3 Tutorial 03

https://nw-syd-gitlab.cseunsw.tech/COMP2511/23T2/content/-/tree/master/tutorials/tute03

A. Code Review & Questions

In your project groups, answer the following questions.

1. Can you override a static method?
2. What is output by executing `A.f()` in the following?

```java
public class A {
    public static void f() {
        C c = new C();
        c.speak();
        B b = c;
        b.speak();
        b = new B();
        b.speak();
        c.speak();
    }
}


public class B {
    public void speak() {
        System.out.println("moo");
    }
}


public class C extends B {
    public void speak() {
        System.out.println("quack");
    }
}
```

3. What is output by executing `A.f()` in the following?

```java
public class A {
    public static void f() {
        B b1 = new B();
        B b2 = new B();
        b1.incX();
        b2.incY();
        System.out.println(b1.getX() + " " + b1.getY());
        System.out.println(b2.getX() + " " + b2.getY());
    }
}

public class B {
    private int x;
    private static int y;

```

```
    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public void incX() {
        x++;
    }

    public void incY() {
        y++;
    }
}
```

B. Domain Modelling

In this problem, we are going to create an Object-Oriented domain model for a system with the following requirements.

With success in student projects like Sunswift and Redback, UNSW have decided that they would like to build a system that can show all of the student-built and other cars that they have in order to showcase to prospective students interested in STEM and attract students from other universities. They have asked you, as a designer to produce a model for what this system will look like.

Requirements

A Car has one or more engines and a producer. The producer is a manufacturing company who has a brand name. Engines are produced by a manufacturer and have a speed. There are only two types of engines within UNSW's cars:

- **Thermal Engines**, which have a default max speed of 114, although they can be produced with a different max speed, and the max speed can change to any value between 100 and 250.
- **Electrical Engines**, which have a default max speed of 180. This is the speed at which they are produced, and the max speed can change to any value that is divisible by 6.

Cars are able to drive to a particular location `x`, `y`.

Since UNSW is a world-leader in technology innovation, they want you to be able to model the behaviour of Time Travelling for *any* vehicle, and to model a time travelling car. A vehicle that travels in time *stays in the same location* but travels to a `LocalDateTime`.

Create a UML diagram which models the domain.

During the lab, you will build on this UML diagram to incorporate further requirements.

C. Wondrous

The **Wondrous Sequence** is generated by the simple rule:

- If the current term is even, the next term is half the current term.
- If the current term is odd, the next term is three times the current term, plus 1.

For example, the sequence generated by starting with `3` is:

```
3 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1
```

If the starting term is `1`, then an empty list is returned.

Inside `src/wondrous/Wondrous.java` there is an implementation of this algorithm. Inside `src/wondrous/test/WondrousTest.java` there is a single test for the function. The test currently fails.

Part 1 - IDE Programming

Explore the IDE tools built into VSCode by:

1. Put a breakpoint on line 13 and run the tests in Debug Mode.
2. Briefly discuss different features of Debug Mode:
    - The variables section
    - The debug console
    - The 'watch' section
    - The call stack
    - Debug control
3. Use the debug tools and the given algorithm to determine why the test is failing, and fix the bug.

Part 2 - Writing Tests with JUnit

There is a further bug in the function not caught by the given unit test. Find the other bug, and write a corresponding unit test inside `WondrousTest`.

You can learn more about JUnit here.

Part 3 - Exceptional Conditions

Modify the method such that if a `start` is less than 1, an `IllegalArgumentException` is thrown. Write a corresponding test for this inside `WondrousTest`.

In many cases when we throw an exception we need to update the method signature and existing tests but here we don't - why is this?