Лабораторная работа №3 по дисциплине «Объектно-ориентированное программирование»

Выполнил: Чичкин Данила Александрович

Группа: 6204-010302D

Оглавление

Изменения кода

Перед выполнением лабораторной работы я изучил основные принципы документирования кода в Java. Из классов FunctionPoint и TabulatedFunction я убрал все однострочные комментарии и добавил к каждому методу, а также и к самим классам, JavaDoc комментарии с тегами (см. скрин 1).

```
1   package functions;
2
    /**
     * Класс для работы с табулированными функциями одной переменной. Функция задаётся таблицей
     * точек, упорядоченных по координате X. Для создания точек используется класс FunctionPoint
     *
     * See Also: FunctionPoint
     *
     * Version: 1.1
     *
     * Author: haidyonish
     */
12  public class TabulatedFunction {  3 usages
13      private FunctionPoint[] points;  33 usages
14      private int size;  26 usages
15
        /**
         * Создаёт табулированную функцию с равномерной сеткой и нулевыми значениями.
         *
         * Params: leftX – левая граница области определения
         *         rightX – правая граница области определения
         *         pointsCount – количество точек табулирования (не менее 2)
         */
23      public TabulatedFunction(double leftX, double rightX, int pointsCount) {  12 usages
24          size = pointsCount;
25          points = new FunctionPoint[size];
26          double step = (rightX - leftX) / (size - 1);
27          for (int i = 0; i < (size - 1); i++) {
28              points[i] = new FunctionPoint( x: leftX + i*step,  y: 0);
29          }
30          points[size-1] = new FunctionPoint(rightX,  y: 0);
31      }
```

Скрин 1 – JavaDoc комментарии к коду

Задание 1

Я ознакомился со следующими исключениями:

– java.lang.Exception – базовый класс проверяемых исключений;

– java.lang.IndexOutOfBoundsException – указывает, что какой-то индекс находится вне допустимого диапазона;

– java.lang.ArrayIndexOutOfBoundsException – схож с предыдущим исключением, но используется только в контексте массивов;

– java.lang.IllegalArgumentException – указывает, что в метод был передан недопустимый параметр;

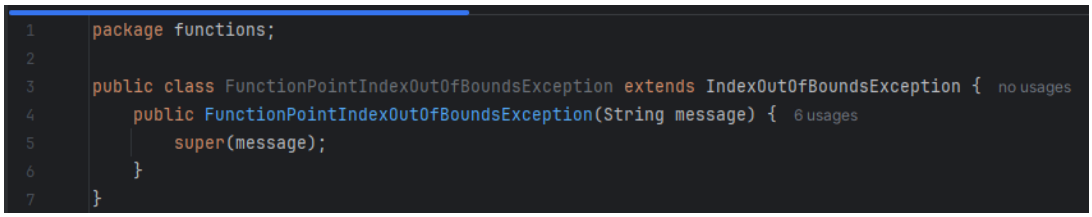
– java.lang.IllegalStateException – указывает на недопустимое состояние объекта при попытке его изменить.

Задание 2

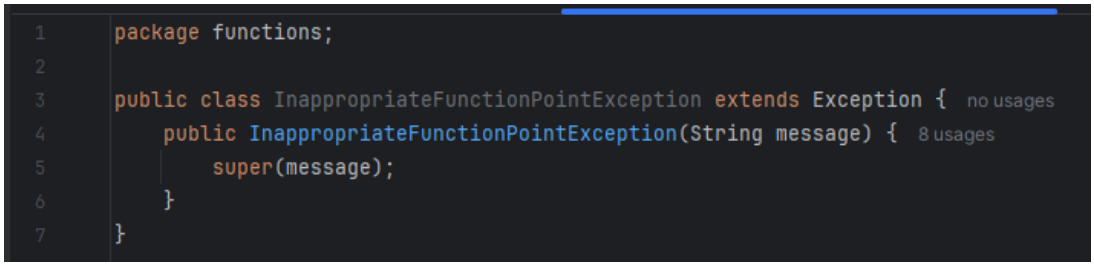
В пакете functions я создал два класса исключений:

– FunctionPointIndexOutOfBoundsException – исключение выхода за границы набора точек при обращении к ним по номеру, наследует от класса IndexOutOfBoundsException (см. скрин 2);

– InappropriateFunctionPointException – исключение, выбрасываемое при попытке добавления или изменения точки функции несоответствующим образом, наследует от класса Exception (см. скрин 3).



