МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

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

ОТЧЕТ ПО
ЛАБОРАТОРНОЙ РАБОТЕ № 7

«Работа с паттернами и использование рефлексии JAVA»

по курсу
Объектно-ориентированное программирование

Выполнила: Егорова-Екимкова Яна,
студент группы 6203-010302D

Оглавление

Задание №1

Для выпаолнения задания 1 мне необходимо было расширить класс TabulatedFunction параметризованным типом (Iterator<FunctionPoint>), для этого я переопределила метод iterator() в классах ArrayTabulatedFunction и LinkedListTabulatedFunction Методы внутри iterator() анонимные и не вызывают публичные методы табулированной функции, выбрасывают исключения UnsupportedOperationException(remove()) и NoSuchElementException(next()). Далее я проверила работу кода.

```
public class LinkedListTabulatedFunction implements TabulatedFunction {  7 usages  new *
    @Override  new *
    public Iterator<FunctionPoint> iterator() {
        return new Iterator<FunctionPoint>() {  new *
            private FunctionNode currentNode = head.next;  4 usages
            private int currentIndex = 0;  2 usages

            @Override  new *
            public boolean hasNext() {
                return currentNode != head && currentIndex < pointsCount;
            }

            @Override  new *
            public FunctionPoint next() {
                if (!hasNext()) {
                    throw new NoSuchElementException("No more points in tabulated function");
                }
                FunctionPoint point = new FunctionPoint(currentNode.point);
                currentNode = currentNode.next;
                currentIndex++;
                return point;
            }

            @Override  new *
            public void remove() {
                throw new UnsupportedOperationException("Remove operation is not supported");
            }
        };
    }
```

Рисунок 1

```java
public class ArrayTabulatedFunction implements TabulatedFunction {  11 usages  new *
    public Iterator<FunctionPoint> iterator() {
        return new Iterator<FunctionPoint>() {  new *
            private int currentIndex = 0;  2 usages

            @Override  new *
            public boolean hasNext() {
                return currentIndex < pointsCount;
            }

            @Override  new *
            public FunctionPoint next() {
                if (!hasNext()) {
                    throw new NoSuchElementException("No more points in tabulated function");
                }
                return new FunctionPoint(array[currentIndex++]);
            }

            @Override  new *
            public void remove() {
                throw new UnsupportedOperationException("Remove operation is not supported");
            }
        };
    }
```

Рисунок 2

```java
public class Main {  new *
    public static void main(String[] args){  new *
        System.out.println("Task 1");

        TabulatedFunction f = new ArrayTabulatedFunction( leftX: 1,  rightX: 100,  pointsCount: 10);
        for(FunctionPoint p : f) { System.out.println(p); }
    }
}
```

```
Main ×

C:\Users\sweet\.jdks\openjdk-24.0.2+12-54\bin\java.exe "-javaagent:C:\Program Files\JetBrains\Intel
Task 1
(1.0; 0.0)
(12.0; 0.0)
(23.0; 0.0)
(34.0; 0.0)
(45.0; 0.0)
(56.0; 0.0)
(67.0; 0.0)
(78.0; 0.0)
(89.0; 0.0)
(100.0; 0.0)

Process finished with exit code 0
```

