**Домашнее задание 2.**

Модель.

Ниже будет представлена структура данных - “дерево” - связный ориентированный ацикличный граф.

Базовые понятия модели:

* Каждый узел дерева хранит числовое значение (может быть как положительным, так и отрицательным)
* У каждого узла может быть несколько потомков и один родитель (узел может не иметь ни одного потомка или родителя)
* Узел, у которого нет родителя называется “корнем” дерева.
* Дерево/узел могут быть как изменяемыми, так и неизменяемыми. Для узла это значит, что если он изменяемый – то ему можно поменять родителя, потомков и значение в самом узле, если он неизменяемый – то нельзя. Для дерева это значит, что если оно изменяемое – то все изменения происходят непосредственно с самим деревом, если оно неизменяемое – то при каких-либо “изменениях” создаётся изменённая копия этого дерева.
* Если дерево неизменяемое, то оно не может содержать изменяемых поддеревьев, также это относится к изменяемому дереву, в котором не могут быть неизменяемые поддеревья.
* Узел, вершина – синонимы.

Модель состоит из следующих классов/интерфейсов:

1. **interface Wrapper<T extends Number>** - обобщенный интерфейс обёртки значения типа T, хранящегося в узле дерева.

|  |  |
| --- | --- |
| Методы | Описание |
| **T getValue()** | Возвращает значение, хранящееся внутри обёртки. |

2. **interface Node<T extends Number>** - интерфейс, представляющий вершину в дереве, где T – параметризованный тип значения, хранящегося в узле, наследуется от интерфейса **Wrapper**.

|  |  |
| --- | --- |
| Методы | Описание |
| **Node<T> getParent()** | Возвращает родителя узла, если его нет, тогда **null**. |
| **Collection<Node<T>> getChildren()** | Возвращает потомков узла, если их нет – тогда пустую коллекцию. |
| **void print(int indents)** | Выводит в консоль вершину. **indents** – сколько пробелов перед вершиной. |

3. **class MutableNode<T extends Number>** - обобщенный класс изменяемого узла дерева, реализует интерфейс **Node**.

|  |  |
| --- | --- |
| Методы | Описание |
| **void setValue(T value)** | Устанавливает новое значение вершине. |
| **void setParent(MutableNode<T> parent)** | Устанавливает нового предка вершине. |
| **void setChildren(Collection<MutableNode<T>> children)** | Устанавливает новых потомков вершине. |
| **void addChild(MutableNode<T> child)** | Добавляет нового потомка вершине. |
| **void removeChild(MutableNode<T> child)** | Удаляет потомка у вершины. |

|  |  |
| --- | --- |
| Поля | Описание |
| **T value** | Значение в вершине. |
| **MutableNode<T> parent** | Родительская вершина. |
| **Collection<Node<T>> children** | Потомки вершины. |

4. **class ImmutableNode<T extends Number>** - обобщенный класс неизменяемого узла дерева, реализует интерфейс **Node.**

|  |  |
| --- | --- |
| Поля | Описание |
| **T value** | Значение в вершине. |
| **ImmutableNode<T> parent** | Родительская вершина. |
| **Collection<Node<T>> children** | Потомки вершины. |

**Подсказка**: конструктор класса **ImmutableNode** может иметь сигнатуру:

public ImmutableNode(T value,

ImmutableNode<T> parent,

Function<ImmutableNode<T>, Collection<? extends Node<T>>> childrenConstructor

)

5. **abstract class AbstractTree<T extends Number>** - обобщенный абстрактный класс дерева.

|  |  |
| --- | --- |
| Методы | Описание |
| **Node<T> getRoot()** | Возвращает корень дерева. |
| **int getSize()** | Возвращает количетсво вершин в дереве. |
| **T getSum()** | Возвращает сумму значений узлов дерева. |
| **abstract AbstractTree<T> removeSubtree(Node<T> rootSubTree)** | Возвращает дерево, из которого удалено поддерево, корень которого **rootSubTree**. (Если дерево изменяемое, то дерево меняется и возвращает инстанс самого дерева, иначе возвращает новое неизменяемое дерево) |
| **abstract AbstractTree<T> maximize(int k)** | Возвращает дерево, которое имеет максимальную сумму всех значений узлов дерева при удалении из этого дерева не более **k** поддеревьев. (Если дерево изменяемое, то дерево меняется и возвращает инстанс самого дерева, иначе возвращает новое неизменяемое дерево) |
| **abstract AbstractTree<T> maximize()** | Возвращает дерево, которое имеет максимальную сумму всех значений узлов дерева. Для этого из дерева удаляются поддеревья в том количестве, в котором это необходимо. (Если дерево изменяемое, то дерево меняется и возвращает инстанс самого дерева, иначе возвращает новое неизменяемое дерево) |

|  |  |
| --- | --- |
| Поля | Описание |
| **Node<T> root** | Корень дерева. |
| **BinaryOperator<T> adder** | Сумматор, с помощью которого будут происходить необходимые вычисления. (Он будет передаваться при создании экземпляра наследника класса **AbstractTree**) |
| **T sum** | Сумма, которую необходимо вычислить по условию. |
| **Comparator<T> comparator** | Объект для сравнивания объектов типа T |
| **T zero** | Объект «Ноль» для различия отрицательных и положительных чисел |

6. **class MutableTree<T extends Number>** - обобщенный класс изменяемого дерева, реализация абстрактного класса **AbstractTree**.

7. **class ImmutableTree<T extends Number>** - обобщенный класс неизменяемого дерева, реализация абстрактного класса **AbstractTree**.

Диаграмма классов.

Методы, которые написаны курсивом – абстрактные.

A screenshot of a cell phone

Description automatically generated

Постановка задачи.

Дано дерево с узлами, в которых хранятся числовые значения. Необходимо реализовать метод, который будет удалять поддерево по переданному узлу, а также метод, который будет максимизировать общую сумму значений узлов дерева удаляя некоторые поддеревья. Дерево может быть как изменяемым, так и неизменяемым.

Результат выполнения домашнего задания

Результат выполнения домашнего задания загружается в moodle до установленного срока (deadline: 08.12.2019, 23:00 Moscow Time) в виде файла–архива с именем **HW2\_<группа>\_<Фамилия>\_<Имя>.rar** (или **zip**).

Архив должен содержать проект в IntelliJ IDEA с исходными файлами решения задачи и разработанными Unit-тестами (с использованием JUnit5).

Кроме проекта с исходными файлами кода, необходимого для решения поставленной задачи, и тестов, использованных при этом, предоставляемый архив может – при необходимости - содержать файлы, поясняющие решение.