National University of Computer and Emerging Sciences

Islamabad Campus

Software Design & Analysis (CS-3004)

Date: Sep 23, 2024

Course Instructors

Basharat Hussain, Javaria Imtiaz, Majid Hussain, Sidra Khalid

Sessional-1 Exam

Total Time (Hrs): 1

Total Marks: 50

Total Questions: 4

Roll No _____ Section _____ Student Signature _____

Attempt all the questions on answer sheet. Only attempt MCQs (Question #1) on question paper and return with the answer sheet.

Answer Mode is Displayed!

Question # 1 [Marks = 10]

Answer all the multiple-choice questions in the bubble sheet provided here. You must fill the roll number as follows: 22i-9876 to 229876

DO NOT WRITE BELOW THIS LINE

1. **Consider the following activities.**

   (i) Transforming the Analysis Use cases to Design Use cases

   (ii) Identify class attributes, class behaviors and responsibilities

   (iii) Model object states

   Which of the above statements is/are activities of Object-Oriented Design?

   (a) Only (i)
   (b) Only (ii)
   (c) Only (ii) and (iii)
   (d) Only (i) and (iii)
   (e) All

   Answer: e) All

2. **Which of the following is the correct way of implementing an interface A and inheriting class B by class C?**

   (a) class C implements class A extends B
   (b) class C inherits B implements A
   (c) class C imports B implements A
   (d) None of the mentioned

   Answer: d) None of the mentioned

3. **Consider the following statements with respect to Use case modeling.**

   (i) An automatic system backup that runs every evening can be represented by a Time actor.