Рисунок 3

```
public class Main {  new *
    public static void main(String[] args){  new *
        System.out.println("Task 1");
        /*TabulatedFunction f = new ArrayTabulatedFunction(1, 100, 10);
        for(FunctionPoint p : f) { System.out.println(p); }*/
        TabulatedFunction f1 = new LinkedListTabulatedFunction( leftX: 1.0,  rightX: 100.0,  pointsCount: 10);
        for (FunctionPoint p : f1) { System.out.println(f1); }
```

```
Main ×

C:\Users\sweet\.jdks\openjdk-24.0.2+12-54\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.2\
Task 1
{(1,0; 0,0), (12,0; 0,0), (23,0; 0,0), (34,0; 0,0), (45,0; 0,0), (56,0; 0,0), (67,0; 0,0), (78,0; 0,0), (89,0; 0,0), (100,0; 0,0)}
{(1,0; 0,0), (12,0; 0,0), (23,0; 0,0), (34,0; 0,0), (45,0; 0,0), (56,0; 0,0), (67,0; 0,0), (78,0; 0,0), (89,0; 0,0), (100,0; 0,0)}
{(1,0; 0,0), (12,0; 0,0), (23,0; 0,0), (34,0; 0,0), (45,0; 0,0), (56,0; 0,0), (67,0; 0,0), (78,0; 0,0), (89,0; 0,0), (100,0; 0,0)}
{(1,0; 0,0), (12,0; 0,0), (23,0; 0,0), (34,0; 0,0), (45,0; 0,0), (56,0; 0,0), (67,0; 0,0), (78,0; 0,0), (89,0; 0,0), (100,0; 0,0)}
{(1,0; 0,0), (12,0; 0,0), (23,0; 0,0), (34,0; 0,0), (45,0; 0,0), (56,0; 0,0), (67,0; 0,0), (78,0; 0,0), (89,0; 0,0), (100,0; 0,0)}
{(1,0; 0,0), (12,0; 0,0), (23,0; 0,0), (34,0; 0,0), (45,0; 0,0), (56,0; 0,0), (67,0; 0,0), (78,0; 0,0), (89,0; 0,0), (100,0; 0,0)}
{(1,0; 0,0), (12,0; 0,0), (23,0; 0,0), (34,0; 0,0), (45,0; 0,0), (56,0; 0,0), (67,0; 0,0), (78,0; 0,0), (89,0; 0,0), (100,0; 0,0)}
{(1,0; 0,0), (12,0; 0,0), (23,0; 0,0), (34,0; 0,0), (45,0; 0,0), (56,0; 0,0), (67,0; 0,0), (78,0; 0,0), (89,0; 0,0), (100,0; 0,0)}
{(1,0; 0,0), (12,0; 0,0), (23,0; 0,0), (34,0; 0,0), (45,0; 0,0), (56,0; 0,0), (67,0; 0,0), (78,0; 0,0), (89,0; 0,0), (100,0; 0,0)}
{(1,0; 0,0), (12,0; 0,0), (23,0; 0,0), (34,0; 0,0), (45,0; 0,0), (56,0; 0,0), (67,0; 0,0), (78,0; 0,0), (89,0; 0,0), (100,0; 0,0)}

Process finished with exit code 0
```

Рисунок 4

Задание №2

Для задания 2 я описала базовый интерфейс TabulatedFunctionFactory, где перегружен метод createTabulatedFunction().

```
package functions;

public interface TabulatedFunctionFactory {  2 usages  2 implementations  new *
    TabulatedFunction createTabulatedFunction(double leftX, double rightX, int pointsCount) throws IllegalArgumentException;
    TabulatedFunction createTabulatedFunction(double leftX, double rightX, double[] values) throws IllegalArgumentException;
    TabulatedFunction createTabulatedFunction(FunctionPoint[] points) throws IllegalArgumentException;  no usages  2 implementation
}
```

Рисунок 5

После, я описала в классах ArrayTabulatedFunction и LinkedListTabulatedFunction классы фабрик ArrayTabulatedFunctionFactory и LinkedListTabulatedFunctionFactory, реализующие интерфейс фабрики и порождающие объекты соответствующих классов табулированных функций, сделала фабричные классы вложенными и публичными.

```
public static class ArrayTabulatedFunctionFactory implements TabulatedFunctionFactory {  no usages  new *
    @Override  no usages  new *
    public TabulatedFunction createTabulatedFunction(double leftX, double rightX, int pointsCount) throws IllegalArgumentException {
        return new ArrayTabulatedFunction(leftX, rightX, pointsCount);
    }

    @Override  no usages  new *
    public TabulatedFunction createTabulatedFunction(double leftX, double rightX, double[] values) throws IllegalArgumentException {
        return new ArrayTabulatedFunction(leftX, rightX, values);
    }

    @Override  no usages  new *
    public TabulatedFunction createTabulatedFunction(FunctionPoint[] points) throws IllegalArgumentException {
        return new ArrayTabulatedFunction(points);
    }
}
```

