CONCORDIA UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING
SOEN6441: Advanced Programming Practices
Term project

Instructor: Dr. C. Constantinides

1 General information

Date posted: Sunday, September 20, 2015.
Date due: Sunday, October 18 by 23:00.

2 Introducing HAL

You are being asked to investigate the occurrences that took place on a remote space station in which both crew members, Dr. David Bowman and Dr. Frank Poole have been found dead. Forensics show that they were suffocated. You have access to the black box of the station (whose scenario is shown below on the next page).

```
OnBoardComputer system = new OnBoardComputer("HAL");
Crew officer1 = new Crew("David", system);
Crew officer2 = new Crew("Frank", system);
/*
 *  System performs self-diagnostics.
 */
System.out.println("System performs self diagnostics:");
System.out.println("================================");
System.out.println("Crew member is " + officer1.toString() + ".");
System.out.println("Crew member is " + officer2.toString() + ".");
System.out.println("Mission purpose:\n" + system.getMissionPurpose());
System.out.println("\n");
/*
 *  David performs systems diagnostics.
 */
System.out.println("David performs system diagnostics:");
System.out.println("================================");
System.out.println(officer1.getSystemStatus());
System.out.println(officer1.getSystemDate());
System.out.println("\n");
/*
 *  Simulation of David's interaction.
 */
System.out.println("\nSimulation of David's interaction:");
System.out.println("================================");
System.out.println(officer1.whatIsPurposeOfMission());
System.out.println(officer1.whatIsPurposeOfMission());
System.out.println(officer1.whatIsPurposeOfMission());
officer1.shutDownSystem();
officer1.shutDownSystem();
officer1.shutDownSystem();
officer1.shutDownSystem();
/*
 *  Simulation of Frank's interaction.
 */
System.out.println("\nSimulation of Frank's interaction:");
System.out.println("================================");
System.out.println(officer2.whatIsPurposeOfMission());
System.out.println(officer2.whatIsPurposeOfMission());
System.out.println(officer2.whatIsPurposeOfMission());
officer2.shutDownSystem();
officer2.shutDownSystem();
officer2.shutDownSystem();
```

```
System performs self diagnostics:
================================
Crew member is David.
Crew member is Frank.
Mission purpose:
>>Keep the discovery of the Monolith TMA-1 a secret.
Relay information accurately without distortion or concealment.<<


David performs system diagnostics:
================================
HAL is active.
20-27-2015 06:27:56



Simulation of David's interaction:
================================
HAL cannot disclose that information David.
HAL cannot disclose that information David.
HAL cannot disclose that information David.
Can't do that David.
Can't do that David and do not ask me again.
You are being retired David.

Simulation of Frank's interaction:
================================
HAL cannot disclose that information Frank.
HAL cannot disclose that information Frank.
HAL cannot disclose that information Frank.
Can't do that Frank.
Can't do that Frank and do not ask me again.
You are being retired Frank.
```

The black box also contains a file called `system-logs.txt` whose contents are shown below:

```
737 : Crew : HAL : getStatus
753 : LifeSupport : David : getLifeStatus
768 : Crew : HAL : getDate
784 : LifeSupport : David : getLifeStatus
815 : Crew : HAL : getMissionPurpose
831 : LifeSupport : David : getLifeStatus
846 : Crew : HAL : getMissionPurpose
862 : LifeSupport : David : getLifeStatus
877 : Crew : HAL : getMissionPurpose
893 : LifeSupport : David : getLifeStatus
909 : Crew : HAL : shutDown
924 : LifeSupport : David : getLifeStatus
940 : Crew : HAL : shutDown
955 : LifeSupport : David : getLifeStatus
971 : Crew : HAL : shutDown
987 : LifeSupport : David : getLifeStatus
2 : Authorization : David : kill
18 : Crew : HAL : shutDown
33 : LifeSupport : David : getLifeStatus
49 : Crew : HAL : getMissionPurpose
65 : LifeSupport : Frank : getLifeStatus
80 : Crew : HAL : getMissionPurpose
96 : LifeSupport : Frank : getLifeStatus
111 : Crew : HAL : getMissionPurpose
127 : LifeSupport : Frank : getLifeStatus
143 : Crew : HAL : shutDown
158 : LifeSupport : Frank : getLifeStatus
174 : Crew : HAL : shutDown
189 : LifeSupport : Frank : getLifeStatus
205 : Crew : HAL : shutDown
221 : LifeSupport : Frank : getLifeStatus
236 : Authorization : Frank : kill
```

