

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 №2

Студент _____ Чячяков А.И.

Группа _____ 6301-030301D

Руководитель _____ Борисов Д. С.

Оценка

Задание 1

Создал пакет functions, в котором далее будут создаваться классы программы.



Задание 2

В пакете functions создал класс FunctionPoint, объект который описывает одну точку табулированной функции.

Состояние объектов содержит два аспекта: координату точки по оси абсцисс и координату точки по оси ординат. При написании класса учел особенности инкапсуляции.

В классе описаны следующие конструкторы:

- FunctionPoint(double x, double y) – создаёт объект точки с заданными координатами;
- FunctionPoint(FunctionPoint point) – создаёт объект точки с теми же координатами, что у указанной точки;
- FunctionPoint() – создаёт точку с координатами (0; 0).

```
© Main.java © FunctionPoint.java × © TabulatedFunction.java
1 package functions;
2
3 public class FunctionPoint { 18 usages
4
5     private double x; 4 usages
6     private double y; 4 usages
7
8
9     public FunctionPoint(double x, double y){ 4 usages
10         this.x = x;
11         this.y = y;
12     }
13
14     @
15     public FunctionPoint(FunctionPoint point){ 3 usages
16         this.x = point.getX();
17         this.y = point.getY();
18     }
19
20     public FunctionPoint(){ no usages
21         this( 0.0, 0.0 );
22     }
23
24     public double getX() { 19 usages
25         return x;
26     }
27
28     public void setX(double x) { 1 usage
29         this.x = x;
30     }
31
32     public double getY() { 8 usages
33         return y;
34     }
35
36     public void setY(double y) { 1 usage
37         this.y = y;
38     }
39 }
```

Задание 3

В пакете functions создал класс TabulatedFunction, объект которого описывает табулированную функцию.

Для хранения данных о точках используется массив типа FunctionPoint. При этом организована работу с массивом так, чтобы точки в нём были всегда упорядочены по значению координаты x.

В классе описаны следующие конструкторы:

- TabulatedFunction(double leftX, double rightX, int pointsCount) – создаёт объект табулированной функции по заданным левой и правой границе области определения, а также количеству точек для табулирования (значения функции в точках при этом следует считать равными 0);
- TabulatedFunction(double leftX, double rightX, double[] values) – аналогичен предыдущему конструктору, но вместо количества точек получает значения функции в виде массива.

В обоих случаях точки должны создаваться через равные интервалы по x.

```
1 package functions;
2
3 public class TabulatedFunction { 3 usages
4     private FunctionPoint[] points; 32 usages
5     private int pointsCount; 20 usages
6
7     public TabulatedFunction(double leftX, no usages
8                             double rightX,
9                             int pointsCount) {
10        this.pointsCount = pointsCount;
11        this.points = new FunctionPoint[pointsCount];
12
13        double step = (rightX - leftX) / (pointsCount - 1);
14
15        for (int i = 0; i < pointsCount; i++) {
16            double x = leftX + step * i;
17            double y = 0.0;
18            points[i] = new FunctionPoint(x, y);
19        }
20    }
21
22 @
23     public TabulatedFunction(double leftX, 1 usage
24                             double rightX,
25                             double[] values) {
26        this.pointsCount = values.length;
27        this.points = new FunctionPoint[pointsCount];
28
29        double step = (rightX - leftX) / (pointsCount - 1);
30
31        for (int i = 0; i < pointsCount; i++) {
32            double x = leftX + step * i;
33            double y = values[i];
34            points[i] = new FunctionPoint(x, y);
35        }
36    }
37 }
```

Задание 4

В классе TabulatedFunction описаны методы, необходимые для работы с функцией.

