

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО  
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САМАРСКИЙ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени  
академика С.П. КОРОЛЁВА»

КАФЕДРА «ТЕХНИЧЕСКАЯ КИБЕРНЕТИКА»

ОТЧЕТ  
ПО ЛАБОРАТОРНОЙ РАБОТЕ

«Объектно-ориентированное программирование»

ЛАБОРАТОРНАЯ РАБОТА №7

Студент Богданчиков М.А.

Группа 6301-030301D

Руководитель Борисов Д. С.

Оценка \_\_\_\_\_

**Цель лабораторной работы:** Внести изменения в существующий набор типов табулированных функций, позволяющие обрабатывать точки функций по порядку (паттерн «Итератор»), а также выбирать тип объекта табулированной функции при его неявном создании (паттерн «Фабричный метод» и средства рефлексии).

### Ход выполнения лабораторной работы

**Задание 1:** Сделано так, что все объекты типа TabulatedFunction можно использовать в качестве объекта-агрегата в «улучшенном цикле for» (вариант for-each), извлекаемые объекты при этом имеют тип FunctionPoint.

В интерфейсе TabulatedFunction добавлен необходимый родительский тип, используйте при этом параметризованный тип (generic type).

В классах, реализующих интерфейс TabulatedFunction, добавлен требующийся метод, возвращающий объект итератора.

Классы итераторов сделаны анонимными.

В методе main() программы проверена работа итераторов классов табулированных функций.

```
public interface TabulatedFunction extends Function, Cloneable, Serializable, Iterable<FunctionPoint>
    int getPointsCount();
```

### Код 1.1

```
@Override
public Iterator<FunctionPoint> iterator() {
    return new Iterator<FunctionPoint>() {
        private int currentIndex = 0;

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

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

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

### Код 1.2 (Добавлено в ArrayTabulatedFunction)

```

@Override
public Iterator<FunctionPoint> iterator() {
    return new Iterator<FunctionPoint>() {
        private FunctionNode currentNode = head.next;
        private int currentIndex = 0;

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

        @Override
        public FunctionPoint next() {
            if (!hasNext()) {
                throw new NoSuchElementException("No more elements");
            }
            FunctionPoint result = new FunctionPoint(currentNode.point);
            currentNode = currentNode.next;
            currentIndex++;
            return result;
        }

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

```

### **Код 1.3 (Добавлено в LinkedListTabulatedFunction)**

```

6  public class Main {
7      public static void main(String[] args) {
8          TabulatedFunction func = new ArrayTabulatedFunction(0, 10, 5);
9          for (FunctionPoint p : func) {
10              System.out.println(p);
11          }
12      }

```

### **Код 1.4 (Добавлено в Main)**

```

PS C:\Javalabs\lab7> javac Main.java functions/*.java
PS C:\Javalabs\lab7> java Main
(0.0; 0.0)
(2.5; 0.0)
(5.0; 0.0)
(7.5; 0.0)
(10.0; 0.0)

```

## **Реализация**

**Задание 2:** В пакете functions описан базовый интерфейс фабрик табулированных функций TabulatedFunctionFactory. Интерфейс объявляет три перегруженных метода TabulatedFunction createTabulatedFunction(), параметры которых соответствуют параметрам конструкторов классов табулированных функций.

Описаны в

классах ArrayTabulatedFunction и LinkedListTabulatedFunction классы фабрик ArrayTabulatedFunctionFactory и LinkedListTabulatedFunctionFactory, реализующие интерфейс фабрики и порождающие объекты соответствующих классов табулированных функций.

В классе TabulatedFunctions объявлено приватное статическое поле типа TabulatedFunctionFactory и проинициализировано объектом одного из описанных классов фабрик. Также объявлен метод setTabulatedFunctionFactory(), позволяющий заменить объект фабрики.

В классе TabulatedFunctions описаны три перегруженных метода TabulatedFunction createTabulatedFunction(), возвращающих объекты табулированных функций, созданные с помощью текущей фабрики.

В остальных методах класса, где требуется создание объектов табулированных функций, замените явное создание объектов с помощью конструкторов на вызов соответствующего метода createTabulatedFunction().

