ADVANCED JAVA PROGRAMMING AJP-P4-2012-2013

INTRODUCTION

This exercise will test your knowledge of the *chain of responsibility* and *template method* design patterns.

SET-UP

Import the *Netbeans* project folder named 'AJP-P4-2012-2013-STUDENT'. Open it up. You have been given a set of unit tests. Initially, there will be LOTS of syntax errors in these unit tests. This is expected. The errors will disappear as you write the required classes.

INSTRUCTIONS FOR BRONZE TASK

In the *src* folder of your new project, in the package named *uk.ac.tees.bronze.username,* create a file called *Malfunction.java*. This class represents a malfunction on a spaceship. Your class should satisfy the following criteria.

- The *Malfunction* class should have a *severity* instance variable that records the severity of the problem.
- The different levels of problem on a spaceship are as follows – TRIVIAL, LOW, MEDIUM and HIGH. These should be defined in a separate *enum* called *Severity.java.*
- The *Malfunction* class should also have a *description* instance variable that stores a textual description of the problem. This variable should be of type *String*.
- The constructor for the *Malfunction* class should be *Malfunction(Severity severity, String description)*. Null and empty descriptions should be caught and converted into the default description - “No description available. Probably serious”.
- You should write accessor methods for the *description* and *severity* instance variables.

Next, create a new interface called MalfunctionHandler.java. This interface should define the following two methods:

```
public void processMalfunction(Malfunction malfunction);

public void setNextHandler(MalfunctionHandler next);
```

Now implement this interface in 4 concrete classes, named as follows:

- *SpaceMonkey.java*
- *ServiceRobot.java*
- *Engineer.java*
- *Captain.java*

Each of the classes above should have a constructor that takes a *Severity* parameter. This parameter is the *competence* level of the crew member. A crew member can handle problems up to (and including) their own competence level. For example, *new Engineer(Severity.MEDIUM)* creates an *Engineer* object that can deal with anything up to (and including) medium severity problems.

Inside each of these 4 classes, the *processMalfunction(Malfunction)* method should function like this:

```
IF severity of malfunction is equal to or below my level of my competence
     WRITE the problem description to the log file
     WRITE “---> XXX assigned to problem.\n\n” to the log file
ELSE
     Pass the malfunction to superior officer (the next element in the chain)
END IF
*where XXX is the type of handler e.g. Space Monkey, and the log file is 'log-bronze.txt'
```

I have given you a utility class called *FileUtility.java* to simplify the process of writing to the log file.

Complete your solution so that it demonstrates the *Chain of Responsibility* pattern.

The unit tests I have provided will guide your solution. Note that the expected result for this task can be found in *expected-bronze.txt'* which is in the root folder of your project.

Make sure that your solution to the BRONZE task does not have any *CheckStyle* errors. You cannot pass this task if your code contains *CheckStyle* errors.

INSTRUCTIONS FOR SILVER TASK

In the *src* folder of your new project, in the package named *uk.ac.tees.silver.username,* modify your solution to the previous task so that:

- *MalfunctionHandler* is now defined as an abstract class containing the instance variables *severity* and *description*
- *MalfunctionHandler* should also define an instance variable *f*, a *File* object that refers to the log file *log-silver.txt*.
- In this class, change the *processMalfunction(Malfunction)* method from *abstract* to *final protected*. The method should function like this:

```
IF severity of malfunction is equal to or below my level of my competence
     handleProblem()
ELSE
     Pass the malfunction to superior officer (the next element in the chain)
END IF
```

- In *MalfunctionHandler,* define a new abstract method called *handleProblem()*
- Change the 4 concrete handler classes so that they extend *MalfunctionHandler*.

- The constructors in the concrete classes should call the superconstructor.
- Implement the *handleProblem()* method in each class to achieve the output listed in *expected-silver.txt.*

The unit tests I have provided will guide your solution. Note that the expected result for this task can be found in *expected-silver.txt'* which is in the root folder of your project.

Make sure that your solution to the SILVER task does not have any *CheckStyle* errors. You cannot pass this task if your code contains *CheckStyle* errors.

INSTRUCTIONS FOR GOLD TASK

In the *src* folder of your new project, in the package named *uk.ac.tees.gold.username,* create a new file called *Bot.java*. This class represents a robot used to explore Mars. The *Bot* class should

- Be an abstract class
- Should declare an instance variable called *identifier* of String type. This will hold the unique serial number of the robot.
- Should declare an instance variable called *f*, a reference to the *File* where the bot will write log entries. This should be set to *log-gold.txt*.
- The constructor will initialise the *identifier* field.
- The Bot class will also define a *checkEnvironment()* method, that always returns true
- Should declare a method called *powerUp()* that writes the following to the log file:

  ```
  <ID>: Powering up.
  ```
- Should declare a method called *powerDown()* that writes the following to the log file:

  ```
  <ID>: Powering down.
  ```
- The *Bot* class should declare an abstract method called doTask()
- The Bot class should declare a template method called executeTask() that does the following

  ```
  if ( checkEnvironment ) {
     powerUp()
     doTask()
     powerDown()
  }
  ```
- The *Bot* class should be extended by *DiggerBot*, *ScannerBot* and *AerialBot*
- The constructors of these sub-classes should call the superconstructor.
- Each of the sub-classes should implement the *doTask*() method
- When the *DiggerBot* is told to *executeTask*(), it will write the following to the log file:

  ```
  <ID>: Burrowing through the Martian regolith.
  ```

- When the *ScannerBot* is told to *executeTask*(), it will write the following to the log file:

  ```
  <ID>: Scanning local terrain for water.
  ```

- When the *AerialBot* is told to *executeTask*(), it will write the following to the log file:

  ```
  <ID>: Flying through the thin Martian atmosphere.
  ```

- Now modify the *checkEnvironment*() method in *AerialBot* to prevent an *AerielBot* flying during a Martian dust storm i.e. the *checkEnvironment()* method will return false when *Environment.dustStorm=true*.
- Now modify the *checkEnvironment*() method in *ScannerBot* to prevent a *ScannerBot* from working if radiation levels exceed MEDIUM i.e. the checkEnvironment() will return false when Environment.RadiationLevel > MEDIUM

The unit tests I have provided will guide your solution. Note that the expected result for this task can be found in *expected-gold.txt* and *expected-gold2.txt* which are in the root folder of your project.

Make sure that your solution to the GOLD task does not have any *CheckStyle* errors. You cannot pass this task if your code contains *CheckStyle* errors.