- Метод double getLeftDomainBorder() возвращает значение левой границы области определения табулированной функции. Очевидно, что оно совпадает с абсциссой самой левой точки в описывающей функцию таблице.
- Аналогично, метод double getRightDomainBorder() возвращает значение правой границы области определения табулированной функции.
- Метод double getFunctionValue(double x) возвращает значение функции в точке x, если эта точка лежит в области определения функции. В противном случае метод должен возвращать значение неопределённости (оно хранится, например, в поле NaN класса Double). При расчёте значения функции используется линейная интерполяция, т.е. считать, что на интервале между заданными в таблице точками функция является прямой линией. Для написания кода метода использовалось уравнением прямой, проходящей через две заданные различающиеся точки.

```
>Main.java    FunctionPoint.java    TabulatedFunction.java ×

3   public class TabulatedFunction { 3 usages ▾ Kirill
4
5
6
7       public double getLeftDomainBorder() { 1 usage ▾ Kirill
8           return points[0].getX();
9       }
10
11      public double getRightDomainBorder() { 1 usage ▾ Kirill
12          return points[pointsCount - 1].getX();
13      }
14
15      public double getFunctionValue(double x) { 3 usages ▾ Kirill
16          if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
17              return Double.NaN;
18          }
19
20          double eps = 1e-9;
21
22          for (int i = 0; i < pointsCount; i++) {
23              if (Math.abs(points[i].getX() - x) < eps) {
24                  return points[i].getY();
25              }
26          }
27
28          int i = 0;
29          while (!(points[i].getX() < x && x < points[i + 1].getX())) {
30              i++;
31          }
32
33          double x1 = points[i].getX();
34          double y1 = points[i].getY();
35          double x2 = points[i + 1].getX();
36          double y2 = points[i + 1].getY();
37
38          return y1 + (x - x1) * (y2 - y1) / (x2 - x1);
39      }
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
```

Задание 5

В классе TabulatedFunction описаны методы, необходимые для работы с точками табулированной функции. Считать, что нумерация точек начинается с ноля.

- Метод `int getPointsCount()` должен возвращать количество точек.
- Метод `FunctionPoint getPoint(int index)` должен возвращать копию точки, соответствующей переданному индексу. Возвращение ссылки на саму точку противоречит принципу инкапсуляции.
- Метод `void setPoint(int index, FunctionPoint point)` должен заменять указанную точку табулированной функции на переданную. Для корректной инкапсуляции замените на копию переданной точки. В случае если координата x задаваемой точки лежит вне интервала, определяемого значениями соседних точек табулированной функции, то замену точки проводить не следует. Например, для функции, определяемой точками $\{(0; 0), (1; 1), (2; 4)\}$, точку с индексом 1 нельзя заменить точкой $(-1; 5)$.
- Метод `double getX(int index)` должен возвращать значение абсциссы точки с указанным номером.
- Метод `void setX(int index, double x)` должен изменять значение абсциссы точки с указанным номером. Аналогично методу `setPoint()`, данный метод не должен изменять точку, если новое значение попадает в другой интервал табулирования.
- Метод `double getY(int index)` должен возвращать значение ординаты точки с указанным номером.
- Метод `void setY(int index, double y)` должен изменять значение ординаты точки с указанным номером.

Код 5

```
public int getPointsCount() {  
    return pointsCount;  
}  
  
public FunctionPoint getPoint(int index) {  
    return new FunctionPoint(points[index]);  
}  
  
public void setPoint(int index,  
                     FunctionPoint point) {  
    double newX = point.getX();  
  
    if (index > 0) {  
        double leftX = points[index - 1].getX();  
        if (newX <= leftX) {  
            return;  
        }  
    }  
  
    if (index < pointsCount - 1) {
```

```

        double rightX = points[index + 1].getX();
        if (newX >= rightX) {
            return;
        }
    }

    points[index] = new FunctionPoint(point);
}

public double getPointX(int index) {
    return points[index].getX();
}

public void setPointX(int index, double x) {
    if (index > 0) {
        double leftX = points[index - 1].getX();
        if (x <= leftX) {
            return;
        }
    }

    if (index < pointsCount - 1) {
        double rightX = points[index + 1].getX();
        if (x >= rightX) {
            return;
        }
    }

    points[index].setX(x);
}

public double getPointY(int index) {
    return points[index].getY();
}

public void setPointY(int index, double y) {
    points[index].setY(y);
}

```

6 задание

В классе TabulatedFunction описаны методы, изменяющие количество точек табулированной функции.

- Метод void deletePoint(int index) удаляет заданную точку табулированной функции.
- Метод void addPoint(FunctionPoint point) добавляет новую точку табулированной функции. При написании метода обеспечил корректную инкапсуляцию.

При написании методов учитывается, что точки в массиве должны быть упорядочены по значению координаты x.

Для копирования участков массивов воспользовался методом `arraycopy()` класса `System`.

Также следует понимать, что создание нового массива каждый раз при выполнении операций удаления и вставки точки является расточительством по отношению к памяти и скорости работы программы. Поэтому длина массива в общем случае не должна совпадать с количеством точек в табулированной функции, а замена массива на массив большей длины должна производиться только в некоторых случаях.