```
1 package functions;
2
3 public interface TabulatedFunctionFactory {
4     TabulatedFunction createTabulatedFunction(double leftX, double rightX, int pointsCount);
5     TabulatedFunction createTabulatedFunction(double leftX, double rightX, double[] values);
6     TabulatedFunction createTabulatedFunction(FunctionPoint[] points);
7 }
```

## Код 2.1

```
public static class ArrayTabulatedFunctionFactory implements TabulatedFunctionFactory {  
    @Override  
    public TabulatedFunction createTabulatedFunction(double leftX, double rightX, int pointsCount) {  
        return new ArrayTabulatedFunction(leftX, rightX, pointsCount);  
    }  
  
    @Override  
    public TabulatedFunction createTabulatedFunction(double leftX, double rightX, double[] values) {  
        return new ArrayTabulatedFunction(leftX, rightX, values);  
    }  
  
    @Override  
    public TabulatedFunction createTabulatedFunction(FunctionPoint[] points) {  
        return new ArrayTabulatedFunction(points);  
    }  
}
```

## Код 2.2

```
public static class LinkedListTabulatedFunctionFactory implements TabulatedFunctionFactory {  
    @Override  
    public TabulatedFunction createTabulatedFunction(double leftX, double rightX, int pointsCount) {  
        return new LinkedListTabulatedFunction(leftX, rightX, pointsCount);  
    }  
  
    @Override  
    public TabulatedFunction createTabulatedFunction(double leftX, double rightX, double[] values) {  
        return new LinkedListTabulatedFunction(leftX, rightX, values);  
    }  
  
    @Override  
    public TabulatedFunction createTabulatedFunction(FunctionPoint[] points) {  
        return new LinkedListTabulatedFunction(points);  
    }  
}
```

## Код 2.3

```

9  private static TabulatedFunctionFactory factory =
10    new ArrayTabulatedFunction.ArrayTabulatedFunctionFactory();
11  public static void setTabulatedFunctionFactory(TabulatedFunctionFactory factory) {
12    TabulatedFunctions.factory = factory;
13  }
14  public static TabulatedFunction createTabulatedFunction(double leftX, double rightX, int pointsCount) {
15    return factory.createTabulatedFunction(leftX, rightX, pointsCount);
16  }
17  public static TabulatedFunction createTabulatedFunction(double leftX, double rightX, double[] values) {
18    return factory.createTabulatedFunction(leftX, rightX, values);
19  }
20  public static TabulatedFunction createTabulatedFunction(FunctionPoint[] points) {
21    return factory.createTabulatedFunction(points);
22  }
23  public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount) {
24    if (pointsCount < 2) {
25      throw new IllegalArgumentException("Points count must be at least 2");
26    }
27    if (leftX < function.getLeftDomainBorder() - 1e-10 ||
28        rightX > function.getRightDomainBorder() + 1e-10) {
29      throw new IllegalArgumentException("Tabulation interval is outside function domain");
30    }
31    FunctionPoint[] points = new FunctionPoint[pointsCount];
32    double step = (rightX - leftX) / (pointsCount - 1);
33    for (int i = 0; i < pointsCount; i++) {
34      double x = leftX + i * step;
35      double y = function.getFunctionValue(x);
36      points[i] = new FunctionPoint(x, y);
37    }
38    return createTabulatedFunction(points); // вместо new ArrayTabulatedFunction(points)
39  }

```

## Код 2.4

```

public class Main {
    public static void main(String[] args) {
        TabulatedFunction func = new ArrayTabulatedFunction(0, 10, 5);
        for (FunctionPoint p : func) {
            System.out.println(p);
        }
        // 2 задание
        Function f = new Cos();
        TabulatedFunction tf;
        tf = TabulatedFunctions.tabulate(f, 0, Math.PI, 11);
        System.out.println(tf.getClass());

        TabulatedFunctions.setTabulatedFunctionFactory(
            new LinkedListTabulatedFunction.LinkedListTabulatedFunctionFactory());

        tf = TabulatedFunctions.tabulate(f, 0, Math.PI, 11);
        System.out.println(tf.getClass());
    }
}

```

## Код 2.5

```
PS C:\Javalabs\lab7> javac Main.java functions/*.java
PS C:\Javalabs\lab7> java Main
(0.0; 0.0)
(2.5; 0.0)
(5.0; 0.0)
(7.5; 0.0)
(10.0; 0.0)
class functions.ArrayTabulatedFunction
class functions.LinkedListTabulatedFunction
PS C:\Javalabs\lab7>
```

## Реализация 2

**Задание 3:** В классе TabulatedFunctions добавлены ещё три перегруженных версии метода createTabulatedFunction().

В классе TabulatedFunctions перегружены методы, создающие объекты табулированных функций, добавив версии, принимающие также ссылку типа Class на описание класса, объект которого требуется создать. В эти методы можно передать только ссылки на классы, реализующие интерфейс TabulatedFunction.

Проверена в методе main() работа методов рефлексивного создания объектов, а также методов класса TabulatedFunctions, использующих создание объектов.

