ЛАБОРАТОРНАЯ РАБОТА №3

Табулированные функции: реализация на массивах и связных списках

по курсу
Объектно-ориентированное программирование

Группа 6204-010302D

Студент: Е.Д.Васильев.

Преподаватель: Борисов Дмитрий Сергеевич.

Задание 1

Изучение классов исключений входящих в API Java:

Были изучены следующие классы исключений Java:

- **Exception** - базовый класс всех проверяемых исключений
- **IndexOutOfBoundsException** - недопустимый индекс(например, вне границ массива или списка)
- **ArrayIndexOutOfBoundsException** - частный случай предыдущего класса, выход за границы массива
- **IllegalArgumentException** - неверный аргумент метода
- **IllegalStateException** - непроверяемое исключение, возникает при недопустимом состоянии объекта

Задание 2

Создание пользовательских исключений

Созданы два класса исключений в пакете functions:

FunctionPointIndexOutOfBoundsException.java:

```
package functions;

public class FunctionPointIndexOutOfBoundsException extends
IndexOutOfBoundsException{

        // Конструктор без параметров
        public FunctionPointIndexOutOfBoundsException() {
            super();
        }

        // Конструктор с сообщением
        public FunctionPointIndexOutOfBoundsException(String message) {
            super(message);
        }
    }
```

Наследуется от **IndexOutOfBoundsException**, используется при обращении к несуществующему индексу точки.

InappropriateFunctionPointException.java:

```
package functions;

public class InappropriateFunctionPointException extends Exception{
    // Конструктор без параметров
    public InappropriateFunctionPointException() {
        super();
```

```
    }

    // Конструктор с сообщением
    public InappropriateFunctionPointException(String message) {
        super(message);
    }
}
```

Наследуется от **Exception**, используется при нарушении порядка точек или дублировании абсцисс.

Задание 3

Модификация TabulatedFunction

Класс TabulatedFunction реализует табулированную функцию с использованием массива.

Конструкторы:

1. **Конструктор с равномерными интервалами по X:**

```
public ArrayTabulatedFunction(double leftX, double rightX, int pointsCount) {

    if (leftX >= rightX - EPS) {
        throw new IllegalArgumentException("Left border must be less than
right border");
    }

    if (pointsCount < 2) {
        throw new IllegalArgumentException("At least two points required");
    }
```

2. **Конструктор с заданными значениями:**

```
public ArrayTabulatedFunction(double leftX, double rightX, double[] values) {

    if (leftX >= rightX - EPS) {
        throw new IllegalArgumentException("Left border must be less than
right border");
    }

    if (pointsCount < 2) {
        throw new IllegalArgumentException("At least two points required");
```

Конструкторы выбрасывают IllegalArgumentException если левая граница больше правой или счетчик точек меньше двух.

Методы:

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

Методы:

setPoint() и setPointX() выбрасывают исключение **InappropriateFunctionPointException** если координата x задаваемой точки лежит вне интервала, определяемого значениями соседних точек табулированной функции. Также addPoint() выбрасывает это же исключение, если 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

Задание 4

Реализация связного списка

Класс LinkedListTabulatedFunction реализует двусвязный циклический список с выделенной головой.

Внутренний класс FunctionNode:

```
public class LinkedListTabulatedFunction{

    private static class FunctionNode {
        FunctionPoint point;
        FunctionNode next;
        FunctionNode prev;

        FunctionNode(FunctionPoint point) {
            this.point = point;
        }
```

- **static** - не требует доступа к внешнему классу
- **private** - инкапсуляция внутри внешнего класса

Структура списка:

- Голова (head) не хранит данных, только связи
- Список циклический, даже когда пустой

Реализованные методы:

- **getNodeByIndex(int index)** - возвращает ссылку на объект элемента списка по его номеру.
- **addNodeToTail()** - добавляет новый элемент в конец списка и возвращающий ссылку на объект этого элемента.