Скрин 2 – класс FunctionPointIndexOutOfBoundsException



Скрин 3 – класс InappropriateFunctionPointException

При создании классов я использовал средства IDE IntelliJ IDEA для автоматической генерации конструкторов.

Задание 3

В классе TabulatedFunction я внёс изменения в код и JavaDoc комментарии.

Конструкторы теперь выбрасывают исключение IllegalArgumentException, если левая граница области определения больше или равна правой, а также если предлагаемое количество точек меньше двух (см. скрин 4).

```
    Создаёт табулированную функцию с равномерной сеткой и нулевыми значениями.
    Params: leftX – левая граница области определения (меньше правой)
            rightX – правая граница области определения
            pointsCount – количество точек табулирования (не менее двух)
    Throws: IllegalArgumentException – если левая граница области определения больше
            или равна правой или если количество точек менее двух
24  public TabulatedFunction(double leftX, double rightX, int pointsCount) {  12 usages
25      if (leftX >= rightX)
26          throw new IllegalArgumentException("Левая граница области определения leftX должна быть строго меньше правой границы rightX");
27      if (pointsCount < 2)
28          throw new IllegalArgumentException("Количество точек pointsCount должно быть не меньше двух");
29      size = pointsCount;
30      points = new FunctionPoint[size];
31      double step = (rightX - leftX) / (size - 1);
32      for (int i = 0; i < (size - 1); i++) {
33          points[i] = new FunctionPoint( x: leftX + i*step,  y: 0);
34      }
35      points[size-1] = new FunctionPoint(rightX,  y: 0);
36  }
37
    Создаёт табулированную функцию с равномерной сеткой и заданными значениями.
    Params: leftX – левая граница области определения (меньше правой)
            rightX – правая граница области определения
            values – значения функции в точках табулирования (не менее двух значений в
            массиве)
    Throws: IllegalArgumentException – если левая граница области определения больше
            или равна правой или если в массиве менее двух значений
46  public TabulatedFunction(double leftX, double rightX, double[] values) {  13 usages
47      if (leftX >= rightX)
48          throw new IllegalArgumentException("Левая граница области определения leftX должна быть строго меньше правой границы rightX");
49      if (values.length < 2)
50          throw new IllegalArgumentException("Количество значений функции в массиве values должно быть не меньше двух");
51      size = values.length;
52      points = new FunctionPoint[size];
53      double step = (rightX - leftX) / (size - 1);
54      for (int i = 0; i < (size - 1); i++) {
55          points[i] = new FunctionPoint( x: leftX + i*step, values[i]);
56      }
57      points[size-1] = new FunctionPoint(rightX, values[size-1]);
58  }
```

Скрин 4 – изменения конструкторов

Я написал два приватных метода (см. скрин 5):

– checkIndexBounds(int index) – этот метод выбрасывает исключение FunctionPointIndexOutOfBoundsException, если переданный в метод номер выходит за границы набора точек;

