

# Лабораторная работа №5

Выполнила: Сингатуллина Алина Марсовна

Группа 6204-010302D

## Оглавление

Задание 1 .....	3
Задание 2 .....	3
Задание 3.....	5
Задание 4.....	6
Задание 5.....	6

## Задание 1

В классе `FunctionPoint` переопределила следующие методы.

- `String toString()`
- `boolean equals(Object o)`
- `int hashCode()`
- `Object clone()`

*FunctionPoint.java*

```
@Override
public String toString() {
    return "(" + x + "; " + y + ")";
}

@Override
public boolean equals(Object obj) {
    if (this == obj) return true;
    if (obj == null || getClass() != obj.getClass()) return false;
    FunctionPoint that = (FunctionPoint) obj;
    return Double.compare(that.x, x) == 0 &&
        Double.compare(that.y, y) == 0;
}

@Override
public int hashCode() {
    // преобразуем координаты x и y в битовое представление
    long xBits = Double.doubleToLongBits(x);
    long yBits = Double.doubleToLongBits(y);

    // разбиваем каждое 64 битное число на две 32 битные части
    int xPart1 = (int) (xBits);           // младшие 32 бита x
    int xPart2 = (int) (xBits >>> 32);    // старшие 32 бита x

    int yPart1 = (int) (yBits);           // младшие 32 бита y
    int yPart2 = (int) (yBits >>> 32);    // старшие 32 бита y

    return xPart1 ^ xPart2 ^ yPart1 ^ yPart2;
}

@Override
public Object clone() {
    return new FunctionPoint(this);
}
```

## Задание 2

В классе `ArrayTabulatedFunction` переопределила следующие методы:

- `String toString()`
- `boolean equals()`
- `int hashCode()`
- `Object clone()`

*ArrayTabulatedFunction.java*

```

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

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof TabulatedFunction)) return false;

    TabulatedFunction other = (TabulatedFunction) o;

    if (this.getPointsCount() != other.getPointsCount()) return false;

    // оптимизация для ArrayTabulatedFunction
    if (o instanceof ArrayTabulatedFunction) {
        ArrayTabulatedFunction arrayOther = (ArrayTabulatedFunction) o;
        for (int i = 0; i < pointsCount; i++) {
            if (!this.points[i].equals(arrayOther.points[i])) {
                return false;
            }
        }
    } else {
        // общий случай для любого TabulatedFunction
        for (int i = 0; i < pointsCount; i++) {
            FunctionPoint myPoint = this.getPoint(i);
            FunctionPoint otherPoint = other.getPoint(i);
            if (!myPoint.equals(otherPoint)) {
                return false;
            }
        }
    }
    return true;
}

@Override
public int hashCode() {
    int result = pointsCount;
    for (int i = 0; i < pointsCount; i++) {
        result ^= points[i].hashCode();
    }
    return result;
}

@Override
public Object clone() {
    // создаем копию через конструктор
    FunctionPoint[] pointsCopy = new FunctionPoint[this.pointsCount];
    for (int i = 0; i < this.pointsCount; i++) {
        pointsCopy[i] = new FunctionPoint(this.points[i]);
    }
    return new ArrayTabulatedFunction(pointsCopy);
}

```

### Задание 3

Аналогично, переопределила методы toString(), equals(), hashCode() и clone() в классе LinkedListTabulatedFunction.

