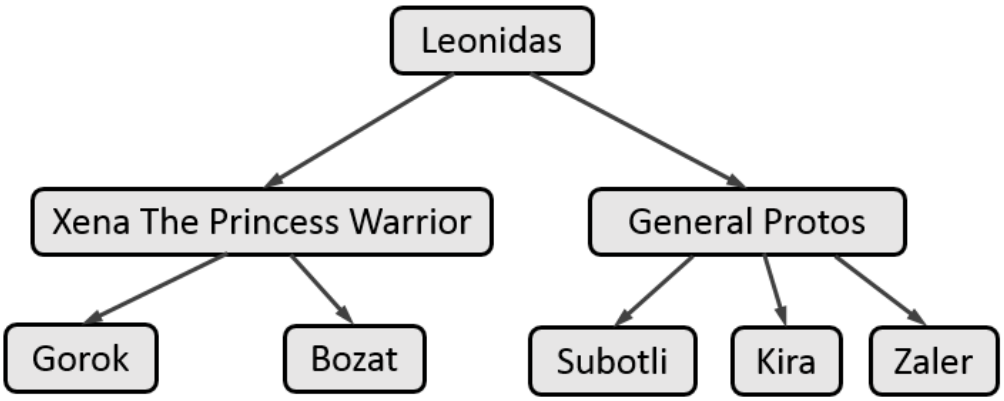
Hierarchy – Data Structures Exam

A **Hierarchy** is a data structure that stores elements in a hierarchical order. See the example:



It supports the following operations:

- **`Add(element, child)`** - adds **`child`** to the hierarchy as a child of **`element`**.
  - Throws an exception if **`element`** does not exist in the hierarchy.
  - Throws an exception if **`child`** already exists (duplicates are not allowed).
- **`Remove(element)`** - removes the element from the hierarchy.
  - If it has children, they become children of the element's parent.
  - If element is root node, throws an exception.
- **`Count`** - returns the count of all elements in the hierarchy
- **`Contains(element)`** - determines whether the element is present in the hierarchy.
- **`Get-Parent(element)`** - returns the parent of the element.
  - Throws an exception if **`element`** does not exist in the hierarchy.
  - Returns the **dafault value for the type** (e.g. **`int`** → **`0`**, **`string`** → **`null`**, etc.) if element has no parent.
- **`Get-Children(element)`** - returns a collection of all direct children of the element in order of their addition.
  - Throws an exception if **`element`** does not exist in the hierarchy.
- **`Get-Common-Elements(Hierarchy other)`** - returns a collection of all elements that are present in both hierarchies (order does not matter).
- **`For-Each()`** - enumerates over all elements in the hierarchy by levels.
  - In the image above, the elements would be enumerated as such - **Leonidas** -> **Xena the Princess Warrior** -> **General Protos** -> **Gorok** -> **Bozat** -> **Subotli** -> **Kira** -> **Zaler**.

Input and Output

You are given a **Visual Studio C# project skeleton** (unfinished project) / **IntelliJ Java project** holding the interface **`IHierarchy`**, the unfinished class **`Hierarchy`** and **tests** covering its **functionality** and its **performance**.

Your task is to **finish this class** to make the tests run correctly.

- You are **not allowed to change the tests**.
- You are **not allowed to change the interface**.

Interface IHierarchy

The interface **`IHierarchy`** in C# looks like the code below:

```
public interface IHierarchy<T> : IEnumerable<T>
{
    int Count { get; }
    void Add(T element, T child);
    void Remove(T element);
    IEnumerable<T> GetChildren(T element);
    T GetParent(T element);
    bool Contains(T element);
    IEnumerable<T> GetCommonElements(IHierarchy<T> other);
}
```

The interface **IHierarchy** in Java looks like the code below:

```
public interface IHierarchy<T> extends Iterable<T> {
    int getCount();
    void add(T element, T child);
    void remove(T element);
    Iterable<T> getChildren(T element);
    T getParent(T element);
    boolean contains(T element);
    Iterable<T> getCommonElements(IHierarchy<T> other);
}
```

Submissions

Submit an archive (.zip) of the source code + external libraries.

Scoring

Each implemented method brings you a specific amount of points, some of the points are awarded for correct behavior, others for performance. You need to cover all tests in a given group in order to receive points. Bellow is a breakdown of all points by methods:

Method	Correct Behaviour	Performance	Total
Add	3	8	11
Remove	4	12	16
Contains	3	10	13
Get Parent	3	11	14
Get Children	3	11	14
Get Common Elements	4	12	16
For Each	4	12	16
Overall:	24	76	100