Отчёт
Лабораторная работа №3

Выполнил: Салихов Камран Рустам угли
Группа: 6204-010302D

В ходе выполнения задания 1 ознакомился со следующими классами исключений, входящих в API Java:

java.lang.Exception - базовый класс для проверяемых исключений

java.lang.IndexOutOfBoundsException - выбрасывается при попытке доступа к элементу коллекции по индексу, который выходит за её пределы

java.lang.ArrayIndexOutOfBoundsException – проверка индекса но для массивов

java.lang.IllegalArgumentException - выбрасывается, когда методу передали некорректное значение параметра

java.lang.IllegalStateException - выбрасывается, когда объект находится в недопустимом для данной операции состоянии

В ходе выполнения задания 2 были созданы два класса исключений и размещены в пакет **functions**:

FunctionPointIndexOutOfBoundsException - исключение выхода за границы набора точек при обращении к ним по номеру, наследует от класса **IndexOutOfBoundsException**

```
public class FunctionPointIndexOutOfBoundsException extends IndexOutOfBoundsException {
}
```

InappropriateFunctionPointException - исключение, выбрасываемое при попытке добавления или изменения точки функции несоответствующим образом, наследует от класса **Exception**.

```
public class InappropriateFunctionPointException extends Exception {
}
```

В ходе выполнения задания 3 в разработанный ранее класс **TabulatedFunction** были внесены изменения, обеспечивающие выбрасывание исключений методами класса.

Так, оба конструктора класса выбрасывают исключение **IllegalArgumentException**, если левая граница области определения больше или равна правой, а также если предлагаемое количество точек меньше двух.

```
if (leftX >= rightX) {
    throw new IllegalArgumentException("левая граница области определения должна быть меньше правой");}
if (pointsCount < 2) {
    throw new IllegalArgumentException("количество точек должно быть не менее двух");}
```

Методы **getPoint(), setPoint(), getPointX(), setPointX(), getPointY(), setPointY()** и **deletePoint()** выбрасывают исключение **FunctionPointIndexOutOfBoundsException**, если переданный в метод номер выходит за границы набора точек.

```
if (index < 0 || index >= size) {
    throw new FunctionPointIndexOutOfBoundsException();}
```

Методы **setPoint()** и **setPointX()** выбрасывают исключение **InappropriateFunctionPointException**, если координата x задаваемой точки лежит вне интервала, определяемого значениями соседних точек табулированной функции. Метод **addPoint()** выбрасывает **InappropriateFunctionPointException**, если в наборе точек функции есть точка, абсцисса которой совпадает с абсциссой добавляемой точки.

Метод **deletePoint()** выбрасывает исключение **IllegalStateException**, если на момент удаления точки количество точек в наборе менее трех.

```
if (size < 3) {
    throw new IllegalStateException("количество точек должно быть не менее трех");
}
```

В ходе выполнения задания 4 в пакете **functions** был создан класс **LinkedListTabulatedFunction**, который также описывает табулированную функцию, но для хранения набора точек в нем используется не массив, а динамическая структура - двусвязный циклический список с выделенной головой. Его особенность в том, что каждый узел содержит: хранимые данные (объект **FunctionPoint**), **prev** - ссылку на предыдущий элемент, **next** - ссылку на следующий элемент. Есть голова списка, которая не хранит данных, всегда присутствует в списке и связывает начало и конец списка. Все элементы связаны в кольцо через голову.

Был описан класс элементов списка **FunctionNode**, содержащий информационное поле для хранения данных типа **FunctionPoint**, а также поля для хранения ссылок на предыдущий и следующий элемент. Класс вложен внутри **LinkedListTabulatedFunction**, так как является неотъемлемой частью реализации связного списка и существует только в контексте **LinkedListTabulatedFunction**. Инкапсуляция в **FunctionNode** такова, что все поля приватные для защиты данных - точка **FunctionPoint** не может быть изменена извне напрямую; сохраняется целостность - ссылки **prev** и **next** изменяются только контролируемо.

Был описан класс **LinkedListTabulatedFunction** объектов списка, содержащий поле ссылки на объект головы, а также иные вспомогательные поля.

В классе **LinkedListTabulatedFunction** был реализован метод **FunctionNode getNodeByIndex**(**int index**), возвращающий ссылку на объект элемента списка по его номеру.

В классе **LinkedListTabulatedFunction** был реализован метод **FunctionNode addNodeToTail**(), добавляющий новый элемент в конец списка и возвращающий ссылку на объект этого элемента.

В классе **LinkedListTabulatedFunction** был реализован метод **FunctionNode addNodeByIndex**(**int index**), добавляющий новый элемент в указанную позицию списка и возвращающий ссылку на объект этого элемента.

В классе **LinkedListTabulatedFunction** был реализован метод **FunctionNode deleteNodeByIndex**(**int index**), удаляющий элемент списка по номеру и возвращающий ссылку на объект удаленного элемента.