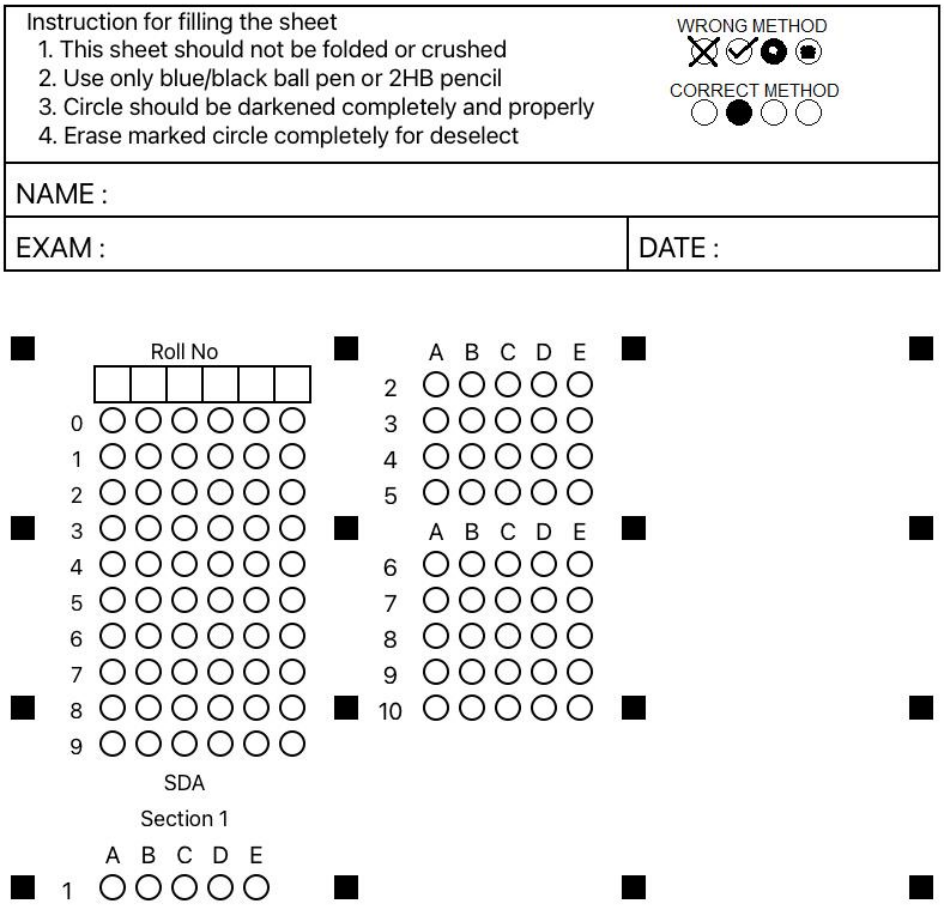

Figure 1: Answer sheet for auto scan

(ii) Credit bureau authorizing the charging by a credit card is an example for an external server actor.

(iii) Warehouses receiving a package order to prepare a shipment is an example for an external receiver actor if the warehouse is outside the scope of the system.

Which of the above statements is/are correct?

(a) (i) only
(b) (ii) only
(c) (i) and (ii) only
(d) (ii) and (iii) only
(e) All

Answer: e) All

4. **Consider the following scenario.** A banking application has several different types of accounts. In some types of accounts, interest is computed as a percentage of the average daily balance during a month. In other types of accounts, interest is computed as a percentage of the minimum daily balance during a month. In a mortgage account, interest is computed as a percentage of the balance at the end of the month. What is/are the most appropriate method(s) to model the above scenario?

(a) Polymorphism
(b) Inheritance
(c) Aggregation
(d) Composition
(e) Association

Answer: a) Polymorphism

5. **Which of the following is an example of a non-functional requirement?**

(a) The system must allow users to upload files
(b) The system must send an email confirmation after registration
(c) The system must support up to 1 million users
(d) The system must allow users to reset their passwords

Answer: c) The system must support up to 1 million users

6. **Which of the following is a key difference between an abstract class and an interface?**

(a) Abstract classes can have both abstract and non-abstract methods; interfaces can only have abstract methods (before Java 8).
(b) Interfaces can have constructors; abstract classes cannot.
(c) Interfaces support multiple inheritance; abstract classes do not.
(d) None of the above.

Answer: a) Abstract classes can have both abstract and non-abstract methods; interfaces can only have abstract methods (before Java 8).

7. **Which of the following statements about abstract class constructors is true?**

(a) An abstract class cannot have a constructor because you cannot create instances of an abstract class.
(b) An abstract class can have a constructor, and it is called when an instance of a concrete subclass is created.
(c) The constructor of an abstract class is only called when an instance of the abstract class is created directly.
(d) An abstract class can have multiple constructors, but they can only be called via super() in the subclass.

Answer: b) An abstract class can have a constructor

8. **Consider the following use case diagram instructions: Use Case A has an "include" relationship with Use Case B. Use Case A has an "extend" relationship with Use Case C.** Which of the following statements is true?

(a) Use Case A always includes the behavior of Use Case B and optionally adds the behavior of Use Case C.
(b) Use Case A optionally includes the behavior of Use Case B and always extends with Use Case C.
(c) Use Case A can execute independently of Use Case B and always includes the behavior of Use Case C.

(d) Use Case A and Use Case B have the same behavior since they are linked by "include."

Answer: a) Use Case A always includes the behavior of Use Case B

9. **Each hourly employee has a wage for the job he/she performs, and a wage is a product of hours worked and the hourly rate. In a Domain Model, wage would be modeled as**

   (a) attribute of class HourlyEmployee

   (b) Class belonging to the association between classes HourlyEmployee and Job.

   (c) sub-class of class HourlyEmployee

   (d) Concept

   Answer: a) Attribute of class HourlyEmployee

10. **How do operations differ from methods?**

    (a) A method is a particular implementation of an operation.

    (b) An operation is a particular implementation of a method.

    (c) Some object-oriented programming languages have methods, while others have operations.