Рисунок 6

```
public static class LinkedListTabulatedFunctionFactory implements TabulatedFunctionFactory {  1 usage  new *
    @Override  1 usage  new *
    public TabulatedFunction createTabulatedFunction(double leftX, double rightX, int pointsCount) throws IllegalArgumentException{
        return new LinkedListTabulatedFunction(leftX, rightX, pointsCount);
    }

    @Override  1 usage  new *
    public TabulatedFunction createTabulatedFunction(double leftX, double rightX, double[] values) throws IllegalArgumentException {
        return new LinkedListTabulatedFunction(leftX, rightX, values);
    }

    @Override  1 usage  new *
    public TabulatedFunction createTabulatedFunction(FunctionPoint[] points) throws IllegalArgumentException {
        return new LinkedListTabulatedFunction(points);
    }
}
```

Рисунок 7

В классе TabulatedFunctions объявила приватное статическое поле типа TabulatedFunctionFactory и проинициализировала его объектом одного из описанных классов фабрик (я выбрала ArrayTabulatedFunction), объявила метод setTabulatedFunctionFactory(), заменяющий объект фабрики, перегрузила три метода createTabulatedFunction(), возвращающих объекты табулированных функций, созданные с помощью текущей фабрики

```
public class TabulatedFunctions {  6 usages  new *

    private TabulatedFunctions(){};  no usages  new *
    // класс обновлен с исп. фабричного метода

    // проинициализировала через array
    private static TabulatedFunctionFactory factory = new ArrayTabulatedFunction.ArrayTabulatedFunctionFactory();  4 usages
    // или можно через list
    /*private static TabulatedFunctionFactory factory = new LinkedListTabulatedFunction.LinkedListTabulatedFunctionFactory();*/

    //сетфектори
    public static void setTabulatedFunctionFactory(TabulatedFunctionFactory factory) {TabulatedFunctions.factory = factory;}  2 usages  new *

    // перегруженные методы создания таб.функ. с исп. фабрики
    public static TabulatedFunction createTabulatedFunction(double leftX, double rightX, int pointsCount) {  no usages  new *
        return factory.createTabulatedFunction(leftX, rightX, pointsCount);
    }

    public static TabulatedFunction createTabulatedFunction(double leftX, double rightX, double[] values) {  no usages  new *
        return factory.createTabulatedFunction(leftX, rightX, values);
    }

    public static TabulatedFunction createTabulatedFunction(FunctionPoint[] points) {  3 usages  new *
        return factory.createTabulatedFunction(points);
    }
```

Рисунок 8

В методах, где создается объект табулированной функции, сделала создание через метод createTabulatedFunction().

```java
// оригинальный табулейт теперь создает объект через create
public static TabulatedFunction tabulate(Function func, double leftX, double rightX, int pointsCount){
    if (func == null) throw new IllegalArgumentException("Function must exist");
    if (leftX > rightX) throw new IllegalArgumentException("Left border must be less than right border");
    if (pointsCount < 2) throw new IllegalArgumentException("Points count must be more than 2");

    FunctionPoint point[] = new FunctionPoint[pointsCount];
    double step = (rightX - leftX) / (pointsCount - 1);
    for (int i = 0; i<pointsCount; i++){
        double x = leftX + i*step;
        double y = func.getFunctionValue(x);
        point[i] = new FunctionPoint(x,y);
    }
    return createTabulatedFunction(point);
}
```

Рисунок 9

```java
// также теперь создает объект таб.функ. через create
public static TabulatedFunction inputTabulatedFunction(InputStream in) throws IOException{  no usages  new *
    DataInputStream cin = new DataInputStream(in);
    int pointsCount = cin.readInt();
    FunctionPoint points[] = new FunctionPoint[pointsCount];
    for (int i =0; i<pointsCount; i++){
        double x = cin.readDouble();
        double y = cin.readDouble();
        points[i] = new FunctionPoint(x,y);
    }
    return createTabulatedFunction(points);
}

public static void writeTabulatedFunction(TabulatedFunction function, Writer out) throws IOException {...}

// теперь создает объект таб.функ через create
public static TabulatedFunction readTabulatedFunction(Reader in) throws IOException {  no usages  new *
    StreamTokenizer tokenizer = new StreamTokenizer(in);
    tokenizer.nextToken();
    int pointsCount = (int) tokenizer.nval;

    FunctionPoint[] points = new FunctionPoint[pointsCount];

    for (int i = 0; i < pointsCount; i++) {
        tokenizer.nextToken();
        double x = tokenizer.nval;
        tokenizer.nextToken();
        double y = tokenizer.nval;
        points[i] = new FunctionPoint(x, y);
    }
    return createTabulatedFunction(points);
}
```

