Prepared by Linan Qiu <lq2137@columbia.edu>

Simple Trees

When we say tree in CS, don't think of real trees (that are bigger at the bottom and smaller at the top.) Instead, think of family trees that get bigger at the bottom. It is simply elements that have parents, children, grandparents, grandchildren etc.

In fact, a tree, without any further specification, is pretty simple (and useless).

Let's start by modeling amoebas in the simplest form: an amoeba can split into two, which can continue to split into two and so on and so forth.

Nodes

Then, a simple amoeba can be a node with two references to other nodes: its children. For the sake of simplicity, let's call them `left` and `right`. Amoebas carry stuff in their cells, so we make amoebas generic.

```
// AwsmNode.java

public class AwsmNode<T> {
  public T data;
  public AwsmNode<T> left;
  public AwsmNode<T> right;

  public AwsmNode(T data, AwsmNode<T> left, AwsmNode<T> right) {
    this.data = data;
    this.left = left;
    this.right = right;
  }

  public String toString() {
    return data.toString();
  }
}
```

To make testing easier, we add a `toString()` method as well so that we can call `System.out.println` on a `AwsmNode` directly.

Creating Trees using Nodes

Can we use this to construct a family of Amoebas? Sure. Let's say an amoeba split into children, who each split into grandchildren. Let's assume the grandchildren have yet to split. Then, we can do this:

```
// Test.java


public class Test {
  public static void main(String[] args) {
    // let's make grandchildren!
    AwsmNode<String> grandChild1 = new AwsmNode<>("g1", null, null);
    AwsmNode<String> grandChild2 = new AwsmNode<>("g2", null, null);
    AwsmNode<String> grandChild3 = new AwsmNode<>("g3", null, null);
    AwsmNode<String> grandChild4 = new AwsmNode<>("g4", null, null);

    // let's make children, and assign them their children
    AwsmNode<String> child1 = new AwsmNode<>("c1", grandChild1, grandChild2);
    AwsmNode<String> child2 = new AwsmNode<>("c2", grandChild3, grandChild4);

    // let's make the parent, and assign it its children
    AwsmNode<String> parent = new AwsmNode<>("p1", child1, child2);
  }
}
```

This should be pretty straightforward. So let's move on to something fun in the next chapter. We will be printing the family tree.