Software Design (F28SD2):
Exercises in Dynamic Analysis 2

Andrew Ireland
School of Mathematical and Computer Sciences
Heriot-Watt University
Edinburgh

Exercise 1

Consider the following traffic signal control program. Note that the source code can be found in `http://www.macs.hw.ac.uk/~air/swdesign/code/Controller.java`.

```
// Controller.java:
//
// A simple traffic signal control program --
// an exercise in class invariant construction.
//
// A. Ireland  Software Design
//

import java.io.*;

class Colour {
    public static final Colour RED   = new Colour();
    public static final Colour GREEN = new Colour();
    private Colour() {}
}

class Signals
{
    static PrintWriter screen = new PrintWriter (System.out, true);
    private int cycle;  // Time
    Colour north;       // Colour signal displayed to north bound traffic
    Colour west;        // Colour signal displayed to west bound traffic

    // Initializes internal cycle time and the colour signals
    // for north and south bound traffic
    public Signals()
    { cycle = 0; north = Colour.RED; west = Colour.GREEN; }

    // Conversion between symbol signal colours and string form
    String convert(Colour sig)
    { if (sig == Colour.RED) return "Red  ";
                        else return "Green";
    }

    // Display colours of both signals at time t
    void display_signals(int t, Colour n, Colour w)
    { screen.println("cycle  = " + t +
                     " north = " + convert(n) +
                     " west  = " + convert(w));
    }
```

```
    // Allow north bound traffic to proceed
    void proceed_north()
    { if (north == Colour.RED){
          west = north; north = Colour.GREEN;
          display_signals(cycle, north, west);
          cycle++;
        }
    }
    // Allow west bound traffic to proceed
      void proceed_west()
      { if (north == Colour.GREEN){
            west  = Colour.GREEN; north = west;
            display_signals(cycle, north, west);
            cycle++;
        }
      }
}

class Controller{

    public static void main(String[] args ) throws IOException
    { Signals junction = new Signals();
      while (true){ junction.proceed_north(); junction.proceed_west(); }
    }
}
```

What crucial safety property would you expect of the traffic signal control program? Formulate your safety property as a class invariant for `Signals`, *i.e.* using the Java `assert` feature. Test the actual code using your class invariant. Do you trust the invariant? Do you trust the code?

Exercise 2

Consider JUnit testing of the `Book` class given in lecture. Your task here is to replicate the unit testing described in the lecture notes and add your own test cases. The `Book` and associated `TestBook` classes can be found in:

`http://www.macs.hw.ac.uk/~air/swdesign/code/mybook/`

Make a copy of these files, before compiling them. In the case of `TestBook.java` you will need to tell the compiler where to find the JUnit classes, *i.e.* `junit-4.8.2.jar`. To achieve this execute the following command line in the `mybook` directory[1]:

```
javac -cp .:/usr/java/bluej/lib/junit-4.8.2.jar  BookTest.java
```

Finally, to run the unit tests given in lecture execute the following command line:

```
java -cp .:/usr/java/bluej/lib/junit-4.8.2.jar junit.textui.TestRunner BookTest
```

Note that `TestRunner` is the java program that runs the tests defined within `BookTest.java`. Congratulations you are now a JUnit tester! Next fix the bug in `Book.java` and retest. Finally experiment by adding your own tests to `TestBook.java` – don't forget to re-compile.

[1] Note that "cp" is the class path flag. Ideally you should add `/usr/java/bluej/lib/junit-4.8.2.jar` to your `CLASSPATH`.