Рисунок 10

Проверила работу кода в main().

```
        System.out.println("Task 2");
        Function f2 = new Cos();
        TabulatedFunction tf;
        tf = TabulatedFunctions.tabulate(f2, leftX: 0, Math.PI, pointsCount: 11);
        System.out.println(tf.getClass());
        TabulatedFunctions.setTabulatedFunctionFactory(new
                LinkedListTabulatedFunction.LinkedListTabulatedFunctionFactory());
        tf = TabulatedFunctions.tabulate(f2, leftX: 0, Math.PI, pointsCount: 11);
        System.out.println(tf.getClass());
        TabulatedFunctions.setTabulatedFunctionFactory(new
                ArrayTabulatedFunction.ArrayTabulatedFunctionFactory());
        tf = TabulatedFunctions.tabulate(f2, leftX: 0, Math.PI, pointsCount: 11);
        System.out.println(tf.getClass());
```

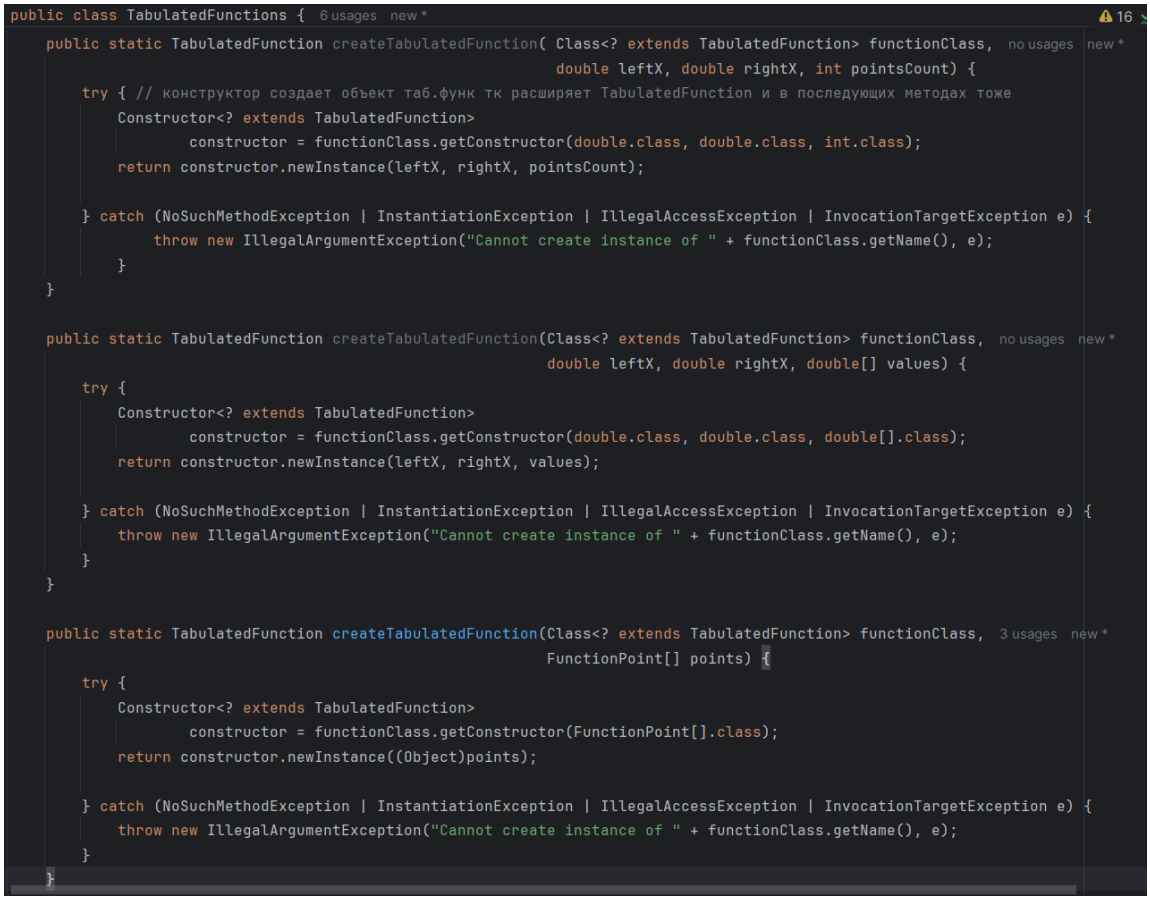
Main ×

```
C:\Users\sweet\.jdks\openjdk-24.0.2+12-54\bin\java.exe "-javaagent:C:\Program Files\JetBrai
Task 2
class functions.ArrayTabulatedFunction
class functions.LinkedListTabulatedFunction
class functions.ArrayTabulatedFunction

Process finished with exit code 0
```