- **addNodeByIndex(int index) -** добавляет новый элемент в указанную позицию списка и возвращающий ссылку на объект этого элемента.
- **deleteNodeByIndex(int index) -** удаляет элемент списка по номеру и возвращающий ссылку на объект удаленного элемента.

Задание 5

Реализация LinkedListTabulatedFunction

Класс реализует те же методы что и в TabulatedFunction, но со особенностями двусвязного циклического списка

Конструкторы

Конструкторы создают список через вызовы addNodeToTail(), метод добавляет новый узел в конец списка.

Методы доступа

Методы доступа используют getNodeByIndex()для доступа к элементам.

Получение границ функции

Получение границ функции реализовано через следующий и предыдущий элементы от head.

Задание 6

Создание интерфейса

Класс TabulatedFunction переименован в ArrayTabulatedFunction. Создан интерфейс TabulatedFunction, объединяющий оба класса:

```java
package functions;

public interface TabulatedFunction {

    // Возвращает количество точек
    int getPointsCount();

    // Возвращает левую границу
    double getLeftDomainBorder();

    // Возвращает правую границу
    double getRightDomainBorder();

    // Возвращает значение функции в точке x
    double getFunctionValue(double x);

    // Возвращает точку по индексу
    FunctionPoint getPoint(int index) throws
FunctionPointIndexOutOfBoundsException;

    // Заменяет точку по индексу
    void setPoint(int index, FunctionPoint point) throws
FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException;

    // Возвращает X точки по индексу
    double getPointX(int index) throws
FunctionPointIndexOutOfBoundsException;

    // Устанавливает X точки по индексу
    void setPointX(int index, double x) throws
FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException;

    // Возвращает Y точки по индексу
    double getPointY(int index) throws
FunctionPointIndexOutOfBoundsException;

    // Устанавливает Y точки по индексу
    void setPointY(int index, double y) throws
FunctionPointIndexOutOfBoundsException;

    // Удаляет точку по индексу
    void deletePoint(int index) throws
FunctionPointIndexOutOfBoundsException, IllegalStateException;

    // Добавляет новую точку (сохраняя порядок)
    void addPoint(FunctionPoint point) throws
InappropriateFunctionPointException;

}
```

Классы ArrayTabulatedFunction и LinkedListTabulatedFunction теперь реализуют этот интерфейс.

Задание 7
Тестирование
Создана функция:
functions.LinkedListTabulatedFunction
Количество точек: 5

--- Проверим работу классов с
корректными значениями ---
Исходные точки функции:
(0,00; 2,00)
(2,50; 8,25)
(5,00; 27,00)
(7,50; 58,25)
(10,00; 102,00)

Проверка getFunctionValue:
f(-5,00) = NaN
f(0,00) = 2,00
f(2,50) = 8,25
f(5,00) = 27,00
f(7,50) = 58,25
f(10,00) = NaN
f(12,00) = NaN

Изменяем значение точки (вторая
точка, Y = 100):
(0,00; 2,00)
(2,50; 100,00)
(5,00; 27,00)
(7,50; 58,25)
(10,00; 102,00)

Добавляем новую точку (5.5; 30.25):
После добавления:
(0,00; 2,00)
(2,50; 100,00)
(5,00; 27,00)
(5,50; 30,25)
(7,50; 58,25)
(10,00; 102,00)

Удаляем точку с индексом 4:
После удаления:
(0,00; 2,00)
(2,50; 100,00)
(5,00; 27,00)

(5,50; 30,25)
(10,00; 102,00)

--- Проверка работы исключений ---
УСПЕХ (IllegalArgumentException):
Left border must be less than right
border
УСПЕХ (IllegalArgumentException):
At least two points required
УСПЕХ
(FunctionPointIndexOutOfBoundsExce
ption): Index out of bounds: 99
УСПЕХ
(InappropriateFunctionPointException)
: New X breaks order of points
УСПЕХ
(InappropriateFunctionPointException)
: Point with same X already exists
УСПЕХ (IllegalStateException):
Cannot delete: at least 3 points required
--- Проверка завершена ---