ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)
ORGANISATION OF ISLAMIC COOPERATION (OIC)
Department of Computer Science and Engineering (CSE)

SEMESTER: BACKLOG EXAMINATION **SUMMER SEMESTER, 2021-2022**

DURATION: 3 Hours **FULL MARKS: 100**

SWE 4501: Design Pattern

Programmable calculators are not allowed. Do not write anything on the question paper.

Answer all **5 (five)** questions. Marks of each question and corresponding CO and PO are written in the right margin with brackets.

---

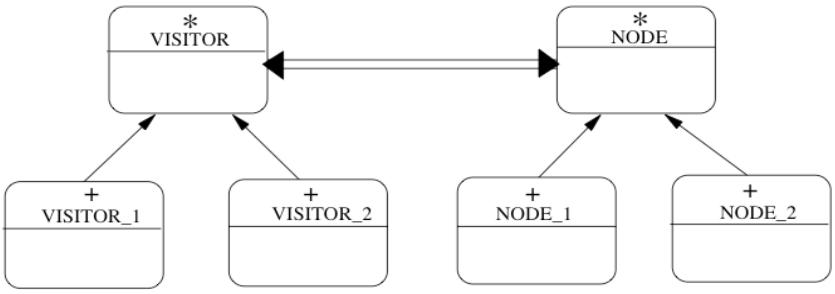
1. a) What are the main components of OOP? Discuss the advantage and disadvantage of using **Composition** over **Inheritance**. **2+3 (CO1) (PO1)**

   b) **Indicate** for each case which **design pattern** you will use: **5 (CO4) (PO2)**
   - i. Be able to replace the implementation of an interface at run time.
   - ii. Decouple clients of a system X from dependencies on subsystems of X.
   - iii. Provide clients with a reference to an object of type X but defer the creation of an expensive object of type X until it is needed.
   - iv. Define a new operation without changing the classes of the elements on which it operates.
   - v. Used to restore state of an object to a previous state.
   - vi. Promote "invocation of a method on an object".

   c) Explain a scenario where **Bridge** pattern can be used. Write the corresponding code for that scenario. Also, draw the UML diagram for that scenario. **10 (CO4) (PO2)**

2. a) What are the relationships between the **Facade** and **Abstract Factory** patterns? **5 (CO3) (PO2)**

   b) **10 (CO4) (PO2)**



Answer the following questions according to the above diagram.
- i) Describe the features required in each deferred class, and a typical effective class in each hierarchy, to support the pattern.
- ii) Suppose a class NODE_C is added as a subclass of NODE. List and describe the required changes to all of the classes affected by the addition.
- iii) Would you advise using the Visitor Pattern if the NODE hierarchy changed frequently? Explain your answer.
- iv) Describe the type of applications that are suitable for the Visitor Pattern.
- v) Explain the term *"Double Dispatch"* in Visitor pattern.

c) Explain **low coupling** and **high cohesion** with example. **5 (CO1) (PO1)**

3. a) Use the **Composite pattern**, to model the notion of a folder in Windows XP. Folders may be nested and may also contain text files and binary files. Files may be opened, closed, and drawn on the screen. Folders may also have items added to and removed from them. Draw the UML diagram for the described model. **5 (CO4) (PO4)**

b) In object-oriented programming one is advised to avoid case (and if) statements. Select one design pattern that helps avoid case statements and explain how it helps. **10 (CO4) (PO4)**

c) Describe the intent and motivation of **Adapter** pattern. Explain a scenario satisfying the motivation. **5 (CO3) (PO4)**

4. a) Draw a UML diagram for **Mediator pattern** between web services and web clients. As web services, the Ebay auction house and Amazon are available. Plan functions to search for an item with a textual description, and to buy an item from the service that gives you the best price. **5 (CO4) (PO2)**

b) **Identify** a pattern which can define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically. Briefly explain that pattern with code. Also discuss the advantages and disadvantages of that pattern. **10 (CO4) (PO2)**

c) It has been suggested that the Decorator design pattern is a degenerate instance of the Composite Design pattern. Explain the statement. **5 (CO3) (PO2)**

5. a) Explain the difference between "**Refactoring"** and "**Design pattern"** **5 (CO2) (PO2)**

b) Write short notes on – "**Duplicated code**" and "**Speculative Generality**". **5 (CO2) (PO1)**

c) Consider the following code snippets – **5+5 (CO2) (PO2)**

```
public class Rental {
   private Movie _movie;
   Private int  _daysRented;

   public Rental (Movie movie, int daysRented) {
      _movie = movie;
      _daysRented = daysRented
   }

   public int getDaysRented() {
      return _daysRented;
   }

   public Movie getMovie() {
      return _movie;)
   }
```

```
    public double amountFor() {
        double thisAmount = 0;
        //determine amounts for each line
        switch (getMovie().getPriceCode()) {
            case Movie.REGULAR:
                thisAmount += 2;
                if (getDaysRented() > 2)
                    thisAmount += (getDaysRented() - 2) * 1.5;
                break;
            case Movie.NEW_RELEASE:
                thisAmount += getDaysRented() * 3;
                break;
            case Movie.CHlLDRENS:
               thisAmount += 1.5;
               if (getDaysRented() > 3)
                    thisAmount += (getDaysRented() - 3) * 1.5;
                break;
    }
    return this.Amount; }
}

public class Movie {
    public static final int CHILDRENS = 2;
    public static final int REGULAR= 0;
    public static final int NEW_RELEASE = 1;

    private String _title;
    private int _priceCode


    public Movie (String title, int priceCode) {
        _title = title;
        _priceCode = priceCode;

     public int getpriceCOde() {
            return _priceCode;
        }
        public void setPriceCode(int arg) {
            _priceCode = arg;
        }
        public String getTitIe {
            return _title;
        }}
```

i. **Identify** two code smells which occur in the code.
ii. **Refacto**r the code removing the smells.