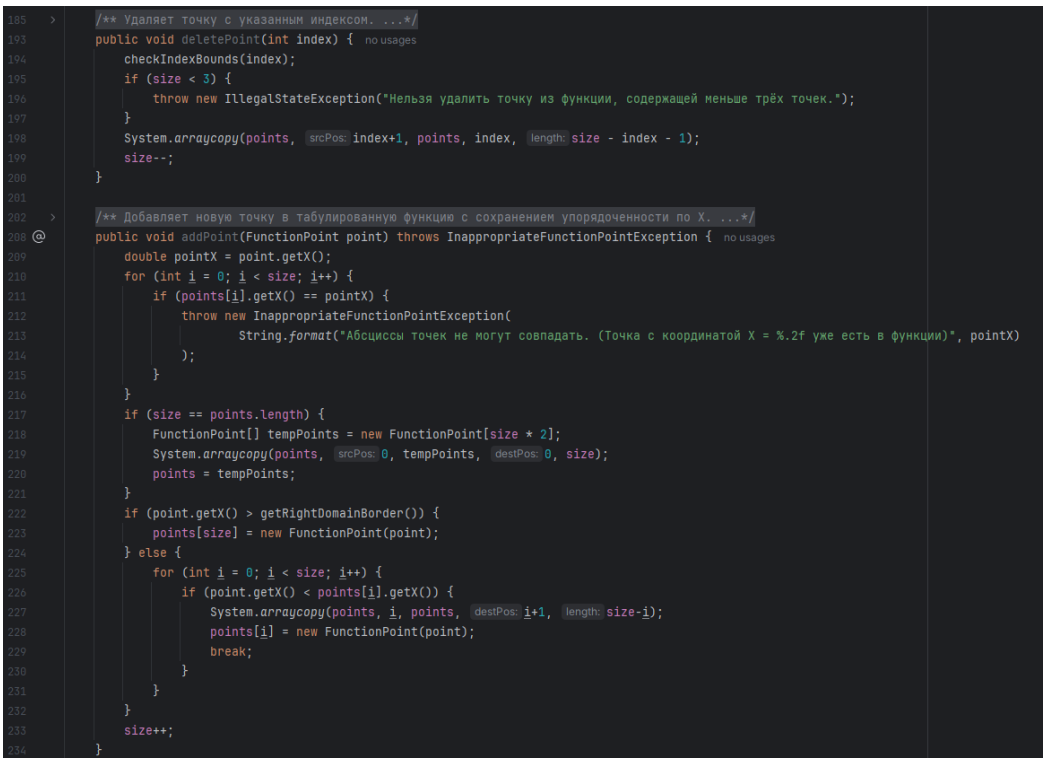
– checkPointOrder(int index, double pointX) – этот метод выбрасывает исключение InappropriateFunctionPointException в том случае, если координата x

задаваемой точки лежит вне интервала, определяемого значениями соседних точек табулированной функции.

Скрин 5 – методы checkIndexBounds и checkPointOrder

В методах getPoint(), getPointX(), getPointY(), setPointY() и deletePoint() в начале вызывается checkIndexBounds В методах setPoint() и setPointX(), в начале вызываются оба метода.

Метод addPoint() теперь выбрасывает исключение InappropriateFunctionPointException, если в наборе точек функции есть точка, абсцисса которой совпадает с абсциссой добавляемой точки. Метод deletePoint() теперь выбрасывает исключение IllegalStateException, если на момент удаления точки количество точек в наборе менее трех (см. скрин 6).



Скрин 6 – изменения методов addPoint и deletePoint

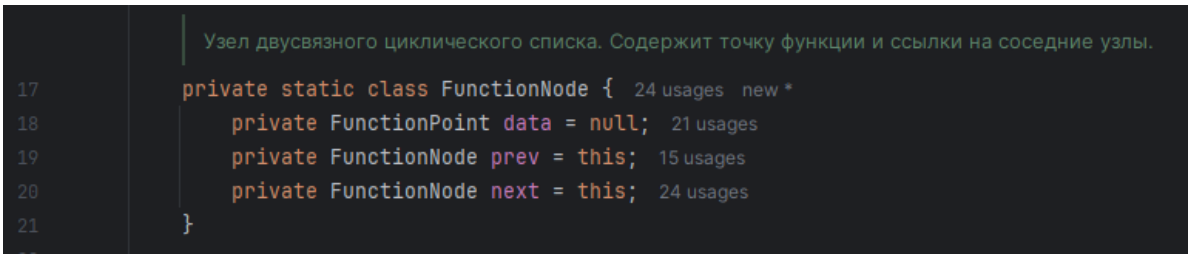
Задание 4

Внутри класса LinkedListTabulatedFunction был описан внутренний класс FunctionNode. Этот класс объявлен как private static, что обеспечивает его использование только внутри внешнего класса и предотвращает нарушение инкапсуляции. Класс FunctionNode содержит следующие поля:

– data – для хранения объекта FunctionPoint (точка функции);

– prev – ссылка на предыдущий элемент списка;

– next – ссылка на следующий элемент списка.



