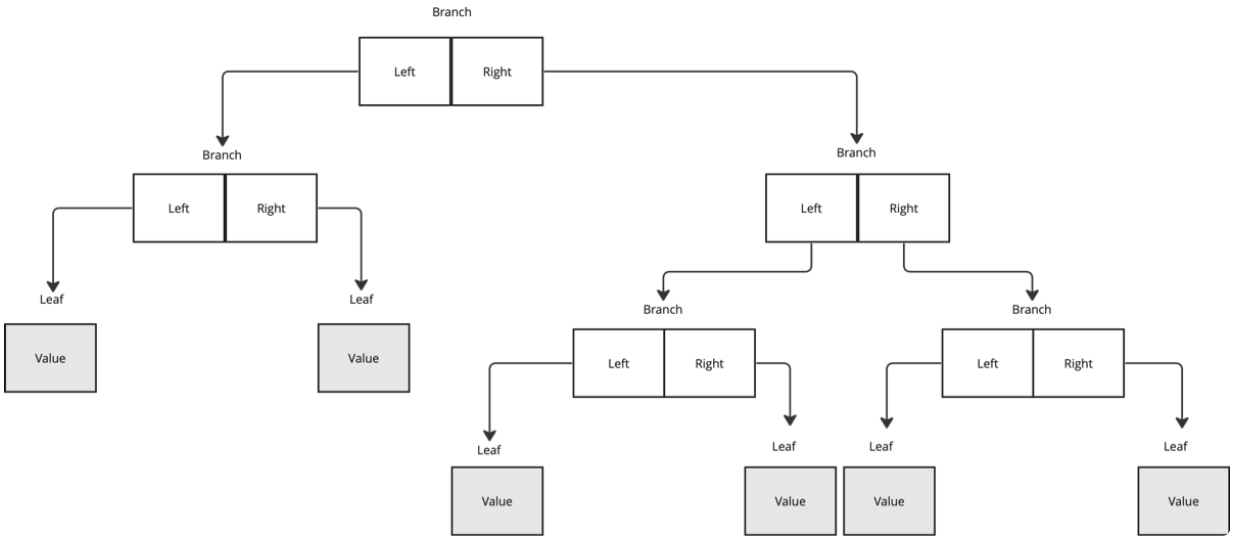
Scala - Exercises 4 - Abstract Data Types

The purpose of this exercise is to practice creating and working with your own abstract data type. You can use the MyLinkedList example in the lectures as an example of doing this

Trees

The aim here is to define an ADT for a binary tree data structure, pictorially this can be represented as:



1. By defining a sealed trait and appropriate case classes, write an ADT representing the above binary tree data structure. Your tree should support any type.
2. Write a function size that counts the number of nodes (i.e. leaves and branches) in the tree
3. Write a function reduce, that takes in a function and reduces your tree to a single value using the function
4. Using your reduce function above, write a function that returns the maximum of a Tree of Ints
5. Write a function depth that returns the maximum path length from the root to any leaf. The function definition is:

   ```
   def depth: Int
   ```

6. Write a function map that applies a given function to each element in the tree

7. Write a function fold that abstracts over the similarities in the above. The function definition should be:

```
def fold[B](f: A => B, g:(B,B) => B): B = ???
```

8. Using the fold function above, rewrite your size, reduce, depth and map functions using fold.