May 2016

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 a closed book exam with one aid sheet allowed written on both sides.
3. You are requested to answer:
   a. Any five (5) questions in PART I (only the first five questions of PART I as they appear in your answer book will be marked)
   b. Any three (3) questions in PART II (only the first three questions of PART II as they appear in your answer book will be marked)
   c. Any four (4) questions in PART III (only the first four questions of PART III as they appear in your answer book will be marked)
   d. Any two (2) questions in PART IV (only the two questions of PART IV as it appears in your answer book will be marked)
   e. Any five (5) questions in PART V (only the five questions 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
Differentiate between the “Waterfall lifecycle model” and the “Spiral model.” How are they related (differences and similarities)?

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

a. functional requirement
b. non-functional requirement

Question 3
What do we mean when we say that non-functional requirements should be quantifiable? Illustrate your answer with examples.

Question 4
Suppose you are designing software for a library. One of your colleagues argues that since the use case diagram shows an actor named Librarian, then the class diagram of your Requirement Analysis phase should have a class named Librarian. Similarly, that colleague argues that since this class diagram shows a class named Patron, then your use case diagram should have an actor named Patron.
Do you agree with those statements? You must justify your answer.

Question 5
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?

Question 6
Explain the difference between a fault, an error, and a failure.

Question 7
What is software reuse?
What are the benefits of reusable components?
What are the challenges of producing reusable software?

PART II—Design by Contract

Question 8
What is the difference between *design by contract* and *defensive programming*? Explain and provide an example.

Question 9
What are the benefits of design by contract?
In your opinion, what are the main challenges of applying design by contract?

Question 10
What is a precondition? What is a post-condition? What is a class invariant?

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

Question 12
Suppose class A has a class invariant that includes the Boolean logic clause (x >= 0 and y = 0).
Suppose class B inherits from A and adds a new class invariant that includes the Boolean logic clause (x >= 0 implies y >= 0).
Is this acceptable? Why or why not?

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
The Proxy design pattern can be used in three different types of situations. Each type of situation has it own name. One of them is the Remote Proxy. What are the other two types of proxies? Define the three types.

Question 15
The "Gang of four" patterns are classified as creational, structural or behavioural. What do these terms mean? Give one example design pattern of each kind and justify the classification of each pattern.

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 <u>not</u> necessarily a boundary (GUI) class and that the ConcreteSubject is <u>not</u> 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.

- extensibility of the system
- response time
- 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 two 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
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 23
Explain the following terms and show how they are related to one another:
Encapsulation Information hiding Message passing

Question 24
Explain the following terms and how they relate to one another:

- polymorphism
- dynamic binding
- overloading
- overriding

Question 25
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 26
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 27
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.

Question 28
An input variable defines the price for a product. The specification indicates that the price must be greater than or equal to $10.00 and cannot be greater than $1,200.00.
Which of the following is **a** good selection of input values for unit testing the function? (Selection A uses four values, i.e., $6.00, $9.00 ... whereas selection B uses three values...) Select either A, B, C, D, or E and justify your answer.

A	6.00	9.00	55.00	900.00	
B	8.50	150.00	1200.00		
C	5.12	10.00	500.00	1,200.00	1,500.30
D	5.00	600.00	1300.00	1500.00	
E	10.00	150.00	1300.00		