    (d) Both are the same

    Answer: a) A method is a particular implementation of an operation

---

Question # 2 [Marks = 11]

(a) What will be the output of the following code? (2 Marks)

Code Listing 1: Java Code Example

```
interface Drivable {
    void drive();
}
abstract class Machine {
    void start() {
        System.out.println("Machine started");
    }
}
class Engine extends Machine {
    @Override
    void start() {
        System.out.println("Engine started");
    }
}
class Vehicle extends Machine implements Drivable {
    Engine engine = new Engine(); // Composition
    @Override
    public void drive() {
        engine.start();
        System.out.println("Vehicle is driving");
    }
    @Override
    void start() {
        System.out.println("Vehicle started");
    }
}
```

```
27 class Car extends Vehicle {
28     @Override
29     public void drive() {
30         System.out.println("Car is being driven");
31     }
32 }
33 public class Main {
34     public static void main(String[] args) {
35         Vehicle vehicle = new Car();
36         vehicle.start();
37         vehicle.drive();
38     }
39 }
```

(b) What will be the output of the following code? (5 Marks)

Code Listing 2: Java Code Example

```
abstract class Worker {
    Worker() {
        System.out.println("Worker is created");
    }

    abstract void doWork();
}

class Developer extends Worker {
    Developer() {
        super();
        System.out.println("Developer is ready");
    }

    @Override
    void doWork() {
        System.out.println("Developer is writing code");
    }
}

class SeniorDeveloper extends Developer {
    SeniorDeveloper() {
        super();
        System.out.println("Senior Developer is ready");
    }

    @Override
    void doWork() {
        super.doWork();
        System.out.println("Senior Developer is optimizing code");
    }
}

public class Main {
    public static void main(String[] args) {
        SeniorDeveloper seniorDev = new SeniorDeveloper();
        seniorDev.doWork();
    }
}
```

(c) Write the required Java classes to illustrate an aggregation relationship between Department and Employee using a List collection. (4 Marks)

Answer:

A: Vehicle started
Car is being driven
B: Worker is created
Developer is ready
Senior Developer is ready
Developer is writing code
Senior Developer is optimizing code

C: class Department {
private String name;

```
private List¡Employee¿ employees;

public Department(String name, List¡Employee¿ employees) {
this.name = name;
this.employees = employees;
}
}
```

Question # 3 - A Case Study [Marks = 24]

Consider the requirements for Customer queue management system for restaurants and attempt part a-d.

Restaurant Waiting List At a popular restaurant within a mall, a system is used to manage the waiting queue. When a new customer arrives at the restaurant, on given the number of people in the party, waiter checks on the system the expected waiting time to get a table; then the customer provides his/her name and mobile phone number, the system then sends a confirmation SMS to the customers phone. At this point the customer has a table reserved. Then the customer is free to walk away and can visit the shops in the mall or roam around. Every five minutes the system will send an SMS with the estimated residual waiting time. When the table for the customer becomes available, a waiter sets the table as ready, and the system sends an SMS to the customer asking her/him to get to the restaurant as soon as possible to get the reserved table. If the customer does not show up within a fixed time, then the next customer in the waiting queue is recalled.

(a) Draw a Use Case Diagram for the above case study. Clearly mark the system boundary and label all actors properly. Use proper naming conventions for Use Cases. (5 Marks)

(b) Write the Expanded/Fully-dressed Use case for Accept a New Customer (14 Marks))

(a) Use case name (1)
(b) Scope the system under design (1)
(c) Level (1)
(d) Primary actor (1)
(e) Stakeholders and interests (2)
(f) Preconditions (1)
(g) Post conditions (1)
(h) Main success scenario (4)
(i) Extensions (2)

(c) Draw a Domain model diagram of this case study. Clearly identify the conceptual classes, their attributes and relationships between them. (5 Marks)

Answer:

(a)

(d) **UC1: Accept a New Customer**

Actor: Waiter
Type: Primary

Description: Waiters log in to the Customer Queue Management System. After securely logging in, the waiter enters the number of people for the reservation of a table and the customer's mobile number. The waiter accepts the customer into the waiting queue if the customer receives a confirmation SMS on their mobile number.