Код 6

```
public void deletePoint(int index) {
    if (pointsCount <= 1) {
        return;
    }

    if (index < pointsCount - 1) {
        System.arraycopy(points,
                        index + 1,
                        points,
                        index,
                        pointsCount - index - 1);
    }

    pointsCount--;
    points[pointsCount] = null;
}

public void addPoint(FunctionPoint point) {
    if (pointsCount == points.length) {
        return;
    }

    double x = point.getX();

    int insertIndex = 0;
    while (insertIndex < pointsCount && points[insertIndex].getX() <
x) {
        insertIndex++;
    }

    if (insertIndex < pointsCount) {
        System.arraycopy(points,
                        insertIndex,
                        points,
                        insertIndex + 1,
                        pointsCount - insertIndex);
    }

    points[insertIndex] = new FunctionPoint(point);
```

```
    pointsCount++;  
}
```

Задание 7

Проверил работу написанных классов.

В пакете по умолчанию (вне пакета functions) нужно создать класс Main, содержащий точку входа программы.

В методе main() создал экземпляр класса TabulatedFunction и задал для него табулированные значения функции.

Вывел в консоль значения функции на ряде точек. Рекомендуется использовать такой шаг (или такие точки), чтобы среди них оказались точки вне области определения функции, а также чтобы несколько точек попали в один интервал табулированной функции.

Проверил, как изменяется результат работы программы после изменения точек, добавления и удаления точек.

Код 7

```
>Main.java  FunctionPoint.java  TabulatedFunction.java :  
4 ▷ public class Main {  
5 ▷     public static void main(String[] args) {  
6         double left = 0.0;  
7         double right = 4.0;  
8         double[] values = {0.0, 1.0, 4.0, 9.0, 16.0};  
9  
10        TabulatedFunction f = new TabulatedFunction(left, right, values);  
11  
12        System.out.println("Точки сразу после конструктора");  
13        for (int i = 0; i < f.getPointsCount(); i++) {  
14            FunctionPoint p = f.getPoint(i);  
15            System.out.println("Точка " + i + ": (" + p.getX() + "; " + p.getY() + ")");  
16        }  
17  
18        System.out.println("Значения функции (в том числе вне области)");  
19        for (double x = -1.0; x <= 5.0; x += 1.0) {  
20            double y = f.getFunctionValue(x);  
21            System.out.println("x = " + x + ", f(x) = " + y);  
22        }  
23  
24        System.out.println("Изменение точки с индексом 2");  
25        f.setPointY(index: 2, y: 100.0);  
26        System.out.println("Новая точка 2: (" + f.getPointX(index: 2) + "; " + f.getPointY(index: 2) + ")");  
27  
28        System.out.println("Добавление новой точки");  
29        FunctionPoint newPoint = new FunctionPoint(x: 2.5, y: 6.25);  
30        f.addPoint(newPoint);  
31        for (int i = 0; i < f.getPointsCount(); i++) {  
32            FunctionPoint p = f.getPoint(i);  
33            System.out.println("Точка " + i + ": (" + p.getX() + "; " + p.getY() + ")");  
34        }  
35  
36        System.out.println("Удаление точки с индексом 1");  
37        f.deletePoint(index: 1);  
38        for (int i = 0; i < f.getPointsCount(); i++) {  
39            FunctionPoint p = f.getPoint(i);  
40        }
```

Вывод:

Точки сразу после конструктора

Точка 0: (0.0; 0.0)

Точка 1: (1.0; 1.0)

Точка 2: (2.0; 4.0)

Точка 3: (3.0; 9.0)

Точка 4: (4.0; 16.0)

Значения функции в узлах:

x = 0.0, f(x) = 0.0

x = 1.0, f(x) = 1.0

x = 2.0, f(x) = 4.0

x = 3.0, f(x) = 9.0

x = 4.0, f(x) = 16.0

Значения функции между узлами (проверка интерполяции):

x = 0.5, f(x) = 0.5

x = 1.5, f(x) = 2.5

x = 2.5, f(x) = 6.5

x = 3.5, f(x) = 12.5

Значения функции вне области:

x = -1.0, f(x) = NaN

x = 5.0, f(x) = NaN

Изменение точки с индексом 2

Новая точка 2: (2.0; 100.0)

Добавление новой точки

Точка 0: (0.0; 0.0)

Точка 1: (1.0; 1.0)

Точка 2: (2.0; 100.0)

Точка 3: (3.0; 9.0)

Точка 4: (4.0; 16.0)

Удаление точки с индексом 1

Точка 0: (0.0; 0.0)

Точка 1: (2.0; 100.0)

Точка 2: (3.0; 9.0)

Точка 3: (4.0; 16.0)

Process finished with exit code 0

