UNIVERSITY OF JOHANNESBURG

Computer Science 2A

Practical Assignment 08

2016-04-19

Time: Deadline — 2016-04-26 12h00 Marks: 75

This practical assignment must be uploaded to eve.uj.ac.za **before** 2016-04-26 12h00. Late or incorrect submissions **will not be accepted**, and will therefore not be marked. You are **not allowed to collaborate** with any other student.

Good coding practices include a proper coding convention and a good use of JavaDoc comments. Marks will be deducted if these are not present. Every submission **must** include a batch file. See the reminder page for more details.

The Java Development Kit (JDK) has been installed on the laboratory computers along with the Eclipse Integrated Development Environment (IDE).

This practical aims to solidify your understanding of the Visitor Design Pattern.

The story continues from Practical07. After making modifications to the system to view the locations of crew members, you must now make upgrades to the way in which both the layout of the ship and the location of crew members are displayed. The Visitor design pattern has been identified as a means to make it easier to implement future system upgrades. To do this you must implement and make modifications to a number of classes.

Create an **`IDrawable`** interface to specify which classes are drawable. In this interface, create an ***`accept`*** method which takes an **`IDrawVisitor`** as a parameter.

Create an **`IDrawVisitor`** interface to define the visitor for the drawable classes. Create two ***`draw`*** methods which accept a **`CrewEntity`** and a **`ShipLayout`** object respectively.

In the **`ShipLayout`** class, implement the **`IDrawable`** interface. To do this you must implement an ***`accept`*** method, which calls the visitor's ***`draw`*** method and passes ***`this`*** class into it as a parameter. You must also implement int width and height attributes with associated ***`getter`*** and ***`setter`*** methods.

In the **`CrewEntity`** class, implement the **`IDrawable`** interface. To do this you must implement an ***`accept`*** method, which calls the visitor's ***`draw`*** method and passes ***`this`*** class into it as a parameter.

Create a **DrawGraphicsVisitor** class which implements the **IDrawVisitor** interface. To do this you must implement ***draw*** methods for a **CrewEntity** and a **ShipLayout** to be drawn. (Hint: the code which specifies how each of these are drawn is already implemented in the ***paintComponent*** of the **ShipPanel**.) This class needs a **java.awt.Graphics** instance and a method called ***setGraphics*** to set it. Additionally you must create a static int attribute called tileSize, it must be set to be same size as the tileSize in your **ShipPanel** (50).

Update the **ShipPanel** class as follows: Once you have read the layout file and got your **ShipLayout** instance, use the ***setter*** methods created in the **ShipLayout** class to set the width and height attributes of the **ShipLayout**. (Hint: use the ShipPanel's inherited ***getWidth*** and ***getHeight*** methods.) Next, create an instance of the **DrawGraphic-sVisitor** class. Then move the x, y, hts and qts attributes of the **ShipPanel** to the **DrawGraphicsVisitor** class.

In the ***paintComponent*** method of the **ShipPanel** class, use the ***setGraphics*** method to pass the Graphics (g) instance to the **DrawGraphicsVisitor** class instance. Move the code to draw the layout to the ***draw*** method for the **ShipLayout**, then move the code to draw a **CrewEntity** to the ***draw*** method for **CrewEntities**. (Both of these methods are in the **DrawGraphicsVisitor** class.) Next, you must get the layout instance to ***accept*** the visitor. Then loop through each of the **CrewEntity** objects and get each of them to ***accept*** the visitor.

The remaining classes are left unchanged.

Note: Once you have made all of the specified changes, your program should run and look exactly the same as it did before!

Mark sheet

1. **IDrawable** interface and ***accept*** method. [02]
2. **IDrawVisitor** interface and ***draw*** methods. [03]
3. **ShipLayout**
   (a) implements **IDrawable**. [01]
   (b) ***accept*** method. [01]
   (c) width and height attributes (and ***getter***/***setter*** methods). [02]
4. **CrewEntity**
   (a) implements **IDrawable**. [01]
   (b) ***accept*** method. [01]
5. **DrawGraphicsVisitor**
   (a) Grapics instance with ***setter*** method. [02]

  (b) implements **IDrawVisitor**. [01]
  (c) ***draw(CrewEntity)*** method with code moved from **ShipPanel**. [04]
  (d) ***draw(ShipLocation)*** method with code moved from **ShipPanel**. [06]

6. **ShipPanel**
   (a) **DrawGraphicsVisitor** instance. [02]
   (b) set width and height of **ShipLayout** instance (using ***getHeight*** and ***getWidth***). [02]
   (c) ***paintComponent***
      i. **ShipLayout** instance ***accept***s visitor. [02]
      ii. Each **CrewEntity** instance (in the ArrayList) ***accept***s the visitor. [05]
      iii. Does not drawing anything. [05]

7. Packages [05]

8. Coding convention (structure, layout, OO design) and commenting (normal and JavaDoc commenting). [10]

9. Correct execution. [20]

NB

Submissions which **do not compile** will be capped at 40%!

Execution marks are awarded for a correctly functioning application and not for having some related code.

Reminder

Your submission must follow the naming convention as set out in the general learning guide.

```
SURNAME_INITIALS_STUDENTNUMBER_SUBJECTCODE_YEAR_PRACTICALNUMBER
```

Example

Surname	Berners-Lee
Initials	TJ
Student number	209912345
Module Code	CSC2A10
Current Year	2016
Practical number	P00

```
Berners-Lee_TJ_209912345_CSC2A10_2016_P00
```

Your submission must include the following folders:

- `bin` - *(Required)* Should be empty at submission but will contain runnable binaries when your submission is compiled.
- `docs` - *(Required)* Contains the batch file to compile your solution, UML diagrams, and any additional documentation files. Do not include generated JavaDoc.
- `src` - *(Required)* Contains all relevant source code. Source code must be places in relevant sub-packages!
- `data` - *(Optional)* Contains all data files needed to run your solution.
- `lib` - *(Optional)* Contains all libraries needed to compile your solution.

NB

Every submission **must** include a batch file. This batch files must contain commands which will compile your Java application source code, compile the associated application JavaDoc and run the application. **Do not** include generated JavaDoc in your submission. All of the classes/methods which were created/updated need to have JavaDoc comments.