YORK UNIVERSITY
JULY 13, 2023 MIDTERM
EECS 3311
SOFTWARE DESIGN
Duration - 90 min
Aids: None

SOLUTION

York University and you, as a student, share a commitment to academic integrity. You are reminded that you may be charged with an academic offence for using any unauthorized aids during the writing of an exam. A typical penalty for an academic offence may cause you to fail the course.

Please note: you ***CANNOT*** *petition to* ***re-write*** *an examination once the exam has begun.*

Good Luck!

Question 1. [5 MARKS]

In each of the scenarios given below, name the design pattern that would be the most appropriate to implement.

(a) You are designing an application with a Graphical User Interface (GUI). You want to implement a menu structure, where each menu element is either (1) an action that the user can perform, or (2) a submenu containing other menu elements.

Composite

(b) Your application needs a tracker class to maintain a history of some debugging information (e.g., memory usage). Your tracker class will be used by many other objects within your application, but you want to ensure that they are all sharing **one** instance of your tracker class. In other words, you want to avoid the situation where multiple tracker objects are (accidentally or deliberately) instantiated.

Singleton

(c) You are working a photo editing tool that supports exporting images into various formats. You want to make it easy to extend your application to support new file formats. It would be useful to design the export functionality in a way that makes it easy to simply write a new class for any new file format that needs to be supported, with minimal or no changes to the existing code.

Strategy

(d) You are designing a web application that supports several layouts (e.g., smartphone, tablet, and desktop layouts). You want to define a class so that when the user clicks on a link to view a page, the web application will invoke a method in your class to generate a template page based on the user's current layout. Your web application can then insert the content into the template page and send it to the user's device.

Factory or Builder

(e) You are writing a GUI application. You need to implement a pattern that allows the user interface to be automatically notified (so that it can update itself) whenever the data on backend changes.

Observer

Question 2. [6 marks]

Consider the following code:

```
import java.util.Observable;
import java.util.Observer;

class Observation extends Observable implements Observer {
    private String name;

    public Observation(String name) {
        this.name = name;
    }
    public void hello() {
        System.out.println(this.name + " says hello.");
        this.setChanged();
        this.notifyObservers();
    }
    public void update(Observable o, Object arg) {
        System.out.println(this.name + " updates.");
        this.setChanged();
        this.notifyObservers();
    }
    public static void main(String[] args) {
        Observation arnold = new Observation("arnold");
        Observation larry = new Observation("larry");
        Observation ilir = new Observation("ilir");
        Observation student = new Observation("student");

        ____________________________________________    [2 marks]

        ____________________________________________    [2 marks]

        ____________________________________________    [2 marks]

        arnold.hello();
        larry.hello();
        ilir.hello();
        student.hello();
    }
}
```

Complete the **three** blank lines in the above code so that the `main` method has exactly the following output. You're NOT allowed to added any additional lines.

```
arnold says hello.
larry says hello.
ilir says hello.
arnold updates.
student says hello.
larry updates.
ilir updates.
arnold updates.
```

The three lines they need to add are:

ilir.addObserver(arnold);
student.addObserver(ilir);
student.addObserver(larry);

Anything else will not get marks

Note: the order in which observers are updated is not specified per the Javadoc of addObserver, plus the observers are stored as a Set, therefore the order in which ilir and larry are added as observers to student doesn't actually matter.

This also means that the order of the last 3 statements is not set in stone.
Either "larry updates." comes first, or else the combination of "ilir updates." and "arnold updates." comes first.

My thought process of how to solve this:

1. I didn't really get it at first, so I put it in an IDE because it's open book
2. I read the Javadoc of Observer and Observable, to see what methods from those I could use. Specifically I first looked at the notifyObservers method in order to see how that works exactly.
3. I read the slides to refresh my memory of what needs to be done to make the Observer design pattern work.
4. I realized that the code given does NOT subscribe anyone to anything (which would make that update method useless). Therefore, I need to do addObserver in order to make update useful.
5. I played around with the code in the IDE until I got the things in order. Note: I already knew by this point about the thing with the order of the output.

Question 3. [6 MARKS]

Visitor design pattern is a behavioral design pattern that lets you separate algorithms from the objects on which they operate. In Visitor pattern, we use a visitor class which changes the executing algorithm of an element class. By this way, execution algorithm of element can vary as and when visitor varies. As per the pattern, element object has to accept the visitor object so that visitor object handles the operation on the element object.

Given the following example project that use visitor design pattern:

We are going to create a `ComputerComponent` interface defining *accept* operation. It has four concrete subclasses, i.e., `Keyboard`, `Mouse`, `Monitor`, and `Computer` that implement the `ComputerComponent` interface. There is another interface `ComputerComponentVisitor` that defines a *visitor* operation, we implement the `ComputerComponentVisitor` interface with a concrete class `ComputerDisplayVisitor`. `VisitorDemo` is demo and entrance class of this project, it use `Computer` and `ComputerDisplayVisitor` classes to demonstrate use of visitor pattern.