Expanded Use Case

- **Use case name:** Accept a New Customer
- **Scope of the System Under Design:** Customer Queue Management System

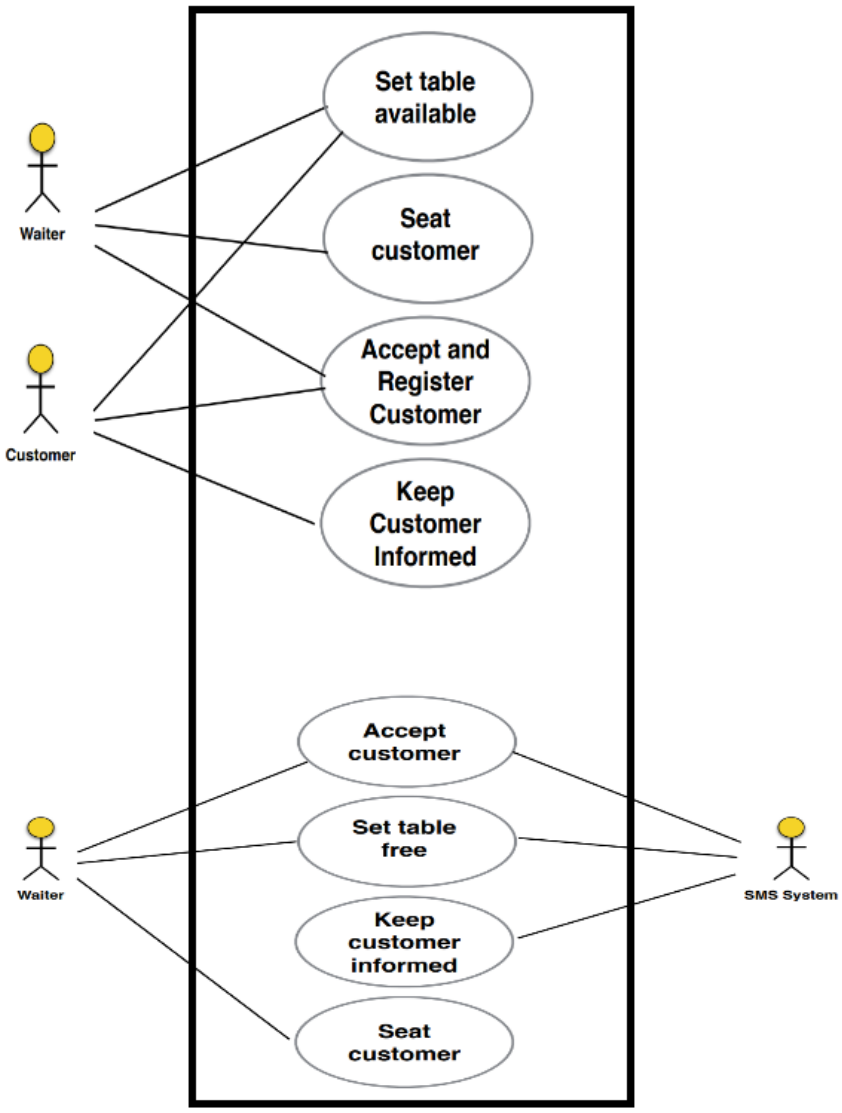

Figure 2: LU factorization algorithm

- **Level:** User-goal
- **Primary Actor:** Waiter
- **Stakeholders and Interests:**
  - Customer: Appraise the expected waiting time and enter the waiting list.
  - Restaurant Owner: Increase sales and revenue by facilitating the customer.
  - SMS Provider Company: Profit and facilitate communication between customer and waiter.
- **Preconditions:**
  - The customer has a valid mobile number.
  - Waiter is logged into the system.
- **Postcondition:** The customer is entered into the waiting list for a suitable table.
- **Main Success Scenario:**
  1. The waiter enters the number of people in the party.
  2. The system identifies a suitable table and provides the expected waiting time.
  3. The waiter enters the name and mobile number for the customer.
  4. The system validates and sends the customer an SMS.

