ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

ORGANISATION OF ISLAMIC COOPERATION (OIC)

Department of Computer Science and Engineering (CSE)

SEMESTER FINAL EXAMINATION — SUMMER SEMESTER, 2022-2023

DURATION: 3 HOURS — FULL MARKS: 150

CSE 4851: Design Patterns

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

Answer **all 6 (six)** questions. Figures in the right margin indicate full marks of questions with corresponding COs and POs in parentheses.

---

1. a) What is a design pattern? Explain why design patterns are important in software development. — 5 (CO1) (PO1)

   b) For each of the following cases, indicate which design pattern you will apply — — 2 × 5 (CO4) (PO2)

      i. Be able to replace the implementation of an interface at run time.

      ii. Encapsulates a request as an object, allowing for the separation of sender and receiver.

      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.

   c) Describe the Singleton design pattern. What problem does it solve, and how is it implemented? Provide a simple code example demonstrating the implementation of the Singleton pattern. — 10 (CO3) (PO1)

2. a) What are the differences between the Strategy and Decorator pattern? — 5 (CO1) (PO1)

   b) A popular online bookstore platform that caters to readers worldwide. A customer places an order for historical novels on the bookstore platform. The order is passed through a chain of responsibility, consisting of distinct handlers for validation, discount application, payment processing, and shipping respectively. The order moves seamlessly through each handler in the chain. If at any stage the order fails validation or encounters an issue, the processing is halted, ensuring a smooth and error-resistant order fulfillment process.

   Which design pattern will you apply to implement the scenario? Write the corresponding code to implement the scenario using that appropriate pattern. — 12 (CO4) (PO2)

   c) "Program to an Interface, not to an Implementation" - Explain the statement with an appropriate example. — 8 (CO1) (PO1)

3. a) An application contains an interface, `Shape`, implemented by two concrete shapes (`Circle` and `Rectangle`). Several composite shapes can be created by using these two concrete shapes. Composite objects can be visited by a visitor from the outside of the application with the help of a `ShapeVisitor` interface.

   Write code for the above-mentioned scenario using appropriate pattern and draw the corresponding UML. — 15 (CO4) (PO2)

   b) Explain the intent and motivation of the Proxy pattern. Describe a real-world scenario where you can use State pattern. — 10 (CO3) (PO1)

4. a) Use 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, or drawn on the screen. Folders may also have items added and removed from them. Draw the UML diagram for the described model. — 10 (CO3) (PO1)

   b) Perform a comparative analysis among Singleton, Prototype, and Flyweight patterns. — 10 (CO3) (PO1)

5. 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. — 10 (CO3) (PO1)

   b) Identify two design patterns that reduce memory footprint. Perform a comparative analysis between them. — 10 (CO3) (PO1)

   c) Identify a pattern that decouples an abstraction from its implementation so that the two can vary independently. Explain a scenario satisfying the statement. — 10 (CO4) (PO2)

6. a) Write short notes on "Speculative Generality" and "Primitive Obsession". — 10 (CO1) (PO1)

   b) Consider the classes used in a movie rental system as in Code Snippet 1 and 2. — 5 × 3 (CO2) (PO2)

```
1  public class Rental {
2      private Movie _movie;
3      Private int  _daysRented;
4
5      public Rental (Movie movie, int daysRented) {
6          _movie = movie;
7          _daysRented = daysRented
8      }
9
10     public int getDaysRented() {
11         return _daysRented;
12     }
13
14     public Movie getMovie() {
15         return _movie;)
16     }
17
18     public double amountFor() {
19         double thisAmount = 0;
20         //determine amounts for each line
21         switch (getMovie().getPriceCode()) {
22             case Movie.REGULAR:
23                 thisAmount += 2;
24                 if (getDaysRented() > 2)
25                     thisAmount += (getDaysRented() - 2) * 1.5;
26                 break;
27             case Movie.NEW_RELEASE:
28                 thisAmount += getDaysRented() * 3;
29                 break;
```

```
30              case Movie.CHlLDRENS:
31                 thisAmount += 1.5;
32                 if (getDaysRented() > 3)
33                      thisAmount += (getDaysRented() - 3) * 1.5;
34                  break;
35      }
36      return this.Amount; }
```

Code Snippet 1: Java program of `Rental` class for Question 6.b

```
1  public class Movie {
2      public static final int CHILDRENS = 2;
3      public static final int REGULAR= 0;
4      public static final int NEW_RELEASE = 1;
5
6      private String _title;
7      private int _priceCode
8
9
10
11     public Movie (String title, int priceCode) {
12         _title = title;
13         _priceCode = priceCode;
14
15      public int getpriceCOde() {
16             return _priceCode;
17         }
18         public void setPriceCode(int arg) {
19             _priceCode = arg;
20         }
21         public String getTitIe {
22             return _title;
23         }
24 }
```

Code Snippet 2: Java program of `Movie` class for Question 6.b

Answer the following questions according to Code Snippets 1 and 2.

i. Briefly explain the terms "Code refactoring" and "Code smell".

ii. Identify two code smells that have occurred in the code.

iii. Refactor the code removing the smells.