**Домашнее задание № 2. Mutable/Immutable Trees.**

В данном домашнем задании необходимо

* Рассмотреть ООП модель дерева.
* Рассмотреть две реализации модели.
* Завершить реализацию недостающих методов.

# Модель

Рассмотрите следующую ООП модель, реализующую структуру данных «дерево» – связный неориентированный ацикличный граф.

Базовые понятия модели:

* Каждый узел дерева хранит определенное значение.
* Узел, у которого могут быть дети (children), называется родителем (parent).
* Узел, у которого есть родитель (parent), называется ребенком (child).

Модель состоит из трех базовых интерфейсов и одного абстрактного класса:

1. **interface** IWrapper<T> – обобщенный интерфейс обертки значения типа T, хранящегося в узле дерева.

|  |  |
| --- | --- |
| Метод | Описание |
| T getObject() | Возвращает значение, хранящееся внутри обёртки. |

1. **interface** IParent<T> – обобщенный интерфейс, представляющий узел-родитель, где T – параметризованный тип значения, хранящегося в узле.

|  |  |
| --- | --- |
| Метод | Описание |
| Set<? **extends** IChild<T>> getChildren() | Возвращает множество узлов-детей, принадлежащих данному родителю. |
| Collection<? **extends** IChild<T>> getAllDescendants() | Возвращает коллекцию, состоящую из всех потомков узла, т.е. детей узла, детей детей узла и т.д. |
| **boolean** contains(T childValue) | Возвращает булево значение, сообщающее о наличии ребенка с указанным значением в узле. |
| **boolean** containsDescendants(T childValue) | Возвращает булево значение, сообщающее о наличии потомка с указанным значением, т.е. среди детей узла, детей детей узла и т.д. |

1. **interface** IChild<T> – обобщенный интерфейс, представляющий узел-ребенок, где T – параметризованный тип значения, хранящегося в узле.

|  |  |
| --- | --- |
| Метод | Описание |
| IParent<T> getParent() | Возвращает узел-родитель данного узла. |

1. **abstract class** AbstractTreeNode<T> **implements** IWrapper<T> – абстрактный обобщенный класс, представляющий узел в дереве, где T – параметризованный тип значения, хранящегося в узле.

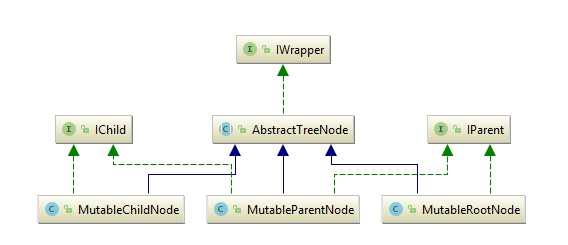
# Реализация

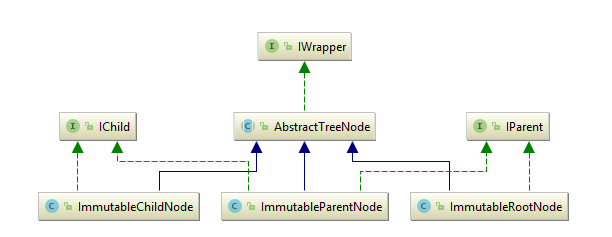
Существует две реализации данной модели: Mutable – это реализация, в которой структура дерева может меняться (ссылки на родителя и детей могут быть изменены), и Immutable, где изменения структуры дерева запрещены.

Каждая реализация (Mutable/Immutable) содержит по три класса:

* MutableChildNode/ImmutableChildNode – узел-ребенок дерева. Это листовой узел дерева, который не имеет своих детей.
* MutableParentNode/ImmutableParentNode – узел-родитель дерева. Это внутренний узел дерева, который имеет как родителя, так и детей.
* MutableRootNode/ImmutableRootNode – корневой узел дерева. Он имеет детей, но не имеет родителя.

Ниже приведены диаграммы иерархии классов.



