Sri Lanka Institute of Information Technology

B.Sc. Honours Degree in Information Technology

Final Examination
Year 2, Semester 2 (2018)

IT2020 – Software Engineering

Duration: 3 Hours

October 2018

Instructions to Candidates:

- This paper has 5 questions.
- Answer all questions in the booklet given.
- The total marks for the paper is 100.
- This paper contains 7 pages, including the cover page.
- Electronic devices capable of storing and retrieving text, including calculators and mobile phones are not allowed.

Question 1 (20 marks)

The following is a use case scenario for "receive product" use case triggered by a user when a shipment arrives. Read the use case scenario and answer the subsequent questions.

Use Case Name: Receive Product
Primary Actor: User
Main Steps:
1. The user enters the shipper identification number (ShipperID). 2. The *ShipperID* is validated by the system. 3. System displays the shipper details on screen. 4. User can get the product details by entering shipment details. (eg: date, sender, number of pieces) 5. For each product in the shipment user finds the purchase order by providing productID. 6. System displays the purchase order on screen. 7. User check off the item and quantity on the purchase order and notify the accounts payable of the updated purchase order. 8. System displays the "Account Payable Notified" message. 9. The system updates inventory for the received products by storing data in database. 10. The entire process will be repeated for all the received products.
Extensions:
2a. If the ShipperID is not found, system displays the "shipper not found" error message and prompt the user to renter the ID.

a) When the lead software engineer analyzed all the use case scenarios for this System, he found out that to execute each of the use case scenarios, the respective user must always login to the system including the above scenario.

What concepts in UML 2.0 can you apply to reuse the already designed login sequence? (1 mark)

b) Applying the concepts you have mentioned in part (i) and using suitable classes, draw the **sequence diagram** for the above use case scenario with stereotypes.

Hint: You may use one boundary class, one controller class and many entity classes.

(19 marks)

Question 2 (20 marks)

Given below is a detailed description of a Hotel Room Reservation System (HRRS). Model a **physical diagram** for the HRRS.

HRRS user interfaces are developed through Java Server Pages (JSP) and are installed in the main web server. Smart phone or desktop PC users can access the HRRS using the browser. JSP UI component is responsible for creating HRRS interface, this interface is needed for the mobile and desktop devices to function properly. The database of HRRS is a Mongo database which runs in a separate database server called HRRS_DB. It has a MongoDB Driver installed which runs the Mongo database.

The backend application of HRRS is named B_Service which is implemented in Java and deployed in a Solaris server named HRRS_Middleware. The Reservation module which is a core component of B_Service application requires JDBC interface which is implemented by Mongo database. The B_Service application consists of Reservation, Payment and Administration modules. JSP communicates with the Reservation class through the iReserve interface, Payment class through iPay interface and Administration class through the iAdmin interface implemented by the respective core modules. Administration module implements iBook interface which is accessed by Reservation module and Payment module accesses the iReserve developed by Reservation module.

All the servers are connected through a LAN. Both desktop and mobile devices connect to the web server via http over internet.

Question 3 (20 marks)

a) What is **Statement coverage** and **Branch coverage** in White box testing? (4 marks)

b) How many minimum test cases are required to achieve 100% statement coverage for the code given below? Show the statement coverage as a percentage for each test case.

(8 marks)

```
public static void passTheCourse(int success) {
       switch (success) {
          case 0:
              System.out.println("No result");
              break;
          case 1:
              System.out.println("Final result: Fail");
              break;
          case 2:
              System.out.println("Final result: Success");
              break;
          default:
              System.out.println("Unknown result");
              break;
               }

 }
```

c) Based on the below given code come up with a control flow graph and then calculate the minimum number of test cases required for full branch coverage. (Show the branch coverage as a percentage for each test case)

(8 marks)

```
int getLargestnumber (int n1, int n2, int n3) {
     if (n1 >= n2) {
            if (n1 >= n3) {
                largestNumber = n1;
            }
           else {
                largestNumber = n3;
            }
            }
           else {
            if (n2 >= n3) {
                largestNumber = n2;
            }
           else {
                largestNumber = n3;
            }
        }
     return largestNumber;
}
```

Question 4 (25 marks)

a) A hotel has a hotel keeper. There are three restaurants inside the hotel such as Sri Lankan restaurant, Indian restaurant and Thai restaurant. The clients need to know different menus in different restaurants, as the hotel keeper has the details of the menus, the clients contact him.

What is the most suitable design pattern that can be used in the above scenario?

(2 marks)

Draw the class diagram for the selected **design pattern** including appropriate methods which can be used for the given scenario.

(7 marks)

b) A software developer, Kevin, worked on an e-commerce website. The website allowed users to shop and pay online. The site was integrated with a 3rd party payment gateway, through which users can pay their bills using their credit card. Everything was going well, until his manager asked him to change the vendor of the payment gateway.
Earlier, the site was attached to the "EasyPay" payment gateway which takes an EasyPay type of objects. The new vendor, "QuickPay", only allows the QuickPay type of objects to allow the process. Kevin does not want to change the whole set of classes which have references to objects of type EasyPay. This also raises the risk on the project, which is already running on the production. Neither can he change the 3rd party tool of the payment gateway. The problem has occurred due to the incompatible interfaces between the two different parts of the code. In order to get the process working, Kevin needs to find a way to make the code compatible with the new vendor's provided API. So he decided to use a Design Pattern as a solution.

Suggest the most suitable **design pattern** that can be used in the above scenario?

(2 marks)

Draw the class diagram for the selected design pattern including appropriate methods which can be used for the given scenario.

(7 marks)

c) Consider the code given bellow and answer the questions.

```
package com.concretepage;
import java.util.Observable;
import java.util.Observer;

public class FirstNewsReader implements Observer {
    public void update(Observable obj, Object arg) {
        System.out.println("FirstNewsReader got The news:"+(String)arg);
    }
}

package com.concretepage;
import java.util.Observable;
import java.util.Observer;
public class SecondNewsReader implements Observer {
    public void update(Observable obj, Object arg) {
        System.out.println("SecondNewsReader got The news:"+(String)arg);
    }
}

package com.concretepage;
import java.util.Observable;
public class News extends Observable {
    void broadcast() {
        String[] news = {"News 1", "News 2", "News 3"};
        for(String s: news){
            //set change
            setChanged();
            //notify observers for change
            notifyObservers(s);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                System.out.println("Error Occurred.");
            }
        }
    }
}
package com.concretepage;
public class ObserverExample {
    public static void main(String args[]) {
        News observedNews = new News();
        FirstNewsReader reader1 = new FirstNewsReader();
        SecondNewsReader reader2 = new SecondNewsReader();
        observedNews.addObserver(reader1);
        observedNews.addObserver(reader2);
        observedNews.broadcast();
    }
}
```

i) Identify the design pattern used in this code.

(1 mark)

ii) Draw the class structure of the design pattern you identified in part i) with appropriate methods for the above scenario. (6 marks)

Question 5 (15 marks)

a) What is the main difference between a state machine diagram and a sequence diagram considering the objects involved? (2 marks)

b) What are the internal activities/behaviors associated with a state chart diagram and briefly explain them with suitable examples. (3 marks)

c) Compare and contrast "**Centralized version control**" and "**Distributed version control**". (3 marks)

d) What does it mean by **Incident management**? (2 marks)

e) Briefly explain the process of incident response. (5 marks)

---------- End of the Paper ----------