Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»

Институт информатики и кибернетики

Отчет по лабораторной работе №5

Дисциплина: «ООП»

Тема «Методы класса “Object”»

Выполнил: Пантелеев Ю.В.

Группа: 6201-120303

Самара, 2025

Задание на лабораторную работу

Задание 1

Я переопределил в классе FunctionPoint следующие методы:

public String toString()

```
public String toString(){  new *
    return  String.format("(%.2f; %.2f)",this.x, this.y);
}
```

public boolean equals(Object o)

```
public boolean equals(Object o) {  new *
    if (o == this)
        return true;
    if (o == null || getClass() != o.getClass())
        return false;

    FunctionPoint FFo = (FunctionPoint) o;

    return (Math.abs(this.x-FFo.getX()) < 1e-10) && (Math.abs(this.y-((FunctionPoint) o).getY()) < 1e-10);
}
```

public int hashCode()

```
public int hashCode() {  new *
    long xBits = Double.doubleToLongBits(x);
    long yBits = Double.doubleToLongBits(y);

    int xHash = (int) (xBits ^ (xBits >>> 32));
    int yHash = (int) (yBits ^ (yBits >>> 32));

    return xHash ^ yHash;
}
```

public Object clone()

```
public Object clone(){  new *
    return new FunctionPoint(this.x, this.y);
}
```

Задание 2

Я переопределил в классе ArrayTabulatedFunction следующие методы:

public String toString()

```
public String toString(){  new *
    StringBuilder sb = new StringBuilder();
    sb.append("{");
    for (int i = 0; i < pointsCount; i++) {
        sb.append(points[i].toString());
        if (i < pointsCount - 1) {
            sb.append(", ");
        }
    }
    sb.append("}");
    return sb.toString();
}
```

public boolean equals(Object o)

```
public boolean equals(Object o){  new *
    if (o == this) return true;

    if (o instanceof ArrayTabulatedFunction) {
        ArrayTabulatedFunction other = (ArrayTabulatedFunction) o;

        if (this.pointsCount != other.pointsCount) {
            return false;
        }

        for (int i = 0; i < pointsCount; i++) {
            if (!this.points[i].equals(other.points[i])) {
                return false;
            }
        }
        return true;
    }

    if (!(o instanceof TabulatedFunction)) {
        return false;
    }

    TabulatedFunction other = (TabulatedFunction) o;
    if (this.pointsCount != other.getPointsCount()) {
        return false;
    }

    for (int i = 0; i < pointsCount; i++) {
        if (!this.getPoint(i).equals(other.getPoint(i))) {
            return false;
        }
    }
    return true;
}
```

public int hashCode()

```
public int hashCode() {  new *
    int hash = pointsCount;

    for (int i = 0; i < pointsCount; i++) {
        hash ^= points[i].hashCode();
    }

    return hash;
}
```

public Object clone()

```
public Object clone(){  new *
    FunctionPoint[] ClonePoints = new FunctionPoint[pointsCount];

    for(int i = 0; i < pointsCount; i++){
        ClonePoints[i] = (FunctionPoint) points[i].clone();
    }

    return new ArrayTabulatedFunction(ClonePoints);
}
```

Задание 3

Я переопределил в классе LinkedListTabulatedFunction следующие методы:

public String toString()

```
public String toString(){  new *
    StringBuilder str = new StringBuilder();
    str.append("{");

    FunctionNode currentNode = head.next;
    for (int i = 0; i < pointsCount; i++){
        str.append(currentNode.point.toString());
        currentNode = currentNode.next;
        if (i<pointsCount-1){
            str.append(", ");
        }
    }
    return str.toString();
}
```

public boolean equals(Object o)

```java
public boolean equals(Object o) {
    if (o == this)
        return true;
    if (!(o instanceof TabulatedFunction)) {
        return false;
    }

    FunctionNode currentNode = head.next;
    if (o instanceof LinkedListTabulatedFunction){
        LinkedListTabulatedFunction LLTFo = (LinkedListTabulatedFunction) o;
        FunctionNode currentNodeLLTFo = LLTFo.head.next;
        if (LLTFo.pointsCount != this.pointsCount)
            return false;
        for (int i = 0; i < pointsCount; i++){
            if (!currentNode.point.equals(currentNodeLLTFo.point))
                return false;
            currentNode = currentNode.next;
            currentNodeLLTFo = currentNodeLLTFo.next;
        }
        return true;
    }

    else{
        TabulatedFunction TFo = (TabulatedFunction) o;
        if (TFo.getPointsCount() != this.pointsCount)
            return false;
        for(int i = 0; i < pointsCount; i++){
            if(!currentNode.point.equals(TFo.getPoint(i)))
                return false;
            currentNode = currentNode.next;
        }
    }
    return true;
}
```

