Самарский национальный исследовательский университет имени академика С. П. Королёва

Лабораторная работа №7

Выполнил:
Студент группы 6301-030301D
Абдуллин А.Р.

Задание 1

```
public Iterator<FunctionPoint> iterator() {
    return new Iterator<FunctionPoint>() {
        private int index = 0;  2 usages
        public boolean hasNext() { return index < capacity; }
        public FunctionPoint next() {
            if (!hasNext()) {
                throw new NoSuchElementException();
            }
            FunctionPoint p = array[index++];
            return new FunctionPoint(p.getX(), p.getY());
        }
        public void remove() { throw new UnsupportedOperationException(); }
    };
}
```

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

        private FunctionNode current = head.next;  4 usages
        public boolean hasNext() { return current != head; }
        public FunctionPoint next() {
            if (!hasNext()) {
                throw new NoSuchElementException();
            }
            FunctionPoint p = current.value;
            current = current.next;
            return new FunctionPoint(p.getX(), p.getY());
        }
        public void remove() { throw new UnsupportedOperationException(); }
    };
}
```

```
/usr/java/jdk-23-oracle-x64/bin/java -javaagent:/snap/intellij-idea-community/691/lib/idea_rt.jar=46139 -Dfile.enc
(0.0; 0.0)
(0.1; 0.09983341664682815)
(0.2; 0.19866933079506122)
(0.30000000000000004; 0.2955202066613396)
(0.4; 0.3894183423086505)
(0.5; 0.479425538604203)
(0.6000000000000001; 0.5646424733950355)
(0.7000000000000001; 0.6442176872376911)
(0.8; 0.7173560908995228)
(0.9; 0.7833269096274834)
(0.0; 0.0)
(0.1; 0.09983341664682815)
(0.2; 0.19866933079506122)
(0.30000000000000004; 0.2955202066613396)
(0.4; 0.3894183423086505)
(0.5; 0.479425538604203)
(0.6000000000000001; 0.5646424733950355)
(0.7000000000000001; 0.6442176872376911)
(0.8; 0.7173560908995228)
(0.9; 0.7833269096274834)

Process finished with exit code 0
```

В ходе задания интрефейс TabulatedFunction был расширен Iterable<FunctionPoint> и в классах ArrayTabulatedFunctio
LinkedListTabulatedFunction были реализован методы iterator(), remove(), next(), что позволило использовать

конструкцию for-each к объектам класса. В main классе проверили работоспособность с помощью конструкции for-each.

Задание 2

```
public static class ArrayTabulatedFunctionFactory implements TabulatedFunctionFactory {  1 usage

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

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

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

```
public static class LinkedListTabulatedFunctionFactory implements TabulatedFunctionFactory {  no usages

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

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

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

```
/usr/java/jdk-23-oracle-x64/bin/java -javaagent:/snap/intellij-idea-community/691/lib/idea_rt.jar=33851 -Dfile.encod
class functions.ArrayTabulatedFunction
class functions.LinkedListTabulatedFunction
class functions.ArrayTabulatedFunction

Process finished with exit code 0
```

Реализовал патерн - фабричный метод. Создал интерфейс TabulatedFunctionFactory и определил в нем 3 метода создания TabulatedFunction. В классах реализациях интерфейса TabulatedFunction добавил создал булчиные вложенные классы, создающие объекты соотвествующих реализаций табулированых функций. Обновил класс TabulatedFunctions, добавив статическое поля выбора фабрики, метода для изменения фабрики, обновил методы создающие табулированные функции, для создания функций в соответствии с установленной фабрикой. Проверил работоспособность в main файле, путем изменения фабрик и проверок классов, созданных объектов.

Задание 3

```java
    public static TabulatedFunction createTabulatedFunction(Class<? extends TabulatedFunction> cls, double leftX, double rightX, int pointsCount) {
        try {
            Constructor<? extends TabulatedFunction> ctor = cls.getConstructor(double.class, double.class, int.class);
            return ctor.newInstance(leftX, rightX, pointsCount);
        } catch (InstantiationException | IllegalAccessException | InvocationTargetException | NoSuchMethodException e) {
            throw new IllegalArgumentException(e);
        }
    }
    public static TabulatedFunction createTabulatedFunction(Class<? extends TabulatedFunction> cls, double leftX, double rightX, double[] values) {
        try {
            Constructor<? extends TabulatedFunction> ctor = cls.getConstructor(double.class, double.class, double[].class);
            return ctor.newInstance(leftX, rightX, values);
        } catch (InstantiationException | IllegalAccessException | InvocationTargetException | NoSuchMethodException e) {
            throw new IllegalArgumentException(e);
        }
    }
    public static TabulatedFunction createTabulatedFunction(Class<? extends TabulatedFunction> cls, FunctionPoint[] points) {
        try {
            Constructor<? extends TabulatedFunction> ctor = cls.getConstructor(FunctionPoint[].class);
            return ctor.newInstance((Object) points);
        } catch (InstantiationException | IllegalAccessException | InvocationTargetException | NoSuchMethodException e) {
            throw new IllegalArgumentException(e);
        }
    }

    public static TabulatedFunction tabulate(Class<? extends TabulatedFunction> cls, Function function, double leftX, double rightX, int pointsCount) throws IllegalArgume
}
```

```java
public static TabulatedFunction inputTabulatedFunction(Class<? extends TabulatedFunction> cls, InputStream in) throws IOException {  no usages  new *
    DataInputStream dataIn = new DataInputStream(in);
    int pointCount = dataIn.readInt();
    FunctionPoint[] array = new FunctionPoint[pointCount];
    for (int i = 0; i < pointCount; i++) {
        array[i] = new FunctionPoint(dataIn.readDouble(), dataIn.readDouble());
    }
    return createTabulatedFunction(cls, array);
}
public static TabulatedFunction readTabulatedFunction(Class<? extends TabulatedFunction> cls, Reader in) throws IOException {  no usages  new *
    StreamTokenizer tokenizer = new StreamTokenizer(in);
    tokenizer.parseNumbers();
    tokenizer.nextToken();
    int pointCount = (int)tokenizer.nval;
    FunctionPoint[] array = new FunctionPoint[pointCount];
    double tempX;
    double tempY;
    for (int i = 0; i < pointCount; i++) {
        tokenizer.nextToken();
        tempX = tokenizer.nval;
        tokenizer.nextToken();
        tempY = tokenizer.nval;
        array[i] = new FunctionPoint(tempX, tempY);
    }
    return createTabulatedFunction(cls, array);
}
```

```
/usr/java/jdk-23-oracle-x64/bin/java -javaagent:/snap/intellij-idea-community/691/lib/idea_rt.jar=39081 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.en
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; 0.5877852522924731), (0.9424777960769379; 0.8090169943749475), (1.2566370614359172; 0.9510565

Process finished with exit code 0
```

Рис. 1: Вывод работы main метода

Была реализована возможность создания табулированых функций с разными реализациями хранения данных с помощью рефлексии. В класс TabulatedFunctions были добавленные перегруженные методы для создания TabulatedFunction принимающие параметр Class. Так же был добавлен перегруженные метод tabulate()

принимающий параметр Class возможности выбора класса реализации интерфейса TabulatedFunction. Были добавлены перегруженные методы чтения через рефлексию.