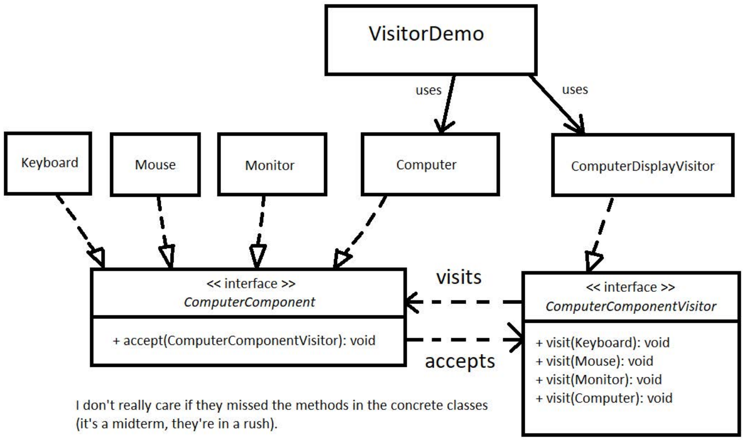
In this question, you task is to draw a UML class diagram to reflect the above project designed by visitor pattern.

VisitorDemo and uses **associations or dependencies**: 2 marks if exactly like below. -0.25 marks for each relationship using the wrong type. -1 mark if a relationship is missing. 0 marks if VisitorDemo or all associations missing. (The previous penalty for -0.5 marks for each relationship that is pointing at an interface instead of a concrete class is removed as the lecture slides directs the uses associations to the interfaces, which conflicts with how the question describes the scenario)

ComputerComponent, **accept**, the 4 implementing classes, correct relationships = 2 marks (each 0.5)

ComputerComponentVisitor, 4 **visit** methods (correspond to concrete Component classes), **ComputerDisplayVisitor**, correct relationship = 2 marks (each 0.5). If # visit methods incorrect (but not totally missing), -0.25

Excluding the VisitorDemo portion, the penalty for "correct relationships" is -0.25 per incorrect relationship (wrong kind used, wrong direction), up to -1 total for the whole question.



I don't really care if they missed the methods in the concrete classes (it's a midterm, they're in a rush).

The minimum for mentioning accept and visit is either:
1. accept in ComputerComponent and 4 visits in the visitor (1 for each concrete class); or
2. visits and accepts dependencies exactly as shown

Misc. penalties:
-1 if you swapped the interface and its implementing class (ex. you labelled ComputerDisplayVisitor as an interface)

Question 4. [6 MARKS]

Consider the following Java implementation of an AirConditioner class that makes use of a BasicThermostat.

```
public class AirConditioner {
   private BasicThermostat thermostat;
   public AirConditioner() {
      thermostat = new BasicThermostat();
   }
   public boolean turnOn() {
     if thermostat.aboveTarget() { ... }
     else { ... }
   }
}
public class BasicThermostat {
   private double targetTemperature;
   public BasicThermostat() {}
   public void setTemperature(double temperature) { ... }
   public boolean aboveTarget() { ... }
}
```

Part (a) [1 MARK]

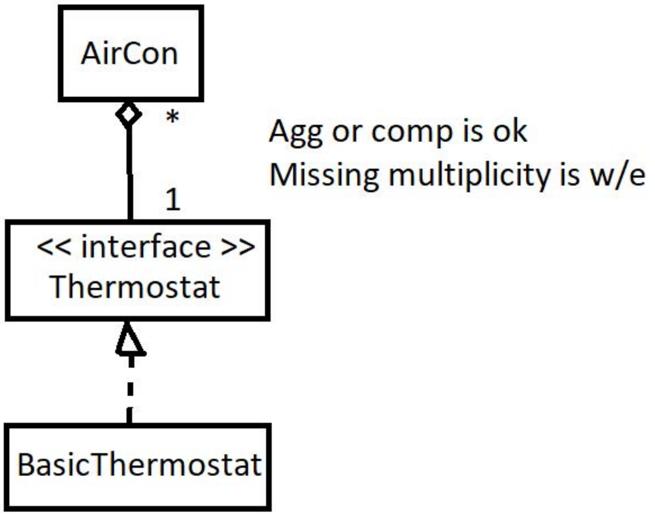
Only Dependency Inversion makes sense.

Name the most relevant SOLID principle violated by this design

Part (b) [5 MARKS]

Provide a UML class diagram of a good solution that addresses the problem you identified in the previous part.

For this one, it doesn't matter if they missed attributes and methods.



If they identified the wrong SOLID principle, but did the UML properly according to the principle they tried to fix, then the maximum they can get is 3/5 for part b.