Скрин 7 – Элемент двусвязного списка

В классе LinkedListTabulatedFunction были добавлены поля:

– head – фиктивный узел (голова списка), образующий циклическую структуру;

– length – количество точек функции (значений, исключая голову);

– lastAccessedNode и lastAccessedNodeIndex – для оптимизации доступа к элементам списка.

Были реализованы следующие методы для работы со связным списком:

– getNodeByIndex(int index) – возвращает узел по индексу, используя кеширование для уменьшения количества проходов по списку;

– addNodeToTail() – добавляет новый узел в конец списка;

– addNodeByIndex(int index) – вставляет узел в указанную позицию;

– deleteNodeByIndex(int index) – удаляет узел по индексу и возвращает ссылку на удаленный узел.

Все методы работы со списком инкапсулированы внутри класса LinkedListTabulatedFunction, что соответствует принципам объектно-ориентированного проектирования.

Задание 5

В классе LinkedListTabulatedFunction были реализованы конструкторы и методы, аналогичные конструкторам и методам класса ArrayTabulatedFunction.

Конструкторы класса LinkedListTabulatedFunction принимают те же параметры и выполняют аналогичную проверку входных данных. При некорректных значениях границ области определения или количестве точек выбрасывается исключение IllegalArgumentException.

Для работы с точками функции были реализованы методы getPoint(), setPoint(), getPointX(), setPointX(), getPointY(), setPointY(), addPoint() и deletePoint(). Все методы выбрасывают те же типы исключений и в тех же ситуациях, что и соответствующие методы в ArrayTabulatedFunction.

При реализации методов активно использовались вспомогательные методы работы со связным списком, такие как getNodeByIndex(), addNodeByIndex() и deleteNodeByIndex(). Это позволило избежать дублирования логики и повысить читаемость кода.

В ряде методов была выполнена оптимизация за счёт прямого доступа к узлам списка. Например, при последовательных обращениях к соседним точкам использовался механизм кеширования последнего узла, что снижает количество проходов по списку.

Таким образом, класс LinkedListTabulatedFunction полностью повторяет функциональность ArrayTabulatedFunction, но использует связный список в качестве внутренней структуры данных.

Задание 6

Класс TabulatedFunction был переименован в ArrayTabulatedFunction, так как его реализация основана на хранении точек функции в массиве.

После этого был создан интерфейс TabulatedFunction, содержащий объявления общих методов для работы с табулированными функциями. В интерфейс были вынесены сигнатуры всех публичных методов, необходимых для получения и изменения точек функции, а также для работы с областью определения.

Классы ArrayTabulatedFunction и LinkedListTabulatedFunction были изменены таким образом, чтобы реализовывать созданный интерфейс TabulatedFunction. Это позволило отделить описание функциональности от конкретной реализации хранения данных.

```
1   package functions;
2
3   /** Интерфейс табулированной функции. ...*/
8   public interface TabulatedFunction {  10 usages  2 implementations  new *
9
10      /** Возвращает значение функции в заданной точке {@code x}. ...*/
18      double getFunctionValue(double x);  1 usage  2 implementations  new *
19
20      /** Возвращает количество точек в функции. ...*/
25      int getPointsCount();  3 usages  2 implementations  new *
26
27      /** Возвращает объект точки с указанным индексом. ...*/
34      FunctionPoint getPoint(int index);  1 usage  2 implementations  new *
35
36      /** Устанавливает новое значение точки по индексу. ...*/
44      void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException;  no usages  2 implementations  new *
45
46      /** Возвращает значение X точки с указанным индексом. ...*/
53      double getPointX(int index);  19 usages  2 implementations  new *
54
55      /** Возвращает значение Y точки с указанным индексом. ...*/
62      double getPointY(int index);  3 usages  2 implementations  new *
63
64      /** Изменяет значение X точки с указанным индексом. ...*/
72      void setPointX(int index, double x) throws InappropriateFunctionPointException;  1 usage  2 implementations  new *
73
74      /** Изменяет значение Y точки с указанным индексом. ...*/
81      void setPointY(int index, double y);  1 usage  2 implementations  new *
82
83      /** Добавляет новую точку в функцию. ...*/
90      void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;  2 usages  2 implementations  new *
91
92      /** Удаляет точку с указанным индексом. ...*/
100     void deletePoint(int index);  2 usages  2 implementations  new *
101
102     /** Возвращает минимальное значение X среди всех точек функции. ...*/
107     double getLeftDomainBorder();  3 usages  2 implementations  new *
108
109     /** Возвращает максимальное значение X среди всех точек функции. ...*/
114     double getRightDomainBorder();  5 usages  2 implementations  new *
115 }
116
```