public int hashCode()

```
public int hashCode(){  new *
    int hash = pointsCount;

    FunctionNode currentNode = head.next;
    for (int i = 0; i < pointsCount; i++){
        hash ^= currentNode.point.hashCode();
        currentNode = currentNode.next;
    }
    return hash;
}
```

public Object clone()

```
public Object clone(){  new *
    FunctionPoint[] ClonePoints = new FunctionPoint[pointsCount];

    FunctionNode currentNode = head.next;
    for(int i = 0; i < pointsCount; i++){
        ClonePoints[i] = (FunctionPoint) currentNode.point.clone();
        currentNode = currentNode.next;
    }

    return new LinkedListTabulatedFunction(ClonePoints);
}
```

Задание 4

Я сделал так, чтобы все объекты типа TabulatedFunction были клонируемыми с точки зрения JVM.

Интерфейс TabulatedFunction:

```
package functions;

public interface TabulatedFunction extends Function{  13 usages  2 implementations  Юрий *
    public int getPointsCount();  7 usages  2 implementations  Юрий
    public FunctionPoint getPoint(int index);  3 usages  2 implementations  Юрий
    public void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException;
    public double getPointX(int index);  2 usages  2 implementations  Юрий
    public void setPointX(int index, double x) throws InappropriateFunctionPointException;  no usages
    public double getPointY(int index);  6 usages  2 implementations  Юрий
    public void setPointY(int index, double y);  2 usages  2 implementations  Юрий
    public void deletePoint(int index);  no usages  2 implementations  Юрий
    public void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;  1 usage  2 im
    public Object clone();  2 implementations  new *
}
```