Рисунок 11

Задание №3

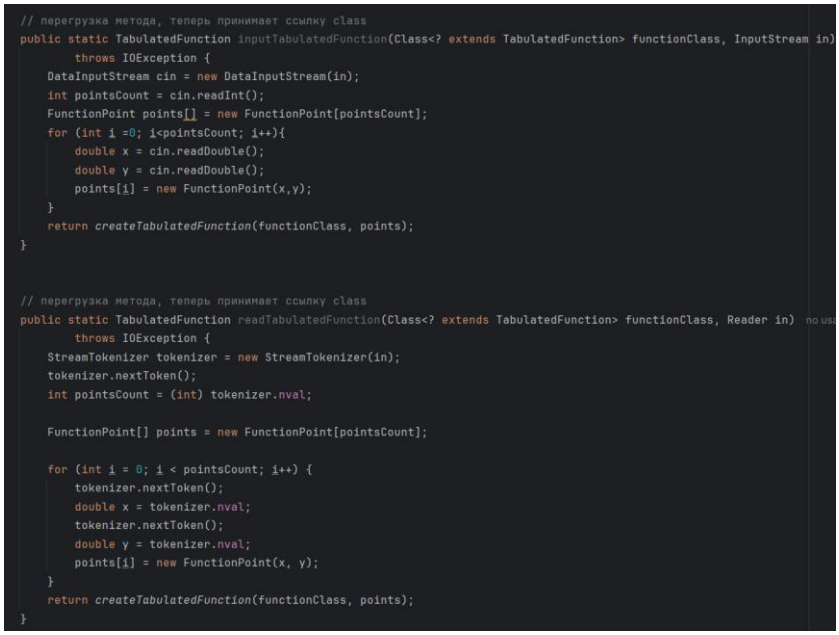
По заданию 3 мне необходимо было добавить ещё три перегруженных метода createTabulatedFunction(), где параметры должны повторять параметры трёх аналогичных фабричных методов, но ещё эти методы должны получать ссылку типа Class на описание класса, объект которого требуется создать, а также в эти методы можно передать только ссылки на классы, реализующие интерфейс TabulatedFunction. Также есть проверка исключений.



```
public class TabulatedFunctions {  6 usages  new *                                                    16
    public static TabulatedFunction createTabulatedFunction( Class<? extends TabulatedFunction> functionClass,  no usages  new *
                                                              double leftX, double rightX, int pointsCount) {
        try { // конструктор создает объект таб.функ тк расширяет TabulatedFunction и в последующих методах тоже
            Constructor<? extends TabulatedFunction>
                    constructor = functionClass.getConstructor(double.class, double.class, int.class);
            return constructor.newInstance(leftX, rightX, pointsCount);

        } catch (NoSuchMethodException | InstantiationException | IllegalAccessException | InvocationTargetException e) {
                throw new IllegalArgumentException("Cannot create instance of " + functionClass.getName(), e);
            }
    }

    public static TabulatedFunction createTabulatedFunction(Class<? extends TabulatedFunction> functionClass,  no usages  new *
                                                             double leftX, double rightX, double[] values) {
        try {
            Constructor<? extends TabulatedFunction>
                    constructor = functionClass.getConstructor(double.class, double.class, double[].class);
            return constructor.newInstance(leftX, rightX, values);

        } catch (NoSuchMethodException | InstantiationException | IllegalAccessException | InvocationTargetException e) {
            throw new IllegalArgumentException("Cannot create instance of " + functionClass.getName(), e);
        }
    }

    public static TabulatedFunction createTabulatedFunction(Class<? extends TabulatedFunction> functionClass,  3 usages  new *
                                                             FunctionPoint[] points) {
        try {
            Constructor<? extends TabulatedFunction>
                    constructor = functionClass.getConstructor(FunctionPoint[].class);
            return constructor.newInstance((Object)points);

        } catch (NoSuchMethodException | InstantiationException | IllegalAccessException | InvocationTargetException e) {
            throw new IllegalArgumentException("Cannot create instance of " + functionClass.getName(), e);
        }
    }
```

