UNIVERSITI TUNKU ABDUL RAHMAN

ACADEMIC YEAR 2018/2019

APRIL EXAMINATION

UECS2344 SOFTWARE DESIGN

THURSDAY, 25 APRIL 2019 TIME : 2.00 PM – 4.00 PM (2 HOURS)

BACHELOR OF SCIENCE (HONS) SOFTWARE ENGINEERING

Instructions to Candidates:

This paper consists of two sections: Section A and Section B.

Answer all 3 questions in Section A.
Answer any 1 out of the 2 questions in Section B.

All questions carry equal marks.

This question paper consists of 5 questions on 11 printed pages.

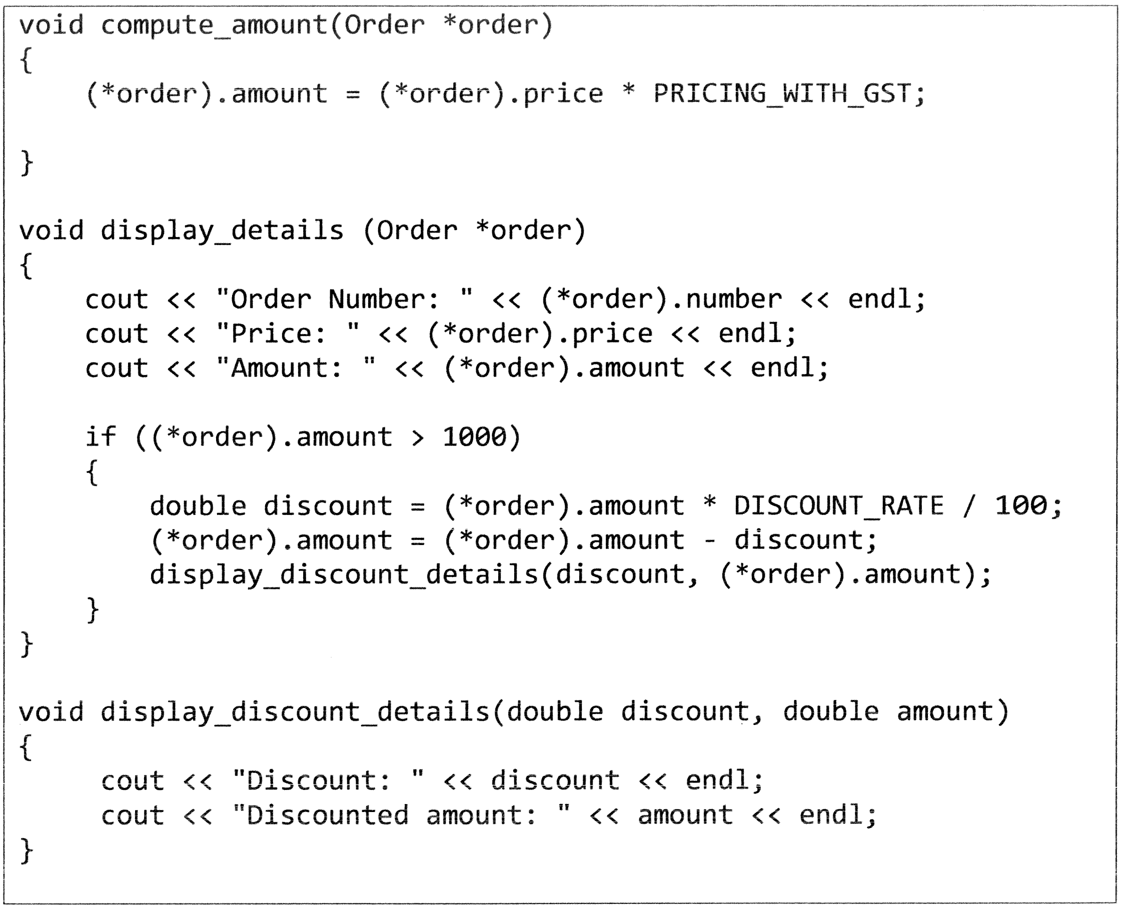
Section A: Answer ALL questions in this Section.