Задание 5

Я проверил работу написанных методов:

```java
import functions.*;
import functions.basic.*;

public class Main {  new *
    public static void main(String[] args) throws Exception {  new *
        System.out.println("=== ЛАБОРАТОРНАЯ РАБОТА №5: ПЕРЕОПРЕДЕЛЕНИЕ МЕТОДОВ OBJECT ===\n");

        testTask5Part1();
        testTask5Part2();
        testTask5Part3();
        testTask5Part4();

        System.out.println("\n=== ВСЕ ТЕСТЫ ВЫПОЛНЕНЫ ===");
    }

    public static void testTask5Part1() {  1 usage  new *
        System.out.println("1. ТЕСТИРОВАНИЕ МЕТОДОВ FunctionPoint:");

        FunctionPoint p1 = new FunctionPoint( x: 1.5,  y: 2.5);
        FunctionPoint p2 = new FunctionPoint( x: 1.5,  y: 2.5);
        FunctionPoint p3 = new FunctionPoint( x: 1.5,  y: 2.6);
        FunctionPoint p4 = new FunctionPoint( x: 1.6,  y: 2.5);

        System.out.println("   Точка p1: " + p1.toString());
        System.out.println("   Точка p2: " + p2.toString());
        System.out.println("   Точка p3: " + p3.toString());
        System.out.println("   Точка p4: " + p4.toString());

        System.out.println("\n   Тестирование equals():");
        System.out.println("   p1.equals(p2): " + p1.equals(p2) + " (ожидается true)");
        System.out.println("   p1.equals(p3): " + p1.equals(p3) + " (ожидается false)");
        System.out.println("   p1.equals(p4): " + p1.equals(p4) + " (ожидается false)");
        System.out.println("   p1.equals(null): " + p1.equals(null) + " (ожидается false)");
        System.out.println("   p1.equals(\"строка\"): " + p1.equals("строка") + " (ожидается false)");

        System.out.println("\n   Тестирование hashCode():");
        System.out.println("   p1.hashCode(): " + p1.hashCode());
        System.out.println("   p2.hashCode(): " + p2.hashCode());
        System.out.println("   p3.hashCode(): " + p3.hashCode());
        System.out.println("   p4.hashCode(): " + p4.hashCode());

        System.out.println("   p1.hashCode() == p2.hashCode(): " +
```

```
            (p1.hashCode() == p2.hashCode()) + " (ожидается true для равных объектов)");

        System.out.println("\n   Тестирование clone():");
        FunctionPoint p1Clone = (FunctionPoint) p1.clone();
        System.out.println("   p1: " + p1);
        System.out.println("   clone: " + p1Clone);
        System.out.println("   p1.equals(clone): " + p1.equals(p1Clone));
        System.out.println("   p1 == clone: " + (p1 == p1Clone) + " (ожидается false)");

        p1Clone.setX(10.0);
        System.out.println("   После изменения clone.x = 10.0:");
        System.out.println("   p1.x: " + p1.getX() + " (ожидается 1.5)");
        System.out.println("   clone.x: " + p1Clone.getX() + " (ожидается 10.0)");
    }

    public static void testTask5Part2() throws Exception {  1 usage  new *
        System.out.println("\n\n2. ТЕСТИРОВАНИЕ МЕТОДОВ ArrayTabulatedFunction:");

        FunctionPoint[] points1 = {
                new FunctionPoint( x: 0,  y: 0),
                new FunctionPoint( x: 1,  y: 1),
                new FunctionPoint( x: 2,  y: 4),
                new FunctionPoint( x: 3,  y: 9)
        };

        FunctionPoint[] points2 = {
                new FunctionPoint( x: 0,  y: 0),
                new FunctionPoint( x: 1,  y: 1),
                new FunctionPoint( x: 2,  y: 4),
                new FunctionPoint( x: 3,  y: 9)
        };

        FunctionPoint[] points3 = {
                new FunctionPoint( x: 0,  y: 0),
                new FunctionPoint( x: 1,  y: 1),
                new FunctionPoint( x: 2,  y: 4)
        };

        ArrayTabulatedFunction func1 = new ArrayTabulatedFunction(points1);
        ArrayTabulatedFunction func2 = new ArrayTabulatedFunction(points2);
```

```
        ArrayTabulatedFunction func3 = new ArrayTabulatedFunction(points3);

        System.out.println("   Функция func1: " + func1.toString());
        System.out.println("   Функция func2: " + func2.toString());
        System.out.println("   Функция func3: " + func3.toString());

        System.out.println("\n   Тестирование equals():");
        System.out.println("   func1.equals(func2): " + func1.equals(func2) + " (ожидается true)");
        System.out.println("   func1.equals(func3): " + func1.equals(func3) + " (ожидается false)");
        System.out.println("   func1.equals(null): " + func1.equals(null) + " (ожидается false)");
        System.out.println("   func1.equals(\"строка\"): " + func1.equals("строка") + " (ожидается false)");

        System.out.println("\n   Тестирование hashCode():");
        System.out.println("   func1.hashCode(): " + func1.hashCode());
        System.out.println("   func2.hashCode(): " + func2.hashCode());
        System.out.println("   func3.hashCode(): " + func3.hashCode());

        System.out.println("   func1.hashCode() == func2.hashCode(): " +
                (func1.hashCode() == func2.hashCode()) + " (ожидается true для равных объектов)");

        System.out.println("\n   Тестирование clone():");
        ArrayTabulatedFunction func1Clone = (ArrayTabulatedFunction) func1.clone();
        System.out.println("   func1: " + func1);
        System.out.println("   clone: " + func1Clone);
        System.out.println("   func1.equals(clone): " + func1.equals(func1Clone));
        System.out.println("   func1 == clone: " + (func1 == func1Clone) + " (ожидается false)");

