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

Кафедра технической кибернетики

Отчет по лабораторной работе №7

Дисциплина: «ООП»

Тема «Паттерны проектирования и рефлексия»

Выполнил: Чечеткин Д.А.

Группа: 6201-120303D

Самара, 2025

Задание 1

Изменили интерфейс **TabulatedFunction**

```
public interface TabulatedFunction extends Function, Cloneable, Iterable<FunctionPoint> {
```

Иттератор в **ArrayTabulatedFunction**

```
@Override
public Iterator<FunctionPoint> iterator() {
    return new Iterator<FunctionPoint>() {

        private int currentIndex = 0;  2 usages

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

        @Override
        public FunctionPoint next() {
            if (!hasNext()) {
                throw new NoSuchElementException();
            }

            FunctionPoint p = points[currentIndex++];
            return new FunctionPoint(p);
        }

        @Override
        public void remove() {
            throw new UnsupportedOperationException();
        }
    };
```

Иттератор в **LinkedListTabulatedFunction**

```java
@Override
public Iterator<FunctionPoint> iterator() {
    return new Iterator<FunctionPoint>() {

        private FunctionNode current = head.next;  4 usages

        @Override
        public boolean hasNext() {
            return current != head;
        }

        @Override
        public FunctionPoint next() {
            if (!hasNext()) {
                throw new NoSuchElementException();
            }

            FunctionPoint result = new FunctionPoint(current.point);
            current = current.next;
            return result;
        }

        @Override
        public void remove() {
            throw new UnsupportedOperationException();
        }
    };
```

Проверка работы итераторов:

```
Задание 1
ArrayTabulatedFunction:
(0.0; 0.0)
(1.0; 1.0)
(2.0; 4.0)
LinkedListTabulatedFunction:
(0.0; 0.0)
(1.0; 1.0)
(2.0; 4.0)
```

Задание 2

Интерфейс фабрик

```java
package functions;

public interface TabulatedFunctionFactory {   4 usages  2 implementations
    TabulatedFunction createTabulatedFunction(FunctionPoint[] points);  1 usage  2 implementations

    TabulatedFunction createTabulatedFunction(double leftX, double rightX, int pointsCount);

    TabulatedFunction createTabulatedFunction(double leftX, double rightX, double[] values);

}
```

Фабрика в **ArrayTabulatedFunction:**

```java
public static class ArrayTabulatedFunctionFactory  2 usages
        implements TabulatedFunctionFactory {

    @Override  1 usage
    public TabulatedFunction createTabulatedFunction(FunctionPoint[] points) {
        return new ArrayTabulatedFunction(points);
    }

    @Override  1 usage
    public TabulatedFunction createTabulatedFunction(
            double leftX, double rightX, int pointsCount) {
        return new ArrayTabulatedFunction(leftX, rightX, pointsCount);
    }

    @Override  1 usage
    public TabulatedFunction createTabulatedFunction(
            double leftX, double rightX, double[] values) {
        return new ArrayTabulatedFunction(leftX, rightX, values);
    }
}
```

Фабрика в **LinkedListTabulatedFunction:**

```java
public static class LinkedListTabulatedFunctionFactory  1 usage
        implements TabulatedFunctionFactory {

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

    @Override  1 usage
    public TabulatedFunction createTabulatedFunction(
            double leftX, double rightX, int pointsCount) {
        return new LinkedListTabulatedFunction(leftX, rightX, pointsCount);
    }

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

Изменения в **TabulatedFuntions:**

```java
public static void setTabulatedFunctionFactory(TabulatedFunctionFactory newFactory) {  2 usages
    if (newFactory == null) {
        throw new IllegalArgumentException("Фабрика не может быть null");
    }
    factory = newFactory;
}

public static TabulatedFunction createTabulatedFunction(FunctionPoint[] points) {  2 usages
    return factory.createTabulatedFunction(points);
}

public static TabulatedFunction createTabulatedFunction(double leftX, double rightX, int pointsCount) {  no usages
    return factory.createTabulatedFunction(leftX, rightX, pointsCount);
}

public static TabulatedFunction createTabulatedFunction(double leftX, double rightX, double[] values) {  1 usage
    return factory.createTabulatedFunction(leftX, rightX, values);
}
```

Проверка работы фабрик:

```
Задание 2
Исходная фабрика: class functions.ArrayTabulatedFunction
Изменение: class functions.LinkedListTabulatedFunction
Изменение: class functions.ArrayTabulatedFunction
```

Задание 3

Изменения в **TabulatedFuntions:**

```java
public static TabulatedFunction createTabulatedFunction(Class<? extends TabulatedFunction> clazz, FunctionPoint[] points) {  1 usage
    try {
        Constructor<? extends TabulatedFunction> constructor = clazz.getConstructor(FunctionPoint[].class);
        return constructor.newInstance((Object) points);

    } catch (NoSuchMethodException | InstantiationException | IllegalAccessException | InvocationTargetException e) {
        throw new IllegalArgumentException(e);
    }
}

public static TabulatedFunction createTabulatedFunction(Class<? extends TabulatedFunction> clazz, double leftX, double rightX, int pointsCount) {  1 usage
    try {
        Constructor<? extends TabulatedFunction> constructor = clazz.getConstructor(double.class, double.class, int.class);
        return constructor.newInstance(leftX, rightX, pointsCount);

    } catch (NoSuchMethodException | InstantiationException | IllegalAccessException | InvocationTargetException e) {
        throw new IllegalArgumentException(e);
    }
}
public static TabulatedFunction createTabulatedFunction(Class<? extends TabulatedFunction> clazz, double leftX, double rightX, double[] values) {  2 usages
    try {
        Constructor<? extends TabulatedFunction> constructor = clazz.getConstructor(double.class, double.class, double[].class);
        return constructor.newInstance(leftX, rightX, values);

    } catch (NoSuchMethodException | InstantiationException | IllegalAccessException | InvocationTargetException e) {
        throw new IllegalArgumentException(e);
    }
```

Проверка работы рефлексивных методов:

```
Задание 3
class functions.ArrayTabulatedFunction
{(0.0; 0.0), (5.0; 0.0), (10.0; 0.0)}
class functions.ArrayTabulatedFunction
{(0.0; 0.0), (10.0; 10.0)}
class functions.LinkedListTabulatedFunction
{(0.0; 0.0), (10.0; 10.0)}
class functions.LinkedListTabulatedFunction
{(0.0; 0.0), (0.3141592653589793; 0.3090169943749474), (0.6283185307179586; 0.587785252292473l), (0.9424777960769379; 0.8090169943749475),
```