Q1. Consider the C++ program shown in Figure 1.1 below which uses the structured or procedural paradigm.

```
struct Order
{
    int number;
    double price;
    double amount;
};

#define PRICING_WITH_GST 1.06
#define DISCOUNT_RATE 10

Order create_order();
void compute_amount(Order *order);
void display_details (Order *order);
void display_discount_details(double discount, double amount);


int main(void)
{
    Order order;
    int count;

    cout << "How many orders? ";
    cin >> count;

    for (int i=0; i<count; i++)
    {
        order = create_order();
        compute_amount(&order);
        display_details(&order);
    }
    return 0;
}

Order create_order()
{
    Order order;

    cout << "Enter order number: ";
    cin >> order.number;

    cout << "Enter price: ";
    cin >> order.price;

    return order;
}
```

Figure 1.1

Q1. (Continued)



```
void compute_amount(Order *order)
{
    (*order).amount = (*order).price * PRICING_WITH_GST;

}

void display_details (Order *order)
{
    cout << "Order Number: " << (*order).number << endl;
    cout << "Price: " << (*order).price << endl;
    cout << "Amount: " << (*order).amount << endl;

    if ((*order).amount > 1000)
    {
        double discount = (*order).amount * DISCOUNT_RATE / 100;
        (*order).amount = (*order).amount - discount;
        display_discount_details(discount, (*order).amount);
    }
}

void display_discount_details(double discount, double amount)
{
     cout << "Discount: " << discount << endl;
     cout << "Discounted amount: " << amount << endl;
}
```

Figure 1.1 (Continued)

(a) Draw a Structure Chart to represent the design of the program above. Include all Data Flows. (8 marks)

(b) Write a Data Dictionary entry for an order as declared and used in the program above. (2 marks)

(c) Comment on whether the program above has a good design in terms of Cohesion. In your answer, give the definition of Cohesion as well. (4 marks)

(d) Comment on whether the program above has a good design in terms of Coupling. In your answer, give the definition of Coupling as well. (4 marks)

(e) The program above is to be rewritten using the object-oriented approach. Write Java code for an Order class based on the processing shown in the program above. Assume another class handles all input and output. (7 marks)

[Total : 25 marks]

Q2. Consider the partial Java class code shown in Figure 2.1 below.

```
public class Vehicle {

    private String regNumber;
    private String model;

    public Vehicle(String regNumber, String model) {
        this.regNumber = regNumber;
        this.model = model;
    }

    public String getRegNumber() {
        return regNumber;
    }

    public String getModel() {
        return model;
    }
}

public class VehicleList {

    private ArrayList<Vehicle> vehicles;

    public VehicleList() {
        // method details left out
    }

    public void addNew(Vehicle aVehicle) {
        // method details left out
    }

    public int getNumberOfVehicles() {
        // method details left out
    }

    public ArrayList<Vehicle> getVehicles() {
        // method details left out
    }
}

public class VehicleApp {

    private static Scanner scanner;

    public static void main(String[] args) {

        VehicleList list = new VehicleList();

        scanner = new Scanner(System.in);
```

Figure 2.1

Q2. (Continued)

```
        int choice;
        do {
            addNewVehicle(list);

            System.out.print("Enter 1(Continue) or 2(Stop): ");
            choice = scanner.nextInt();

        } while (choice == 1);

        displayAllVehicles(list);
    }

    public static void addNewVehicle(VehicleList list) {
        String regNumber;
        String model;
        Vehicle aVehicle;

        System.out.print("Enter reg. number: ");
        regNumber = scanner.next();
        System.out.print("Enter model: ");
        model = scanner.next();

        aVehicle = new Vehicle(regNumber, model);

        list.addNew(aVehicle);
    }

    public static void displayAllVehicles(VehicleList list) {
        ArrayList<Vehicle> listOfVehicles = list.getVehicles();
        int count = list.getNumberOfVehicles();
        Vehicle theVehicle;

        for (int i=0; i<count; i++) {
            theVehicle = listOfVehicles.get(i);
            System.out.println(theVehicle.getRegNumber());
        }
    }
}
```

Figure 2.1 (Continued)

Draw the following diagrams to correspond to the Java code given above:

(a) Design Class Diagram, showing all details given in the code. (7 marks)

(b) THREE (3) Sequence Diagrams (using 'ref' to connect them) for each of the methods in `VehicleApp` class, showing all details given in the code.