Рисунок 12

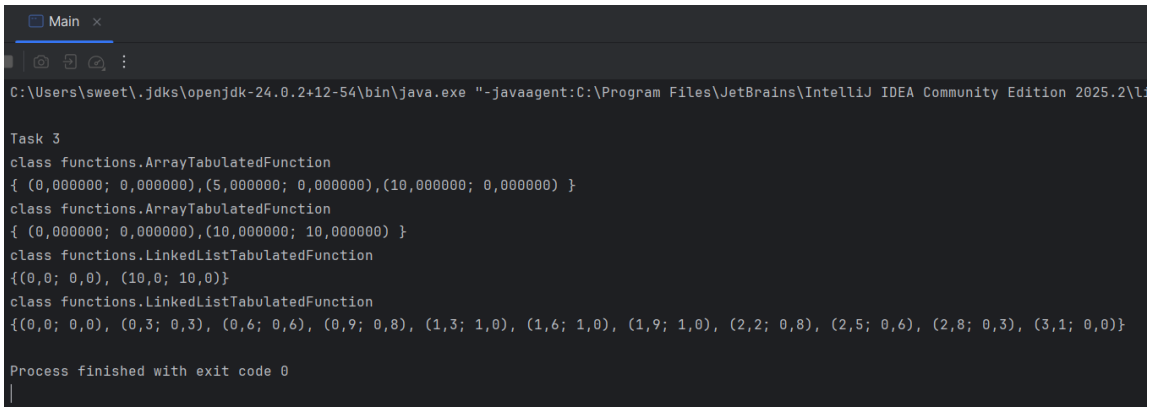
Перегрузила методы readTabulatedFunction() и inputTabulatedFunction().



```java
// перегрузка метода, теперь принимает ссылку class
public static TabulatedFunction inputTabulatedFunction(Class<? extends TabulatedFunction> functionClass, InputStream in)
        throws IOException {
    DataInputStream cin = new DataInputStream(in);
    int pointsCount = cin.readInt();
    FunctionPoint points[] = new FunctionPoint[pointsCount];
    for (int i =0; i<pointsCount; i++){
        double x = cin.readDouble();
        double y = cin.readDouble();
        points[i] = new FunctionPoint(x,y);
    }
    return createTabulatedFunction(functionClass, points);
}


// перегрузка метода, теперь принимает ссылку class
public static TabulatedFunction readTabulatedFunction(Class<? extends TabulatedFunction> functionClass, Reader in)
        throws IOException {
    StreamTokenizer tokenizer = new StreamTokenizer(in);
    tokenizer.nextToken();
    int pointsCount = (int) tokenizer.nval;

    FunctionPoint[] points = new FunctionPoint[pointsCount];

    for (int i = 0; i < pointsCount; i++) {
        tokenizer.nextToken();
        double x = tokenizer.nval;
        tokenizer.nextToken();
        double y = tokenizer.nval;
        points[i] = new FunctionPoint(x, y);
    }
    return createTabulatedFunction(functionClass, points);
}
```

Рисунок 13

Проверила работу кода в main().



```
Main
C:\Users\sweet\.jdks\openjdk-24.0.2+12-54\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.2\li

Task 3
class functions.ArrayTabulatedFunction
{ (0,000000; 0,000000),(5,000000; 0,000000),(10,000000; 0,000000) }
class functions.ArrayTabulatedFunction
{ (0,000000; 0,000000),(10,000000; 10,000000) }
class functions.LinkedListTabulatedFunction
{(0,0; 0,0), (10,0; 10,0)}
class functions.LinkedListTabulatedFunction
{(0,0; 0,0), (0,3; 0,3), (0,6; 0,6), (0,9; 0,8), (1,3; 1,0), (1,6; 1,0), (1,9; 1,0), (2,2; 0,8), (2,5; 0,6), (2,8; 0,3), (3,1; 0,0)}

Process finished with exit code 0
```

Рисунок 14