        func1Clone.setPointY( index: 0,  y: 100);
        System.out.println("\n   После изменения clone.points[0].y = 100:");
        System.out.println("   func1.points[0].y: " + func1.getPointY( index: 0) + " (ожидается 0)");
        System.out.println("   clone.points[0].y: " + func1Clone.getPointY( index: 0) + " (ожидается 100)");
        System.out.println("   func1.equals(clone): " + func1.equals(func1Clone) + " (ожидается false)");
    }

    public static void testTask5Part3() throws Exception {  1 usage  new *
        System.out.println("\n\n3. ТЕСТИРОВАНИЕ МЕТОДОВ LinkedListTabulatedFunction:");

        FunctionPoint[] points1 = {
                new FunctionPoint( x: 0,  y: 0),
                new FunctionPoint( x: 1,  y: 1),
                new FunctionPoint( x: 2,  y: 4)
        };

        LinkedListTabulatedFunction func1 = new LinkedListTabulatedFunction(points1);
        LinkedListTabulatedFunction func2 = new LinkedListTabulatedFunction(points1);

        ArrayTabulatedFunction arrayFunc = new ArrayTabulatedFunction(points1);

        System.out.println("   LinkedList func1: " + func1.toString());
        System.out.println("   LinkedList func2: " + func2.toString());
        System.out.println("   Array func: " + arrayFunc.toString());

        System.out.println("\n   Тестирование equals():");
        System.out.println("   func1.equals(func2): " + func1.equals(func2) + " (ожидается true)");
        System.out.println("   func1.equals(arrayFunc): " + func1.equals(arrayFunc) + " (ожидается true - сравнение через интерфейс)");
        System.out.println("   arrayFunc.equals(func1): " + arrayFunc.equals(func1) + " (ожидается true - сравнение через интерфейс)");

        System.out.println("\n   Тестирование hashCode():");
        System.out.println("   func1.hashCode(): " + func1.hashCode());
        System.out.println("   func2.hashCode(): " + func2.hashCode());
        System.out.println("   arrayFunc.hashCode(): " + arrayFunc.hashCode());

        System.out.println("   func1.hashCode() == func2.hashCode(): " +
                (func1.hashCode() == func2.hashCode()) + " (ожидается true для равных объектов)");

        System.out.println("\n   Тестирование clone():");
        LinkedListTabulatedFunction func1Clone = (LinkedListTabulatedFunction) func1.clone();
        System.out.println("   func1: " + func1);
        System.out.println("   clone: " + func1Clone);
        System.out.println("   func1.equals(clone): " + func1.equals(func1Clone));
        System.out.println("   func1 == clone: " + (func1 == func1Clone) + " (ожидается false)");

        func1Clone.setPointY( index: 1,  y: 100);
        System.out.println("\n   После изменения clone.points[1].y = 100:");
        System.out.println("   func1.points[1].y: " + func1.getPointY( index: 1) + " (ожидается 1)");
        System.out.println("   clone.points[1].y: " + func1Clone.getPointY( index: 1) + " (ожидается 100)");
        System.out.println("   func1.equals(clone): " + func1.equals(func1Clone) + " (ожидается false)");
    }

    public static void testTask5Part4() throws Exception {  1 usage  new *
```

```
        System.out.println("\n\n4. ДОПОЛНИТЕЛЬНЫЕ ТЕСТЫ И СРАВНЕНИЯ:");

        System.out.println("   Тест на согласованность equals() и hashCode():");

        FunctionPoint[][] testPoints = {
                {new FunctionPoint( x: 0,  y: 0), new FunctionPoint( x: 1,  y: 1)},
                {new FunctionPoint( x: 0,  y: 0), new FunctionPoint( x: 1,  y: 1.001)},
                {new FunctionPoint( x: 0,  y: 0), new FunctionPoint( x: 1,  y: 1)}
        };

        ArrayTabulatedFunction[] functions = new ArrayTabulatedFunction[3];
        for (int i = 0; i < 3; i++) {
            functions[i] = new ArrayTabulatedFunction(testPoints[i]);
        }

        System.out.println("   func0.hashCode(): " + functions[0].hashCode());
        System.out.println("   func1.hashCode(): " + functions[1].hashCode());
        System.out.println("   func2.hashCode(): " + functions[2].hashCode());

        System.out.println("   func0.equals(func1): " + functions[0].equals(functions[1]) +
                " (ожидается false из-за небольшого изменения)");
        System.out.println("   func0.equals(func2): " + functions[0].equals(functions[2]) +
                " (ожидается true)");
        System.out.println("   func0.hashCode() == func2.hashCode(): " +
                (functions[0].hashCode() == functions[2].hashCode()) + " (ожидается true)");

        System.out.println("\n   Тест на хэш-код с нулевой точкой:");

        ArrayTabulatedFunction funcWithZero = new ArrayTabulatedFunction(
                new FunctionPoint[]{new FunctionPoint( x: -1,  y: 1), new FunctionPoint( x: 0,  y: 0), new FunctionPoint( x: 1,  y: 1)}
        );

        ArrayTabulatedFunction funcWithoutZero = new ArrayTabulatedFunction(
                new FunctionPoint[]{new FunctionPoint( x: -1,  y: 1), new FunctionPoint( x: 1,  y: 1)}
        );

        System.out.println("   funcWithZero: " + funcWithZero);
        System.out.println("   funcWithoutZero: " + funcWithoutZero);
        System.out.println("   funcWithZero.hashCode(): " + funcWithZero.hashCode());
        System.out.println("   funcWithoutZero.hashCode(): " + funcWithoutZero.hashCode());
        System.out.println("   Хэши разные: " + (funcWithZero.hashCode() != funcWithoutZero.hashCode()));
    }
}
```