(18 marks)

[Total : 25 marks]

Q3. (a) Create the Relational Database Tables to represent the Classes and Associations in the Analysis Class Diagram (Figure 3.1) below. Clearly indicate the Primary Keys and Foreign Keys in the Tables. (11 marks)

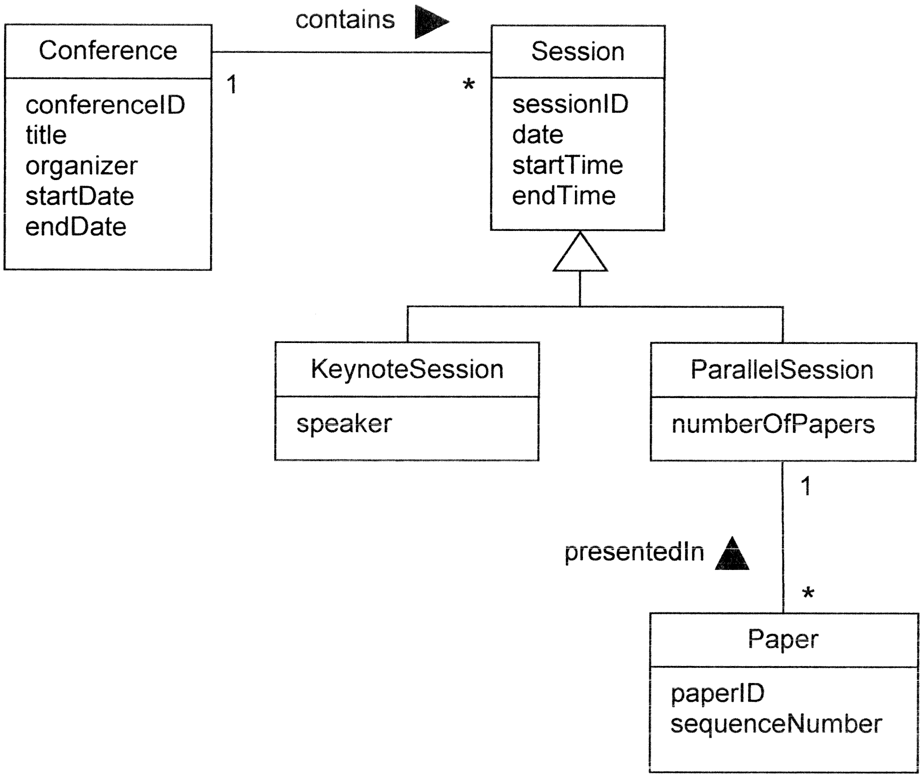


Figure 3.1

Note:
You may assume that conferenceID, sessionID, and paperID are unique for each conference, session, and paper respectively.

(b) Draw a Design Class Diagram based on the Analysis Class Diagram (Figure 3.2) below. Clearly indicate the Navigability of the Associations assuming that it is necessary to know:

- For an article, who the authors are.
- For an author, the articles written by the author.
- For an article, the author responsible for the article. (6 marks)

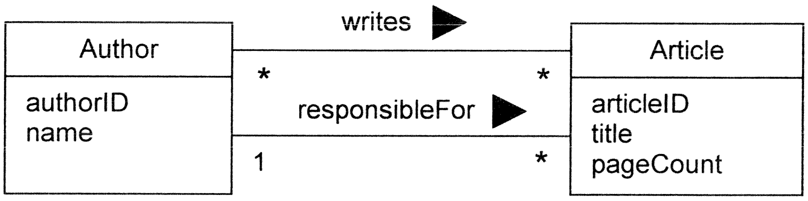


Figure 3.2

Q3(b).(Continued)

Note:
1. *You may assume any reasonable data types for the attributes.*
2. *You need NOT include the operations.*

(c) Consider the Java code (Figure 3.3) below. There is a dependency between the `EmployeeData` and `RelationalDBManager` classes. The Dependency Inversion Principle and Separate Interface from Implementation Principle can be applied to reduce this dependency.

```
public class EmployeeData {
    private RelationalDBManager dbmanager;

    public EmployeeData() {
         dbmanager = new RelationalDBManager();
    }

    public void save(Employee employee) {
         // other method details left out
         dbmanager.save(employee);
    }

    // other methods left out
}

public class RelationalDBManager{
    private SQLDatabase theDatabase;

    public void save(Employee employee) {
         // method details left out
    }

    // other methods left out
}
```

Figure 3.3

(i) Explain why this dependency should be reduced. (2 marks)

(ii) Rewrite the Java code to reduce the dependency. In your answer, explain how you applied the Dependency Inversion Principle and the Separate Interface from Implementation Principle. (6 marks)

Note:
You need NOT write complete code, only what is necessary to illustrate how the principles are applied.

[Total : 25 marks]

Section B: Answer ANY ONE question in this Section.

Q4. (a) When designing object-oriented systems, it is helpful to identify the following categories of objects involved in the system. Briefly describe these categories:

(i) Entity objects (1 mark)

(ii) Boundary objects (1 mark)

(iii) Control objects (1 mark)

(b) An object-oriented system may be designed using a three-layer architecture.