5. The customer receives the SMS message.
6. The waiter confirms the correct reception of SMS.
7. The system enters the customer into the waiting list for the table. The use case terminates with success.

– **Extensions:**
  – 2.b. The system cannot find a suitable table: The system shows a message and the use case ends with failure.
  – 4.b. The mobile number is incorrect: The system shows an error message and the use case returns to step 3.
  – 6.b. The waiter informs that the message has not been received: The use case proceeds to step 3.

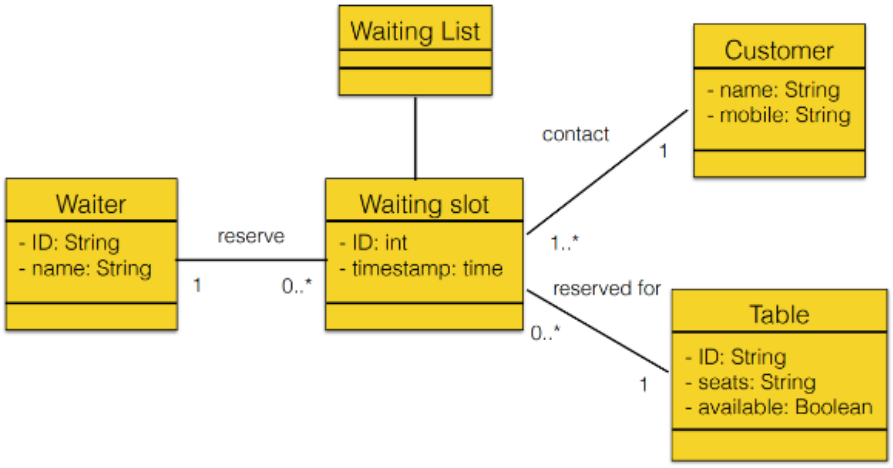

Figure 3: LU factorization algorithm

(e) (a)

Question # 4 [Marks = 05]

(a) Identify the issues in the provided use cases and provide the recommendations for writing good use case in each specific context (Do write on Answer sheet). (5 Marks)

Use Case Scenario	Issues	Solution
(A) Use Case Name: Generate Report Actor: Manager Description: The Manager generates a report based on selected criteria. Steps: 1. Manager logs into the system. 2. Manager navigates to the reports page. 3. Manager selects the report type from the dropdown menu. 4. Manager enters the date range for the report. 5. Manager clicks "Generate". 6. The system queries the database using SQL and joins multiple tables to collect the data. 7. The system formats the data into a CSV file. 8. The report is displayed to the manager.		
(B) Use Case Name: Manage Products Actor: Inventory Clerk Description: The Inventory Clerk can add, edit, delete, and view products. Steps: 1. Clerk logs in. 2. Clerk selects Manage Products from the menu. 3. Clerk can: o Add a product. o Edit a product. o Delete a product. o View products.		
(C) Use Case Name: Approve Purchase Order Actor: Purchasing Manager Description: The Purchasing Manager approves a purchase order submitted by a supplier. Steps: 1. The Purchasing Manager receives the purchase order from the supplier. 2. The Purchasing Manager reviews the purchase order. 3. The Purchasing Manager contacts the supplier if changes are needed. 4. The Purchasing Manager approves the purchase order.		
(D) Use Case Name: Generate Report Actor: Administrator Description: The Administrator generates a report. Steps: 1. System displays the report generation screen. 2. System collects data. 3. System generates a report. 4. System displays the report to the Administrator.		
(E) Use Case Name: Process Loan Application Actor: Loan Officer Description: The Loan Officer processes a loan application for a customer. Steps: 1. Loan Officer logs in. 2. Loan Officer opens the loan application form. 3. Loan Officer enters customer details. 4. Loan Officer uploads customer documents. 5. Loan Officer enters loan amount. 6. Loan Officer selects loan term. 7. Loan Officer submits the form. 8. System checks the credit score of the customer. 9. System validates the loan amount against the customer's income. 10. System checks the uploaded documents for accuracy.		

