
97
98          System.out.println("Количество  Сендвичей  в  заказах:  "
99          + ordersManager.itemsQuantity("Сендвич"));
100         System.out.println("Количество  Хачапури  в  заказах:  "
101         + ordersManager.itemsQuantity("Хачапури"));
102         System.out.println("Количество  Сока  в  заказах:  "
103         + ordersManager.itemsQuantity("Сок"));
104         System.out.println();
105
106         System.out.println("Количество  заказов:  "
107         + (ordersManager.ordersQuantity()+
108         internetOrdersManager.ordersQuantity()));
109         System.out.println("Сумарная  стоимость  заказов:  "
110         + (ordersManager.ordersCostSummary() +
111         internetOrdersManager.ordersCostSummary()));
112         System.out.println();
113
114
115         ordersManager.remove(order1);
116         ordersManager.remove(1);
117
118         ordersManager.remove(3);
119         ordersManager.remove(2);
120
121         System.out.println("Количество  заказов:  "
122         + ordersManager.ordersQuantity());
123         System.out.println();
124
125         System.out.println("Номера свободных столов: ");
126         for  (int  table  :
127             ordersManager.freeTableNumbers())
128         {
129             System.out.print(table + " ");
130         }
131     }
132 }
133
```

Результат выполнения программы:

```
Заказ  Артуро  Галлиярди  31:
{ Пицца Сок Сендвич }
Заказ  Элиза  Маркс  15:
{ Хачапури Хачапури Пицца Сендвич }
Заказ  Петя  Петров  25:
{ Пицца Сендвич Хачапури Водка }
Заказ  Вася  Петров  29:
{ Водка Сендвич }

Количество  Сендвичей  в  заказах:  3
Количество  Хачапури  в  заказах:  3
Количество  Сока  в  заказах:  1

Количество  заказов:  4
Сумарная  стоимость  заказов:  1480.0

Количество  заказов:  0

Номера свободных столов:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
```

Вывод: во время выполнения практической работы были изучны принципы создания динамических структур в Java, механизмы исключений и концепции интерфейсов.

СПИСОК ЛИТЕРАТУРЫ

1. Роберт Лафоре, Структуры данных и алгоритмы в Java

2. В. Белов, В. Чистякова Алгоритмы и структуры данных.Учебник Инфра-м 2016 240 стр. ISBN: 5906818256

3. Богачев К.Ю. Основы параллельного программирования– М.: Бином. Лаборатория знаний, 2003. -.342 с.

4. Зыль С.Н. Операционная система реального времени QNX Neutrino: от теории кпрактике – Изд. 2-е - СПб.: БХВПетербург, 2004. – 192 с.

5. Зорина Н.В. Курс лекций по Объектно-ориентированному программированию на Java, МИРЭА, Москва, 2016

6. Программирование на языке Java: работа со строками и массивами. Методические указания. [Электронный ресурс] : Учебно-методические пособия — Электрон. дан. — СПб. : ПГУПС, 2015. — 24 с

7. Кожомбердиева, Г.И. Программирование на языке Java: создание графического интерфейса пользователя: учеб. пособие. [Электронный ресурс] : Учебные пособия / Г.И. Кожомбердиева, М.И. Гарина. — Электрон. дан. — СПб.: ПГУПС, 2012. — 67 с.

8. Вишневская, Т.И. Технология программирования. Часть 1. [Электронный ресурс] / Т.И. Вишневская, Т.Н. Романова. — Электрон. дан. — М. : МГТУ им. Н.Э. Баумана, 2007. — 59 с.