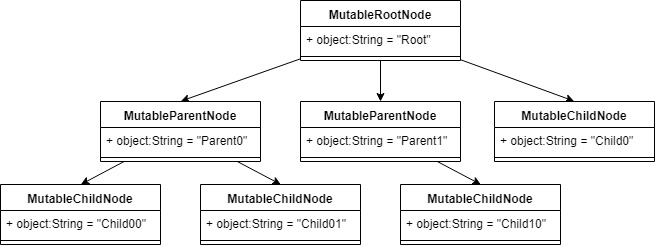


# Завершение реализации

Представленные выше реализации не завершены. Ваша задача – завершить. Для этого необходимо реализовать функции, описанные ниже. В проекте они отмечены с помощью «TODO». Чтобы узнать, как найти все «TODO» в проекте, ознакомьтесь с информацией по ссылке:

<https://www.jetbrains.com/help/idea/todo-tool-window.html>

1. Реализовать метод toStringForm во всех классах, представляющих реализацию узла дерева. Данный метод выводит строковое представление узла в следующем рекурсивно заданном формате: сначала выводится информация об узле в формате <**SimpleRootClassName**>(<**ObjectToString**>), где **SimpleRootClassName** имя класса узла без указания пакета, **ObjectToString** – строковое представление значения узла; затем, в таком же формате, рекурсивно выводится информация о детях узла (если они есть) с добавлением лидирующих 4-х пробелов. Например, для структуры дерева, схематически указанной ниже, будет сгенерировано следующее строковое представление:



MutableRootNode(Root)

MutableParentNode(Parent0)

MutableChildNode(Child00)

MutableChildNode(Child01)

MutableParentNode(Parent1)

MutableChildNode(Child10)

MutableChildNode(Child0)

1. В классе TreeImporter реализовать статический метод importMutableTree, который является обратным к описанному выше – из строкового представления генерирует объектное. В качестве значения <**ObjectToString**> могут выступать: cтрока, 32-битное знаковое целочисленное число, 64-битное число с плавающей точкой. <**SimpleRootClassName**> может принимать значения: MutableRootNode, MutableParentNode и MutableChildNode.
2. Реализовать метод getAllDescendants, объявленный в IParent, во всех классах, которые реализуют его.
3. Реализовать метод contains, объявленный в IParent, во всех классах, которые реализуют его.
4. Реализовать метод containsDescendants, во всех классах, которые реализуют его.
5. Реализовать метод removeChildByValue, объявленный в MutableParentNode и MutableChildNode, которые удаляет из *узла* первого встретившегося ребенка с указанным значением.
6. Реализовать метод removeDescendantsByValue, объявленный в MutableParentNode и MutableChildNode, которые удаляет из *дерева* первого встретившегося ребенка при обходе дерева в ширину. Обход дерева в ширину происходит путём последовательного просмотра отдельных уровней дерева, начиная с заданного узла.
7. Реализовать конструктор класса ImmutableRootNode, который принимает в качестве аргумента MutableRootNode. Структуры исходного и результирующего деревьев при этом – одинаковы, содержат соответствующие одинаковые значения в узлах, но отличаются возможностью (mutable) и невозможностью (immutable) изменения своей структуры.
8. Реализовать конструктор класса MutableRootNode, который принимает в качестве аргумента ImmutableRootNode. Структуры исходного и результирующего деревьев при этом – одинаковы, содержат соответствующие одинаковые значения в узлах, но отличаются возможностью (mutable) и невозможностью (immutable) изменения своей структуры.
9. Реализовать метод addChild, объявленный в MutableParentNode и MutableRootNode, который добавляет ребенка в узел.
10. Для реализованных методов написать JUnit тесты. *Реализованный метод без тестов, направленных на этот метод, не считается реализованным*.

# Оценивание

1. Верно выполненные пункты 1-7, 11 гарантируют оценку 7.
2. Верно выполненные задания 8-10 с соблюдением 11, добавляют по одному баллу к итоговой оценке. Задания 8-10 не рассматриваются при невыполненных заданиях 1-7.

# Результат выполнения задания.

Результатом выполнения задания является архив с именем **HW2\_<фамилия>\_<группа>**, содержащий IntelliJ IDEA проект с выполненным заданием. Архив должен быть загружен в Moodle.

# Приложение.

К заданию прилагается архив с IntelliJ IDEA проектом, в котором есть соответствующие заданию заготовки для его выполнения (**HW2**). Там, где в исходных кодах есть комментарии, указывающие на «временную заплату» (//tmp stub), указания //TODO и/или в коде выдается RuntimeException “not implemented yet”, необходимо поместить код соотвествующей реализации.