(i) Briefly describe the architecture, illustrating with a diagram. (4 marks)

(ii) Identify which layer the three categories of objects (refer to part (a) above) would be placed. (3 marks)

(c) Developers may utilise Design Patterns when designing object-oriented systems. Analyse the following situation and suggest a suitable Design Pattern.

> A charity organisation helps different types of homes such as elderly-care nursing home, etc.. The organisation has many patrons or financial supporters. These supporters may be individuals or companies. The organisation wants a software system to allow all supporters to be registered and to record the homes they are interested to support.
>
> The software system should also help the organisation to record the financial status of each home. Additionally, the software system should allow each supporter to see the list of homes they support and their financial status, which should be automatically updated when the organisation makes any changes to the home's financial status.

The main classes involved are Supporter, Home, and FinancialStatus. Draw a diagram to illustrate how the Design Pattern is applied and indicate the roles of the classes in relation to the Design Pattern.

Note:
You only need to give details based on what is described above and that are necessary to illustrate how the design pattern is applied.

(10 marks)

Q4. (Continued)

(d) Software Product Lines approach may be used when developing object-oriented systems. Analyse the situation below:

> A software company specialises in development of software for home security systems which they customise for each client. Currently, the system is controlled only by a device set up in the home.
>
> The company manager is planning to start a new project to develop a home security system that can be controlled using a mobile phone. The manager has recently heard about software product lines and she wants to know whether a software product line approach would be appropriate to use in the new project.

State whether or not you support the idea of using software product lines approach for the new project and justify your answer. (5 marks)

[Total : 25 marks]

Q5. (a) Analyse the following requirements for a private video sharing system and recommend a suitable Software Architecture. Briefly describe the design of the system based on the recommended architecture and illustrate with a diagram.

(i) Users want a system where the videos will be stored on one of the user's computer. All users will be able to upload their videos to this computer as well as to download any of the shared videos to their own computers. (5 marks)

(ii) Users want a system where each user will store their own videos in their own computers and all these videos will be made available for download by other users to their own computers. (5 marks)

(b) Analyse the Java code (Figure 5.1) below.

(i) Identify the Design Pattern that has been applied. (5 marks)

(ii) What situation does this Design Pattern generally address? (4 marks)

(iii) Draw a Class Diagram (showing only class names and relationships) to represent the code and identify the role the classes play with respect to the Design Pattern. (6 marks)

Q5(b).(Continued)

```
public interface IConnection {
     public void connect(String loginID, String LoginPassword);
     public boolean isAvailable();
}

public class WifiConnection implements IConnection {
    private String userID;
    private String password;

    public WifiConnection(String userID, String password){
        this.userID = userID;
        this.password = password;
    }

    @Override
    public void connect(String loginID, loginPassword) {
        if (loginID.equals(userID) &&
                   loginPassword.equals(password) {
            // other method details left out
            System.out.println("Connected via Wifi network.");
    }

    @Override
    public boolean isAvailable() {
        // method details left out
    }

    // other methods left out
}

public class WiredConnection implements IConnection {
    private String userID;
    private String password;

    public WifiConnection(String userID, String password){
        this.userID = userID;
        this.password = password;
    }

    @Override
    public void connect(String loginID, loginPassword) {
        if (loginID.equals(userID) &&
                   loginPassword.equals(password) {
           // other method details left out
           System.out.println("Connected via Wired network.");
    }
```

Figure 5.1

Q5(b). (Continued)

```
    @Override
    public boolean isAvailable() {
        // method details left out
    }

    // other methods left out
}

public class Network {

    // other methods left out

    public void connectToNetwork(IConnection connection){
        connection.connect();
    }
}


public class NetworkTest {
    public static void main(String [] args) {
        Scanner scanner = new Scanner(System.in);
        Network network = new Network();
        System.out.print("Enter user id: ");
        String id = scanner.next();
        System.out.print("Enter password: ");
        String pwd = scanner.next();

        WiredConnection wired = new WiredConnection(id, pwd);
        if (wired.isAvailable()) {
            network.connectToNetwork(wired);
        } else {
            WifiConnection wifi = new WifiConnection(id, pwd);
            if (wifi.isAvailable()) {
                network.connectToNetwork(wifi);
            }
        } else {
            System.out.println("Cannot connect to network");
        }
    }
}
```

Figure 5.1 (Continued)

[Total : 25 marks]