Вывод в консоль:

```
=== ЛАБОРАТОРНАЯ РАБОТА №5: ПЕРЕОПРЕДЕЛЕНИЕ МЕТОДОВ OBJECT ===

1. ТЕСТИРОВАНИЕ МЕТОДОВ FunctionPoint:
   Точка p1: (1,50; 2,50)
   Точка p2: (1,50; 2,50)
   Точка p3: (1,50; 2,60)
   Точка p4: (1,60; 2,50)

   Тестирование equals():
   p1.equals(p2): true (ожидается true)
   p1.equals(p3): false (ожидается false)
   p1.equals(p4): false (ожидается false)
   p1.equals(null): false (ожидается false)
   p1.equals("строка"): false (ожидается false)

   Тестирование hashCode():
   p1.hashCode(): 2147221504
   p2.hashCode(): 2147221504
   p3.hashCode(): -1288699903
   p4.hashCode(): -429654013
   p1.hashCode() == p2.hashCode(): true (ожидается true для равных объектов)

   Тестирование clone():
   p1: (1,50; 2,50)
   clone: (1,50; 2,50)
   p1.equals(clone): true
   p1 == clone: false (ожидается false)
   После изменения clone.x = 10.0:
   p1.x: 1.5 (ожидается 1.5)
   clone.x: 10.0 (ожидается 10.0)


2. ТЕСТИРОВАНИЕ МЕТОДОВ ArrayTabulatedFunction:
   Функция func1: {(0,00; 0,00), (1,00; 1,00), (2,00; 4,00), (3,00; 9,00)}
   Функция func2: {(0,00; 0,00), (1,00; 1,00), (2,00; 4,00), (3,00; 9,00)}
   Функция func3: {(0,00; 0,00), (1,00; 1,00), (2,00; 4,00)}
```

```
    Тестирование equals():
    func1.equals(func2): true (ожидается true)
    func1.equals(func3): false (ожидается false)
    func1.equals(null): false (ожидается false)
    func1.equals("строка"): false (ожидается false)

    Тестирование hashCode():
    func1.hashCode(): 3801092
    func2.hashCode(): 3801092
    func3.hashCode(): 1048579
    func1.hashCode() == func2.hashCode(): true (ожидается true для равных объектов)

    Тестирование clone():
    func1: {(0,00; 0,00), (1,00; 1,00), (2,00; 4,00), (3,00; 9,00)}
    clone: {(0,00; 0,00), (1,00; 1,00), (2,00; 4,00), (3,00; 9,00)}
    func1.equals(clone): true
    func1 == clone: false (ожидается false)

    После изменения clone.points[0].y = 100:
    func1.points[0].y: 0.0 (ожидается 0)
    clone.points[0].y: 100.0 (ожидается 100)
    func1.equals(clone): false (ожидается false)


3. ТЕСТИРОВАНИЕ МЕТОДОВ LinkedListTabulatedFunction:
    LinkedList func1: {(0,00; 0,00), (1,00; 1,00), (2,00; 4,00)
    LinkedList func2: {(0,00; 0,00), (1,00; 1,00), (2,00; 4,00)
    Array func: {(0,00; 0,00), (1,00; 1,00), (2,00; 4,00)}

    Тестирование equals():
    func1.equals(func2): true (ожидается true)
    func1.equals(arrayFunc): true (ожидается true - сравнение через интерфейс)
    arrayFunc.equals(func1): true (ожидается true - сравнение через интерфейс)

    Тестирование hashCode():
    func1.hashCode(): 1048579
    func2.hashCode(): 1048579
    arrayFunc.hashCode(): 1048579
```

```
   func1.hashCode() == func2.hashCode(): true (ожидается true для равных объектов)

   Тестирование clone():
   func1: {(0,00; 0,00), (1,00; 1,00), (2,00; 4,00)
   clone: {(0,00; 0,00), (1,00; 1,00), (2,00; 4,00)
   func1.equals(clone): true
   func1 == clone: false (ожидается false)

   После изменения clone.points[1].y = 100:
   func1.points[1].y: 1.0 (ожидается 1)
   clone.points[1].y: 100.0 (ожидается 100)
   func1.equals(clone): false (ожидается false)


4. ДОПОЛНИТЕЛЬНЫЕ ТЕСТЫ И СРАВНЕНИЯ:
   Тест на согласованность equals() и hashCode():
   func0.hashCode(): 2
   func1.hashCode(): -1821067152
   func2.hashCode(): 2
   func0.equals(func1): false (ожидается false из-за небольшого изменения)
   func0.equals(func2): true (ожидается true)
   func0.hashCode() == func2.hashCode(): true (ожидается true)

   Тест на хэш-код с нулевой точкой:
   funcWithZero: {(-1,00; 1,00), (0,00; 0,00), (1,00; 1,00)}
   funcWithoutZero: {(-1,00; 1,00), (1,00; 1,00)}
   funcWithZero.hashCode(): -2147483645
   funcWithoutZero.hashCode(): -2147483646
   Хэши разные: true

=== ВСЕ ТЕСТЫ ВЫПОЛНЕНЫ ===
```