В ходе выполнения задания 5 в классе **LinkedListTabulatedFunction** были реализованы конструкторы и методы, аналогичные конструкторам и методам класса **TabulatedFunction**:

public LinkedListTabulatedFunction(double leftX, double rightX, int pointsCount)

public LinkedListTabulatedFunction(double leftX, double rightX, double[] values)

public double getLeftDomainBorder()

public double getRightDomainBorder()

public double getFunctionValue(double x)

public int getPointsCount()

public FunctionPoint getPoint(int index)

public double getPointX(int index)

public double getPointY(int index)

public void setPoint(int index, FunctionPoint point)

public void setPointX(int index, double x)

public void setPointY(int index, double y)

public void deletePoint(int index)

public void addPoint(FunctionPoint point)

Конструкторы имеют те же параметры, методы имеют те же сигнатуры, выбрасываются те же виды исключений в тех же случаях.

В ходе выполнения задания 6 класс **TabulatedFunction** был переименован в класс **ArrayTabulatedFunction**. Был создан интерфейс **TabulatedFunction**, содержащий объявления общих методов классов **ArrayTabulatedFunction** и **LinkedListTabulatedFunction**. Оба класса функций реализуют данный интерфейс через **implements TabulatedFunction**.

```java
package functions;

public interface TabulatedFunction {  5 usages  2 implementations
    double getLeftDomainBorder();  2 usages  2 implementations
    double getRightDomainBorder();  2 usages  2 implementations
    double getFunctionValue(double x);  no usages  2 implementations
    int getPointsCount();  1 usage  2 implementations
    FunctionPoint getPoint(int index);  2 usages  2 implementations
    void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException;
    double getPointX(int index);  1 usage  2 implementations
    void setPointX(int index, double x) throws InappropriateFunctionPointException;  2 usages  2
    double getPointY(int index);  1 usage  2 implementations
    void setPointY(int index, double y);  no usages  2 implementations
    void deletePoint(int index);  2 usages  2 implementations
    void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;  4 usages  2 i
}
```

В ходе выполнения задания 7 была совершенна проверка написанных классов. Результат работы программы при введенных параметрах:

начальные (-1.0, 1.0, 5)

замена (0.57, 0.22), индекс замены (3)

добавление (0.71, 0.34)

индекс удаления (0)

значения Y для **ArrayTabulatedFunction**: (1.0, 0.5, 0.0, -0.5, -1.0)

значения Y для **LinkedListTabulatedFunction**: (2.0, 1.0, 0.0, -1.0, -2.0)

Проверка **ArrayTabulatedFunction**:

```
тестирование ArrayTabulatedFunction
начальные точки:
количество точек 5
   x = -1.0 y = 1.0
   x = -0.5 y = 0.5
   x = 0.0 y = 0.0
   x = 0.5 y = -0.5
   x = 1.0 y = -1.0

проверка getFunctionValue:
f(-1.0) = 1.0
f(-0.5) = 0.5
f(0.0) = 0.0
f(0.3) = -0.3
f(1.0) = -1.0
f(2.0) (вне диапазона) = NaN

после замены точки:
количество точек 5
   x = -1.0 y = 1.0
   x = -0.5 y = 0.5
   x = 0.0 y = 0.0
   x = 0.57 y = 0.22
   x = 1.0 y = -1.0
после добавления точки:
количество точек 6
   x = -1.0 y = 1.0
   x = -0.5 y = 0.5
   x = 0.0 y = 0.0
   x = 0.57 y = 0.22
   x = 0.71 y = 0.34
   x = 1.0 y = -1.0

проверка исключений
поймано FunctionPointIndexOutOfBoundsException при getPoint(1234)
поймано InappropriateFunctionPointException при setPointX(2, 10)
поймано InappropriateFunctionPointException при добавлении существующей точки (0.71, 0.34)
```

```
после удаления точки с индексом 0:
количество точек 5
   x = -0.5 y = 0.5
   x = 0.0 y = 0.0
   x = 0.57 y = 0.22
   x = 0.71 y = 0.34
   x = 1.0 y = -1.0
```

Проверка **LinkedListTabulatedFunction**:

```
тестирование LinkedListTabulatedFunction
начальные точки:
количество точек 5
   x = -1.0 y = 2.0
   x = -0.5 y = 1.0
   x = 0.0 y = 0.0
   x = 0.5 y = -1.0
   x = 1.0 y = -2.0

проверка getFunctionValue:
f(-1.0) = 2.0
f(-0.5) = 1.0
f(0.0) = 0.0
f(0.3) = -0.6
f(1.0) = -2.0
f(2.0) (вне диапазона) = NaN

после замены точки:
количество точек 5
   x = -1.0 y = 2.0
   x = -0.5 y = 1.0
   x = 0.0 y = 0.0
   x = 0.57 y = 0.22
   x = 1.0 y = -2.0
после добавления точки:
количество точек 6
   x = -1.0 y = 2.0
   x = -0.5 y = 1.0
   x = 0.0 y = 0.0
   x = 0.57 y = 0.22
   x = 0.71 y = 0.34
   x = 1.0 y = -2.0

проверка исключений
поймано FunctionPointIndexOutOfBoundsException при getPoint(1234)
поймано InappropriateFunctionPointException при setPointX(2, 10)
поймано InappropriateFunctionPointException при добавлении существующей точки (0.71, 0.34)
```

```
после удаления точки с индексом 0:
количество точек 5
   x = -0.5 y = 1.0
   x = 0.0 y = 0.0
   x = 0.57 y = 0.22
   x = 0.71 y = 0.34
   x = 1.0 y = -2.0

Process finished with exit code 0
```