*LinkedListTabulatedFunction.java*

```
@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("{");
    FunctionNode current = head.next;
    while (current != head) {
        sb.append("(").append(current.point.getX()).append(";");
        sb.append(current.point.getY()).append(")");
        if (current.next != head) {
            sb.append(", ");
        }
        current = current.next;
    }
    sb.append("}");
    return sb.toString();
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof TabulatedFunction)) return false;

    TabulatedFunction other = (TabulatedFunction) o;

    if (this.getPointsCount() != other.getPointsCount()) return false;

    // Оптимизация для LinkedListTabulatedFunction
    if (o instanceof LinkedListTabulatedFunction) {
        LinkedListTabulatedFunction listOther = (LinkedListTabulatedFunction)
o;

        FunctionNode currentThis = this.head.next;
        FunctionNode currentOther = listOther.head.next;

        while (currentThis != head && currentOther != listOther.head) {
            if (!currentThis.point.equals(currentOther.point)) {
                return false;
            }
            currentThis = currentThis.next;
            currentOther = currentOther.next;
        }
    } else {
        // Общий случай для любого TabulatedFunction
        for (int i = 0; i < pointsCount; i++) {
            FunctionPoint myPoint = this.getPoint(i);
            FunctionPoint otherPoint = other.getPoint(i);
            if (!myPoint.equals(otherPoint)) {
                return false;
            }
        }
    }
    return true;
}

@Override
public int hashCode() {
    int result = pointsCount;
```

```

        FunctionNode current = head.next;
        while (current != head) {
            result ^= current.point.hashCode();
            current = current.next;
        }
        return result;
    }
}

@Override
public Object clone() {
    // создаем новый список из копий точек
    FunctionPoint[] pointsCopy = new FunctionPoint[pointsCount];
    FunctionNode current = head.next;

    for (int i = 0; i < pointsCount; i++) {
        pointsCopy[i] = new FunctionPoint(current.point); // используем
        копирующий конструктор
        current = current.next;
    }

    return new LinkedListTabulatedFunction(pointsCopy);
}

```

#### Задание 4

Сделала так, чтобы все объекты типа TabulatedFunction были клонируемыми с точки зрения JVM и внесла метод clone() в этот интерфейс.

*TabulatedFunction.java*

```

package functions;

public interface TabulatedFunction extends Function, Cloneable {
    int getPointsCount();
    FunctionPoint getPoint(int index);
    void setPoint(int index, FunctionPoint point) throws
    InappropriateFunctionPointException;
    double getPointX(int index);
    void setPointX(int index, double x) throws
    InappropriateFunctionPointException;
    double getPointY(int index);
    void setPointY(int index, double y);
    void deletePoint(int index);
    void addPoint(FunctionPoint point) throws
    InappropriateFunctionPointException;
    Object clone();
}

```

#### Задание 5

Создала класс Main и проверила работу написанных методов.

*Main.java*

Тестирование методов toString(), equals(), hashCode(), clone()

1. Тестирование toString():

ArrayTabulatedFunction: {(0.0; 1.0), (1.0; 3.0), (2.0; 5.0)}

LinkedListTabulatedFunction: {(0.0; 1.0), (1.0; 3.0), (2.0; 5.0)}

2. Тестирование equals():  
arrayFunc1.equals(arrayFunc2): true  
listFunc1.equals(listFunc2): true  
arrayFunc1.equals(arrayFunc3): false  
listFunc1.equals(listFunc3): false  
arrayFunc1.equals(listFunc1): true  
listFunc1.equals(arrayFunc1): true

3. Тестирование hashCode():  
arrayFunc1.hashCode(): 1075576835  
arrayFunc2.hashCode(): 1075576835  
arrayFunc3.hashCode(): 1075314691  
listFunc1.hashCode(): 1075576835  
listFunc2.hashCode(): 1075576835  
listFunc3.hashCode(): 1075314691

Проверка согласованности equals() и hashCode():  
arrayFunc1.equals(arrayFunc2) && arrayFunc1.hashCode() == arrayFunc2.hashCode(): true

Изменение объекта и хэш-кода:  
До изменения - arrayFunc1.hashCode(): 1075576835  
После изменения - arrayFunc1.hashCode(): 161897530  
arrayFunc1.equals(arrayFunc2) после изменения: false

4. Тестирование clone() и глубокого клонирования:  
arrayFunc2.equals(arrayClone): true  
arrayFunc2 == arrayClone: false

Проверка глубокого клонирования (ArrayTabulatedFunction):  
До изменения - arrayClone.getPointY(1): 3.0  
После изменения оригинала - arrayClone.getPointY(1): 3.0  
Клон не изменился: true

listFunc2.equals(listClone): true  
listFunc2 == listClone: false

Проверка глубокого клонирования (LinkedListTabulatedFunction):  
До изменения - listClone.getPointY(1): 3.0  
После изменения оригинала - listClone.getPointY(1): 3.0  
Клон не изменился: true