Скрин 8 – Интерфейс

Задание 7

Для проверки корректности работы реализованных классов был доработан класс Main, содержащий точку входа в программу.

Ссылочная переменная для работы с табулированной функцией была объявлена типа TabulatedFunction, а объект создавался как экземпляр конкретного класса реализации. Это позволило легко переключаться между ArrayTabulatedFunction и LinkedListTabulatedFunction, не изменяя остальной код программы.

В методе main() были добавлены тестовые сценарии, проверяющие выбрасывание исключений в различных ситуациях: при обращении к несуществующим точкам, при попытке задать некорректные значения координат, а также при удалении точки при недостаточном количестве элементов.

Результаты работы программы подтверждают корректность реализации всех классов и соответствие их поведения условиям лабораторной работы.

```
=== Проверка ArrayTabulatedFunction ===
Количество точек: 5
Границы области: [0,00, 10,00]

Значения функции:
x=0,00, y=0,00
x=2,50, y=0,00
x=5,00, y=0,00
x=7,50, y=0,00
x=10,00, y=0,00

Изменение точек:
После изменения: y(5,00) = 10,00
Интерполяция при x=4,50 -> y=8,00

Добавление новой точки:
Точка (11, 5) добавлена успешно.

Удаление точки:
Точка с индексом 1 удалена успешно.

Итоговые точки:
[0] x=0,00, y=0,00
[1] x=5,00, y=10,00
[2] x=7,50, y=0,00
[3] x=10,00, y=0,00
[4] x=11,00, y=5,00
```

Скрин 9 – Проверка класса ArrayTabulatedFunction

```
=== Проверка LinkedListTabulatedFunction ===
Количество точек: 5
Границы области: [0,00, 10,00]

Значения функции:
x=0,00, y=0,00
x=2,50, y=0,00
x=5,00, y=0,00
x=7,50, y=0,00
x=10,00, y=0,00

Изменение точек:
После изменения: y(5,00) = 10,00
Интерполяция при x=4,50 -> y=8,00

Добавление новой точки:
Точка (11, 5) добавлена успешно.

Удаление точки:
Точка с индексом 1 удалена успешно.

Итоговые точки:
[0] x=0,00, y=0,00
[1] x=5,00, y=10,00
[2] x=7,50, y=0,00
[3] x=10,00, y=0,00
[4] x=11,00, y=5,00
```

Скрин 10 – Проверка класса LinkedListTabulatedFunction

```
=== Проверка выбрасывания исключений ===
Создание функции с одной точкой:
Ожидаемая ошибка: Количество точек pointsCount должно быть не меньше двух

Создание функции с одинаковыми границами:
Ожидаемая ошибка: Левая граница области определения leftX должна быть строго меньше правой границы rightX

Попытка установить X, нарушающий порядок:
Ожидаемая ошибка: Координата x должна лежать между 0,00 и 4,00 (при индексе 1)

Попытка добавить точку с существующим X:
Ожидаемая ошибка: Абсциссы точек не могут совпадать. (Точка с координатой X = 2,00 уже есть в функции)

Попытка удалить точку при размере 2:
Ожидаемая ошибка: Нельзя удалить точку из функции, содержащей меньше трёх точек.

Попытка обратиться к несуществующему индексу:
Ожидаемая ошибка: Индекс точки должен быть не меньше нуля и меньше количества точек в функции. (На данный момент количество точек в функции - 3)
```

Скрин 11 – Проверка выбрасывания исключений