Лабораторная работа №3

Автор: Ежова Екатерина

Группа: 6203-010302D

Задание 2

Создаем в пакете functions 2 класса: FunctionPointIndexOutOfBoundsException и InappropriateFunctionPointException. В каждом классе 2 конструктора – один без параметров, внутри него расположен super(), который вызывает конструктор родительского класса, и второй – конструктор, в который мы передаем строку, и внутри него также вызываем конструктор родительского класса.

Задание 3

Во все указанные в задании методы добавляем проверку условий для выбрасывания исключений. Чтобы выбросить исключение пишем ключевое слово throw, потом оператор new и затем тип самого исключения. Если исключение проверяемое как InappropriateFunctionPointException, то мы должны объявить его в названии метода с помощью ключевого слова throws.

Задание 4

Внутри класса LinkedListTabulatedFunction создаем внутренний класс FunctionNode, который имеет три приватных поля, хранящих саму точку и ссылку на следующий и предыдущий элемент. В классе FunctionNode есть два конструктора – первый принимает в качестве аргумента точку типа FunctionPoint и создает объект со значением этой точки, второй конструктор – без параметров создает объект, в котором все поля равны null. Также во внутреннем классе есть сеттеры и геттеры для каждого поля, они позволяют записывать и получать значения из внутреннего класса.

Во внешнем классе LinkedListTabulatedFunction есть еще 2 приватных поля- len хранит длину списка, а head типа FunctionNode является головой списка.

Далее во внешнем классе я реализовала метод FunctionNode getNodeByIndex(int index), возвращающий ссылку на объект элемента списка по его номеру. Чтобы оптимизировать поиск, проверяем в первой или второй половине находится индекс, если индекс ближе к началу, перемещаем указатель текущего элемента, на следующий элемент, пока не дойдем до нужного индекса, иначе идем с конца и перемещаем указатель на предыдущий элемент. Возвращаем ссылку на итоговый элемент.

Следующий метод FunctionNode addNodeToTail(FunctionPoint point), добавляет новый элемент в конец списка и возвращает ссылку на объект этого элемента. В этом методе создаем узел со значением новой точки, делаем проверку на то, является ли список пустым, потом меняем связи у соседних элементов

Еще один метод FunctionNode addNodeByIndex(int index, FunctionPoint point) добавляет новый элемент в указанную позицию списка и возвращает ссылку на объект этого элемента. Создаем новый узел, если индекс равен длине списка, добавляем в конец списка, иначе находим по индексу с помощью метода getNodeByIndex() элемент Next на место, которого мы должны поставить новый элемент, меняем у них связи и возвращаем ссылку на новый элемент.

Метод FunctionNode deleteNodeByIndex(int index), удаляет элемент списка по номеру и возвращает ссылку на объект удаленного элемента. Находим элемент, который будем удалять с помощью метода getNodeByIndex(), меняем связи у его соседних элементов, а сам элемент теперь ни на что не указывает, возвращаем ссылку на него.

Задание 5

Берем конструкторы из класса ArrayTabulatedFunction и теперь заполняем список с помощью метода addNodeToTail(), внутрь которого передаем объект FunctionPoint с нужными x и y. Каждый новый элемент добавляем в конец списка.

В Методах getLeftDomainBorder() и getRightDomainBorder() вызываем метод getNodeByIndex(), в него передаем индекс первого или последнего элемента, получаем узел типа FunctionNode, к нему применяем метод getPoint() для получения доступа к точке типа FunctionPoint и затем применяем метод getx() для получения доступа к значению x.

В методах getFunctionValue(double x), getPoint(int index), getPointX(int index), getPointY(int index), setPointX(int index, double x) и setPointY(int index, double y) заменяем arrayPoints[index] на getNodeByIndex(index).getPoint(), остальное оставляем так же.

В методе setPoint(int index, FunctionPoint point) в последней строке: getNodeByIndex(index).setPoint(new FunctionPoint(point)), находим по индексу точку, которую хотим заменить записываем в нее новое значение.

В методе deletePoint(int index) используем метод deleteNodeByIndex(index).

В методе addPoint(FunctionPoint point) используем метод addNodeByIndex().

Задание 6

Создаем интерфейс TabulatedFunction, содержащий объявления общих методов классов ArrayTabulatedFunction и LinkedListTabulatedFunction. В объявлении каждого из этих классов добавляем implements TabulatedFunction, то есть класс осуществляет интерфейс.

Задание 7

В классе Main я изменила класс при создании нового объекта функции. А все тесты оставила те же. Также я добавила блоки try catch для демонстрации того, как работают исключения. В этих блоках я вводила некорректные данные, чтобы выбросить исключение.

```
Конструктор 1:
(X, Y) 1.0 2.0
(X, Y) 3.25 6.5
(X, Y) 5.5 11.0
(X, Y) 7.75 15.5
(X, Y) 10.0 20.0

Конструктор 2:
(X, Y) 1.0 1.0
(X, Y) 3.25 2.0
(X, Y) 5.5 3.0
(X, Y) 7.75 5.0
(X, Y) 10.0 9.0

Левая граница первой функции(y=2x) 1.0
Правая граница первой функции(y=2x) 10.0
Значение функции(y=2x) в точке x = 1.25, входящей в интервал: 2.5
Значение функции(y=2x) в точке x = 8 входящей в интервал: 16.0
Значение функции(y=2x) в точке x = 0 не входящей в интервал: NaN
Значение функции(y=2x) в точке x = 14 не входящей в интервал: NaN
Функция, которая возвращает копию второй точки(X, Y): 3.25 6.5
Заменим вторую точку в начальной функции
(X, Y) 1.0 2.0
(X, Y) 2.0 1.0
(X, Y) 5.5 11.0
(X, Y) 7.75 15.5
(X, Y) 10.0 20.0
```

```
Заменим у третьей точки ординату
(X, Y) 1.0 2.0
(X, Y) 2.0 1.0
(X, Y) 5.5 16.0
(X, Y) 7.75 15.5
(X, Y) 10.0 20.0

Заменим у третьей точки абсциссу
(X, Y) 1.0 2.0
(X, Y) 2.0 1.0
(X, Y) 6.6 16.0
(X, Y) 7.75 15.5
(X, Y) 10.0 20.0

Удалим третью точку
(X, Y) 1.0 2.0
(X, Y) 2.0 1.0
(X, Y) 7.75 15.5
(X, Y) 10.0 20.0

Добавим 3 новых точки
(X, Y) 0.0 0.0
(X, Y) 1.0 2.0
(X, Y) 2.0 1.0
(X, Y) 4.0 8.0
(X, Y) 7.75 15.5
(X, Y) 10.0 20.0
(X, Y) 11.0 9.0
```

```
Исключения
Границы равны
Точек меньше двух
Выход за границы
Выход за границы
Выход за границы интервала соседних точек
Совпадение абсцисс
Точек меньше трех

Process finished with exit code 0
```