Assignment One: The last datatype on earth

A villain has come to Java city and they have wiped out all the integers! They also wiped out all the value types (`boolean`, `float`, `double`, `byte`, etc). They've promised to come after the java standard library and any reference types next.

1 Introduction

Your task is to start rebuilding civilisation by creating a `Church` class which can work just like the integers we lost. It should have the following functionality:

- The integer zero is represented as `null`.
- You can make a `Church` representing one
- You can make a `Church` from any other Church. It will represent the number one more than the `Church` it was built from.
- You can perform simple arithmetic (add, subtract, and multiply) with `Church` objects.

2 Starting Point

Your trusty side-kick Graal has created a class with the right method signatures to get you started. Graal also creates a test file which tests the basics of the `Church` data type/class. You will find these in the `Graal.zip` file on iLearn.

3 Your task

Complete the `Church` class so that it passes all the tests written by Graal and matches the specification given below. To avoid being vaporised by the villain *you must not*:

- Import any libraries
- Use any integers, doubles, floats, boolean, or any other types at all.
- Use `String`s.
- Use any other classes or types at all.

You have access to all the control flow, the `Church` class, and nulls.

4 Specifications

Rules

- Minimum JDK version: JDK 8
- You must attempt the methods in `src/Church.java`
- You must pass the tests provided in `src/tests/ChurchTest.java`
- You should not hard-code your implementations to these tests cases, additional test cases will be used for marking that are trivial for a valid implementation.
- You may add your own tests to these files, noting that test files are not submitted

The `Church` class must implement the following methods (Graal has created the method signatures for you).

constructors There should be two constructors. One takes no arguments and sets up the object to represent the number one. The other takes a *smaller* `Church` and represents the number one larger than the smaller `Church`.

equals Graal has created an equals to override the built-in one and work with tests, but it needs you to complete the more specific `public boolean spec_equals(Church o)` version to work properly. I can tell you that basically none of your tests will pass without this method so make it your priority. Graal snuck the last few `boolean`s they could scrounge up just for this purpose. We have chosen to risk it with these in our test file. We will destroy the test file if it gets detected.

incr Returns a `Church` which represents the number one larger than `this`.

decr returns a `Church` which represents the number one smaller than `this`. If the number already represented 1, then it goes to zero (`null`). You can't decrement zero, attempting to do so should give a null pointer exception.

plus A method which takes one argument, the `other Church` which is added to `this Church`. returns a `Church` which represents the number resulting from the addition.

minus A method which takes one argument - the `other Church` - which is subtracted from `this Church`. Returns a `Church` which represents the number resulting from the subtraction. If the second number is greater than or equal the first (i.e. the result should be zero or negative), the result should be zero (i.e. `null`).

mult A method which takes one argument, the `other Church` which is multiplied to `this Church`. Returns a `Church` which represents the number resulting from the multiplication.

Submission requirements

- You must submit ONLY your `Church.java` file to the ilearn assignment 1 submission box.
- Do not rename the submitted file, it must be named `Church.java` as provided in the starting template.
- Your `Church.java` file must compile. If you see any compilation errors prior to submission, you must fix them. These will show:

- You must NOT make use of any Java standard library packages, classes, or objects. You cannot create equivalents either.
- You must not be at risk of vapourisation by the villain.

Recommendation: Once you have submitted, redownload your submitted file and ensure it is the file you intended to submit. It is worth double-checking by re-running it against the tests and checking if the result is as you expect

5 Hints

This assignment tests your understanding of, and ability to code, *recursive data types* like the linked list we looked at in lectures. Your solution *will need to be a recursive data type very much like that linked list*. You may need to use some imagination to see the similarity, but when you see it, all will be revealed[1].

5.1 Still don't get it?

There will be a discussion in the lecture in week 3. You should watch (and rewatch) that discussion. You should discuss the discussion with your classmates and your tutor. You should write out your understanding of the discussion. If you are still confused after going through these steps, email Matt with a copy of what you wrote up and he will organise a chat to help.

5.2 Tips

You will find the work easier if you write a `toString()` method for your `Church` class. However, this involves using `String`s which is not allowed! *If* you want to go down this route to help you during your development, be sure to triple check you have removed any sign of `String`s before you submit your code or your grades could get vapourised.

6 Bonus

Get extra chances at the lecture prize draw by posting your thoughts regarding why Graal has called the class `Church` to the unit Forum. There are also extra chances up for grabs for coming up with a good name for our villain or for ourselves (the hero).

7 Grading

Your submitted file will be run against a test program consisting of the tests Graal delivered to you and an additional 6 tests. Passing each test is worth marks according to the following table. Note, you will not be given the test script for the additional tests but can guess what they do from the names

[1] The ability to find *isomorphisms* is your superpower, use it to save Java city

test name	**provided by Graal**	**value**
testEqualsDirectly	✓	5
testEqualsIndirectly		5
testIncr	✓	5
testIncrTwo		5
testDecr	✓	5
testDecrTwo		5
testPlus	✓	5
testPlusTwo		5
testMinus	✓	5
testMinusTwo		5
testMult	✓	5
testMultViaEverything		10
total		65

The following penalties will apply on top of that rubric:

error	**penalty**
Use of value type	-30
Use of String	-30
Failure to compile	-65
Importing anything	-20
Risk of vapourisation	-30
Submitting a solution generated by someone or something else	-65