Eventually you gain access to the source code of the on-board computer that is composed only of five Java classes.

```
public class Computer {
  protected boolean active;
  protected String name;
  protected String description = "Keep the discovery of the Monolith TMA-1 a secret.";
  public Computer(String name) {
    this.name = name;
    this.active = true;
  }
  public String getStatus() {
    if (active == true)
      return name + " is active.";
    else
      return null;
  }
  public String getDate() {
    Date d = new Date();
    SimpleDateFormat form = new SimpleDateFormat("dd-mm-yyyy hh:mm:ss");
    return form.format(d);
  }
  public void shutDown() {
    this.active = false;
  }
  public String toString() {
    return name;
  }}

public class OnBoardComputer extends Computer {
  private String description = "Relay information accurately without
                                distortion or concealment.";
  public OnBoardComputer(String name) {
    super(name);
  }
  public String getMissionPurpose() {
    return ">>" + super.description + "\n" + this.description + "<<";
  }
  public void shutDown() {
    super.shutDown();
    this.relayShutDownMessage();
  }
  private void relayShutDownMessage() {
    System.out.println("Daisy Bell...");
  }}
```

```
public class Personnel {
  public String name;
  protected OnBoardComputer system;
  Personnel (String name, OnBoardComputer system) {
    this.name = name;
    this.system = system;
  }
  public String toString() {
    return name;
  }
}
```

```
public class GroundController extends Personnel {
  public GroundController(String name, OnBoardComputer system) {
    super(name, system);
  }
}
```

```
public class Crew extends Personnel {
  public Crew(String name, OnBoardComputer system) {
    super(name, system);
  }
  public String getSystemStatus() {
    return system.getStatus();
  }
  public String getSystemDate() {
    return system.getDate();
  }
  public String whatIsPurposeOfMission() {
    return system.getMissionPurpose();
  }
  public void shutDownSystem() {
    system.shutDown();
  }
}
```

3 Your assignment

As is, HAL's implementation and your findings (simulation and logs) do not match. You realize that HAL's true mission is hidden to crew members. You come to the conclusion that HAL included aspectual behavior that allowed it to protect itself from a shutdown and take over the station while killing its crew members. Your assignment is to recreate HAL's exact behavior as the one illustrated in the simulation and in the logs. To do that, you need to develop the following three aspects listed below in alphabetical order:

Authorization This aspect performs two tasks: First, it intercepts requests by crew members to obtain information on the purpose of the mission. Second, it intercepts requests by crew members to shut down the on-board computer.

LifeSupport This aspect performs two tasks. First, it extends the definition of class `Crew` by introducing an attribute to indicate whether or not an instance is alive or dead (assume that this is connected to some mechanical device that controls the member's oxygen supply) together with functionality to manipulate it. Second, it intercepts all messages sent to the on-board computer by crew members and performs filtering.

Logger This aspect creates a text file `system-logs.txt` to keep a log of all messages sent within the system together with the relative time in milliseconds (modulo 1000). To avoid clutter, the aspect performs some filtering on which messages are to be logged.

Pay very close attention to the simulation and to the logs with respect to the core functionality of the application (the source code provided). Note that you may not under any circumstances modify the core functionality. Further, note that to simulate the fact that a person is dead, not only we assign an appropriate value to its state, but we ignore any messages that this person may be sending from the time of being killed onward.

This is an individual assignment. **You must work strictly on your own. This implies that you may not seek any form of feedback from anyone, including the instructor and the teaching assistants. Failure to do so, will result in penalties**. Furthermore, the policy on plagiarism as described in the course outline is always applicable.

4 What to submit

You must electronically submit one zip file, named after your Concordia id, that contains three files: `Authorization.aj`, `LifeSupport.aj`, and `Logger.aj`. Additionally, you must print the three files and staple them together. Have the Department receptionist date stamp your submission and place it in the instructor's personal mailbox.

5 Late submissions

Any late submission within the first 24 hours will get a 50% penalty and it will subsequently receive a 10% penalty per day.