```

29 |     public static TabulatedFunction createTabulatedFunction(Class<? extends TabulatedFunction> functionClass,
30 |                                         double leftX, double rightX, int pointsCount) {
31 |         try {
32 |             Constructor<? extends TabulatedFunction> constructor =
33 |                 functionClass.getConstructor(double.class, double.class, int.class);
34 |             return constructor.newInstance(leftX, rightX, pointsCount);
35 |         } catch (NoSuchMethodException | InstantiationException |
36 |                  IllegalAccessException | InvocationTargetException e) {
37 |             throw new IllegalArgumentException("Cannot create tabulated function", e);
38 |         }
39 |     }
40 |
41 |     public static TabulatedFunction createTabulatedFunction(Class<? extends TabulatedFunction> functionClass,
42 |                                         double leftX, double rightX, double[] values) {
43 |         try {
44 |             Constructor<? extends TabulatedFunction> constructor =
45 |                 functionClass.getConstructor(double.class, double.class, double[].class);
46 |             return constructor.newInstance(leftX, rightX, values);
47 |         } catch (NoSuchMethodException | InstantiationException |
48 |                  IllegalAccessException | InvocationTargetException e) {
49 |             throw new IllegalArgumentException("Cannot create tabulated function", e);
50 |         }
51 |     }
52 |
53 |     public static TabulatedFunction createTabulatedFunction(Class<? extends TabulatedFunction> functionClass,
54 |                                         FunctionPoint[] points) {
55 |         try {
56 |             Constructor<? extends TabulatedFunction> constructor =
57 |                 functionClass.getConstructor(FunctionPoint[].class);
58 |             return constructor.newInstance((Object) points);
59 |         } catch (NoSuchMethodException | InstantiationException |
60 |                  IllegalAccessException | InvocationTargetException e) {
61 |             throw new IllegalArgumentException("Cannot create tabulated function", e);
62 |         }

```

## Код 3.1

```

public static TabulatedFunction tabulate(Class<? extends TabulatedFunction> functionClass,
                                         Function function, double leftX, double rightX, int pointsCount) {
    if (pointsCount < 2) {
        throw new IllegalArgumentException("Points count must be at least 2");
    }
    if (leftX < function.getLeftDomainBorder() - 1e-10 ||
        rightX > function.getRightDomainBorder() + 1e-10) {
        throw new IllegalArgumentException("Tabulation interval is outside function domain");
    }

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

```

## Код 3.2

```

        System.out.println("\n Задание 3 ");
        TabulatedFunction reflectFunc;

        reflectFunc = TabulatedFunctions.createTabulatedFunction(
                ArrayTabulatedFunction.class, 0, 10, 3);
        System.out.println(reflectFunc.getClass());

        reflectFunc = TabulatedFunctions.createTabulatedFunction(
                ArrayTabulatedFunction.class, 0, 10, new double[] {0, 10});
        System.out.println(reflectFunc.getClass());

        reflectFunc = TabulatedFunctions.createTabulatedFunction(
                LinkedListTabulatedFunction.class,
                new FunctionPoint[] {
                    new FunctionPoint(0, 0),
                    new FunctionPoint(10, 10)
                }
        );
        System.out.println(reflectFunc.getClass());

        reflectFunc = TabulatedFunctions.tabulate(
                LinkedListTabulatedFunction.class, new Sin(), 0, Math.PI, 11);
        System.out.println(reflectFunc.getClass());
    }
}

```

### Код 3.3 (В Main)

```

PS C:\Javalabs\lab7> javac Main.java functions/*.java
PS C:\Javalabs\lab7> java Main

```

**Задание 1**  
 $(0.0; 0.0)$   
 $(2.5; 0.0)$   
 $(5.0; 0.0)$   
 $(7.5; 0.0)$   
 $(10.0; 0.0)$

**Задание 2**  
**class functions.ArrayTabulatedFunction**  
**class functions.LinkedListTabulatedFunction**

**Задание 3**  
**class functions.ArrayTabulatedFunction**  
**class functions.ArrayTabulatedFunction**  
**class functions.LinkedListTabulatedFunction**  
**class functions.LinkedListTabulatedFunction**  
PS C:\Javalabs\lab7> |