Identify the issues in the provided 5 use cases and provide the recommendations for writing good use case in each specific context. (5 Marks)

Use cases	Issues	Solution
Use Case Name: Generate Report **Actor**: Manager **Description**: The Manager generates a report based on selected criteria. **Steps**: 1. Manager logs into the system. 2. Manager navigates to the reports page. 3. Manager selects the report type from the dropdown menu. 4. Manager enters the date range for the report. 5. Manager clicks "Generate". 6. The system queries the database using SQL and joins multiple tables to collect the data. 7. The system formats the data into a CSV file. 8. The report is displayed to the manager.	**Implementation Details:** Step 6 and 7 include technical details like using SQL and joining tables, which are irrelevant in a use case and should be left for the design phase. **Overly Technical:** The use case should focus on what the system does from the user's perspective, not how it achieves those tasks technically.	
Use Case Name: Manage Products **Actor**: Inventory Clerk **Description**: The Inventory Clerk can add, edit, delete, and view products. **Steps**: 1. Clerk logs in. 2. Clerk selects "Manage Products" from the menu. 3. Clerk can: ○ Add a product. ○ Edit a product. ○ Delete a product. ○ View products.	• Too **Generic**: The steps are a simple listing of CRUD actions without any details on how each action is performed. There's no explanation of how products are added, edited, or deleted. • Repetitive: These CRUD actions could be better described with a broader use case, like "Manage Product Inventory", with declarative requirements rather than breaking it into multiple CRUD-based steps. • No **User Goals**: The use case fails to describe the goal the actor wants to achieve. What value is the Inventory Clerk getting from these actions?	
Use Case Name: Approve Purchase Order **Actor**: Purchasing Manager **Description**: The Purchasing Manager approves a purchase order submitted by a supplier. **Steps**: 1. The Purchasing Manager receives the purchase order from the supplier. 2. The Purchasing Manager reviews the purchase order. 3. The Purchasing Manager contacts the supplier if changes are needed.	**Mixing Business and System Use Cases**: The first two steps describe business processes that are outside the system's scope. If this interaction happens through a system, it should focus on how the manager interacts with the system, not the business process outside it. **No System Boundary**: The interaction with the supplier isn't supported by the system, so it shouldn't be included in this use case.	

4. The Purchasing Manager approves the purchase order.		
Use Case Name: Generate Report **Actor**: Administrator **Description**: The Administrator generates a report. **Steps**: 1. System displays the report generation screen. 2. System collects data. 3. System generates a report. 4. System displays the report to the Administrator.	**No Real User Actions**: The use case doesn't describe how the Administrator interacts with the system. The steps are entirely focused on what the system does, with no user input or decisions. **Lack of Detail**: What kind of report? What criteria are used? Does the Administrator select parameters or specify data ranges? This is too vague.	
Use Case Name: Process Loan Application **Actor**: Loan Officer **Description**: The Loan Officer processes a loan application for a customer. **Steps**: 1. Loan Officer logs in. 2. Loan Officer opens the loan application form. 3. Loan Officer enters customer details. 4. Loan Officer uploads customer documents. 5. Loan Officer enters loan amount. 6. Loan Officer selects loan term. 7. Loan Officer submits the form. 8. System checks the credit score of the customer. 9. System validates the loan amount against the customer's income. 10. System checks the uploaded documents for accuracy. 11. System generates a loan approval report. 12. System updates the loan database. 13. System sends a confirmation email to the customer. 14. Loan Officer logs out.	**Overly Detailed**: This use case includes too many technical details that aren't needed, such as steps 8-13, which describe internal system processes. **Too Long**: The use case tries to capture every single step, which makes it difficult to follow. It should focus on the high-level interactions between the Loan Officer and the system.	