98-Comp-B11, Advanced Software Design

3 hours duration

NOTES:

1. If doubt exists as to the interpretation of any question, the candidate is urged to submit, with the answer paper, a clear statement of any assumptions made.
2. No Calculator permitted. This is an open book exam.
3. You are requested to answer:
   a. any three (3) questions in PART I
      (only the first three questions of PART I as they appear in your answer book will be marked)
   b. any four (4) questions in PART II
      (only the first four questions of PART II as they appear in your answer book will be marked)
   c. any three (3) questions in PART III
      (only the first three questions of PART III as they appear in your answer book will be marked)
   d. any (1) question in PART IV
      (only the first question of PART IV as it appears in your answer book will be marked)
   e. any (1) question in PART V
      (only the first question of PART V as it appears in your answer book will be marked)
4. All questions have equal weight.

PART I—General principles

Question 1
What is the difference between the notions of *incremental* and *iterative* when one talks about a software development process? You may use example development processes to illustrate your answer.

Question 2
Explain the following two terms, i.e., provide definitions and examples:

a. functional requirement
b. non-functional requirement

Question 3
Read the following description and identify three functional requirements as well as three non-functional requirements. Justify your answers.
"*To allocate staff to production lines, a process will be run once a week to carry out the allocation based on the skills and experience of operatives. Details of holidays and sick leave will also be taken into account. A first draft Allocation List will be printed off by 12.00 noon on Friday for the following week. Only staff in Production Planning will be able to amend the automatic allocation to fine-tune the list. Once the amendments have been made, the final Allocation List must be printed out by 5.00 pm. The system must be able to handle allocation of 100 operatives at present, and should be capable of expansion to handle double that number.*"

Question 4
What do we mean when we say that non-functional requirements should be quantifiable? Illustrate your answer with examples.

Question 5
There is increasing interest for the use of *agile* methods of software development. Why is this? Are there any dangers associated with agile methods? Discuss.

PART II—Design

Question 6
What is the open-close principle all about?
Why is it important that classes be "open" and "close" at the same time?
Using an example of framework that you know, illustrate the use of the open-close principle and its benefits.

Question 7
Explain the following terms and how they relate to one another:

a. polymorphism
b. dynamic binding
c. overloading
d. overriding

Question 8
What is software reuse?
What are the benefits of reusable components?
What are the challenges of software reuse?

Question 9
Suppose you are designing a software to help dispatch cabs. One of your colleagues argue that since the use case diagram shows an actor named Dispatcher, then the class diagram of your Requirement Analysis phase should have a class named Dispatcher. Similarly, that colleague argues that since this class diagram shows a class named Customer, then your use case diagram should have an actor named Customer.
Do you agree with those statements? You must justify your answer.

Question 10
What is *design by contract*?
Explain and provide an example.
What are the benefits of design by contract?
In your opinion, what are the main challenges of applying design by contract?

Question 11
What is the Liskov substitution principle all about, and why is it useful?

Question 12
During system design, one decides, in general, to follow one design alternative rather than another one because of design goals. What are those goals typically and who sets them?

PART III—Patterns

Question 13

Define the term framework.
Define the term Library.
Define the term Design Pattern.
What are the main differences between a framework, a library and a design pattern?

Question 14

Explain in details three possible uses of the Proxy design pattern.

Question 15

A comparison between the Bridge and Strategy design patterns states that Bridge is a structural pattern and Strategy is a behavioural pattern. What do theses two terms (structural, behavioural) mean?

Question 16

Classes can typically be classified into either one of the following categories: Boundary classes are used for interactions between actors and the software; Entity classes are used to store data; Control classes implement the control flow to realize use cases.
One argues that in the observer design pattern, the ConcreteObserver is not necessarily a boundary (GUI) class and that the ConcreteSubject is not necessarily an entity class.
Why is that? In other words, why (or when) are the roles of the ConcreteObserver and ConcreteSubject played by other types of classes than boundary and entity classes? Justify your answer.

Question 17

Suppose you have to design an application that checks the validity of credit cards. For simplicity, let us consider only three types of credit cards—Visa, MasterCard, DinersClub. The application carries out a series of validations on the input credit card information as a series of four steps, which are always all carried out in the same order: Step1—verify the expiration date; Step 2—verify the length of the credit card number; Step 3—verify that the credit card number has valid characters; Step 4—check whether the account is in good standing. These steps are always performed in the same order but are obviously performed differently depending on the credit card.
Which design pattern(s) would you use to allow the easy implementation of these validation steps, and ensure that it will be easy to add other credit cards in the future?
Justify your decision. Explain the design pattern(s), its (their) constituents... and draw the basic class structure of the pattern(s)-based solution.

PART IV—GUI Design

Question 18

In order to separate the application from its interface, the Model-View-Control (MVC) architecture is a solution often proposed. Completely describe the MVC architecture.

For each design goal below, indicate whether the MVC architecture helps or hurts. Justify your answer in each case.

i. extensibility of the system
ii. response time
iii. modifiability of the design

Question 19

During Analysis and Design, engineers separate class responsibilities into Entity, Control and Boundary classes, the latter being responsible for interfaces between the system and the actors (e.g., human actors). In other words, a Boundary class represents a GUI (or part of it) when the actor is a human. (The other two kinds of classes are responsible for storing data and executing use cases, respectively.)

Why is it important to follow such a strategy for class responsibility assignment, and especially why is it important to separate Boundary (i.e., GUI) classes from the other two?

To answer the question you may discuss the impact of not following such a strategy on several software engineering activities such as implementation, testing and maintenance.

Question 20

Among the following eight Design Patterns, select <u>two</u> that are paramount during GUI Design. Describe the patterns you selected (diagrams are welcome) and justify your selection.

Adapter, Bridge, Command, Composite, Decorator, Façade, Observer, Proxy

Question 21

Site, define, and illustrate three (3) non-functional software requirements that specifically pertain to Human–Machine interfaces. For each non-functional requirement, discuss one example design strategy that can ensure the requirement is met.

PART V—C++/Java and Modular Programming

Question 22
In C++, friend functions and friend classes can be used to improve performance (for instance), but are considered detrimental to modular programming and maintenance. Why? (Justify your answer.)

Question 23
Why is it necessary to sometimes replace *inheritance* with *delegation* both in design and programming? You may use a UML example to aid your answer.

Question 24
Consider the Java definition of class `Point` on the right.

```
public class Point {
  private final int x;
  private final int y;
  public Point(int x, int y) {
    this.x = x;
    this.y = y;
  }
  public int getX() { return x; }
  public int getY() { return y; }

  public boolean equals(Point other) {
    return ( this.getX() == other.getX()
    && this.getY() == other.getY() );
  }
}
```

The following Java code uses class `Point` defined above. Comments indicate the outcome of the three `println()` calls.
Method `contains(Object o)` returns true if the `hashset` contains the specified element `o`.
What can explain the strange outcome of the last `println()` call?

```
import java.util.HashSet;
Point p1 = new Point(1, 2);
Point p2 = new Point(1, 2);
Point q = new Point(2, 3);
System.out.println(p1.equals(p2)); // prints true
System.out.println(p1.equals(q)); // prints false
HashSet<Point> coll = new HashSet<Point>();
coll.add(p1);
System.out.println(coll.contains(p2)); // prints false
```

Question 25
Discuss how a design requiring multiple inheritance can be implemented in Java. You may want to consider different characteristics of the "classes" being inherited and how this impact the actual implementation.