Software Engineering -Methods **Assessed Coursework 1**
Estimated Time: 1.5 hrs **By: 11th November 2002**

Part 1 – State Pattern

(a)

Using the State Design Pattern, rewrite the **Engine** class in the listing below so that the conditional statement (if .. else..) is not required.

(b)

Using the singleton pattern ensure that only one instance of each **State** sub-class is required.

Part 2 – Observer Pattern

Instances of class **ObserverField** are used to monitor the state of the engine. Using the Observer Pattern complete the implementation of the Engine class so that it notifies instances of **ObserverField** with state changes.

To Do

Download the software zip file associated with this coursework from the course web page.

Modify the file Engine.java (listed below) with the above modifications.

Check that it works by compiling all the files and running StateApplication.class – which contains the **main** method.

Hand-in

Listing of updated Engine.java suitably annotated with your name, usercode, and year group (i.e. beng3, meng3, ise3, mac3).

```
package engine;

import java.util.*;
import javax.swing.JTextField;

interface State {
  String printState();
}

class Engine {

// replace this for part 1
  boolean started = false;

  void setState(boolean b) {started = b;}

  String printState() {
    if (started)
      return "Started";
    else
      return "Stopped";
  }

  //interface with GUI
  void start() {setState(true);}
  void stop() {setState (false);}
//up to here

  public void attach(Observer o) {
    //implement for part 2
  }


 }

/*
*  used for part 2 of tutorial 5
*/

interface Observer {
  void update(State s);
}

class ObserverField implements Observer {

    JTextField field;

    ObserverField(JTextField f) {field = f;}

    public void update(State s) {
       field.setText(s.printState());
    }
}
```