Kurdistan Regional Government
Ministry of Higher Education & Scientific Research
University of Salahddin – Hawler
College of Engineering
Software Engineering Dept.

Final Examinations (2011/2012)
Subject: Systems Analysis & Design
Second Year Students
Time allowed: 3 hours
Lecturer: Amanj Sherwany

The highest obtainable mark is 100, the minimum passing mark is 50

Q1/ A: (10 points)

Supporting (maintaining) a legacy software system is very difficult, explain why.

Answer

There are a lot of reasons behind that, mainly:

1. The hardwares used in these legacy systems might be obsolete.
2. The software might be unstructured and wont suit with the current needs, it also might be written in a programming language that is obsolete.
3. The data might be incomplete or inconsistent, which might be in a proprietary format.
4. The software and the data might be undocumented.

B: (15 points)

Giving reasons for your answer, based on the type of system being developed, suggest the most appropriate generic software process model that might be used as a basis for managing the development of the following systems:

- A system to control patients' blood pressure.
- An online student portal, which has a built-in presentation (powerpoint-like) program.
- An application marketplace (like, Apple Appstore, Google Play Store and Chrome Web Store).

Answer

- Waterfall model, since this is a critical system (humans life depend on it)
- Maybe the best would be Component-Based Software model, as one can re-use an already developed presentation software.
- An incremental delivery would be the best, since the application can be lunched with a limited functionality with adding more functionalities in the next releases.

* * *

Q2: (25 points)

The following is an excerpt taken from a *Software Requirement Specification* (SRS) document:

> "A new registered user should have a limited set of functionalities. The web application should not

depend on any proprietary plug-in. Loading the web application should be blazingly fast. The application should have a way to send messages to friends. Its background colour should be customizable."

In the above paragraph:

- Find two ambiguous requirements. **(10 points)**
- Extract the possible functional and non-functional requirements. **(15 points)**

Answer

Ambiguous requirements

- A new registered user should have a limited set of functionalities.
- Loading the web application should be blazingly fast.

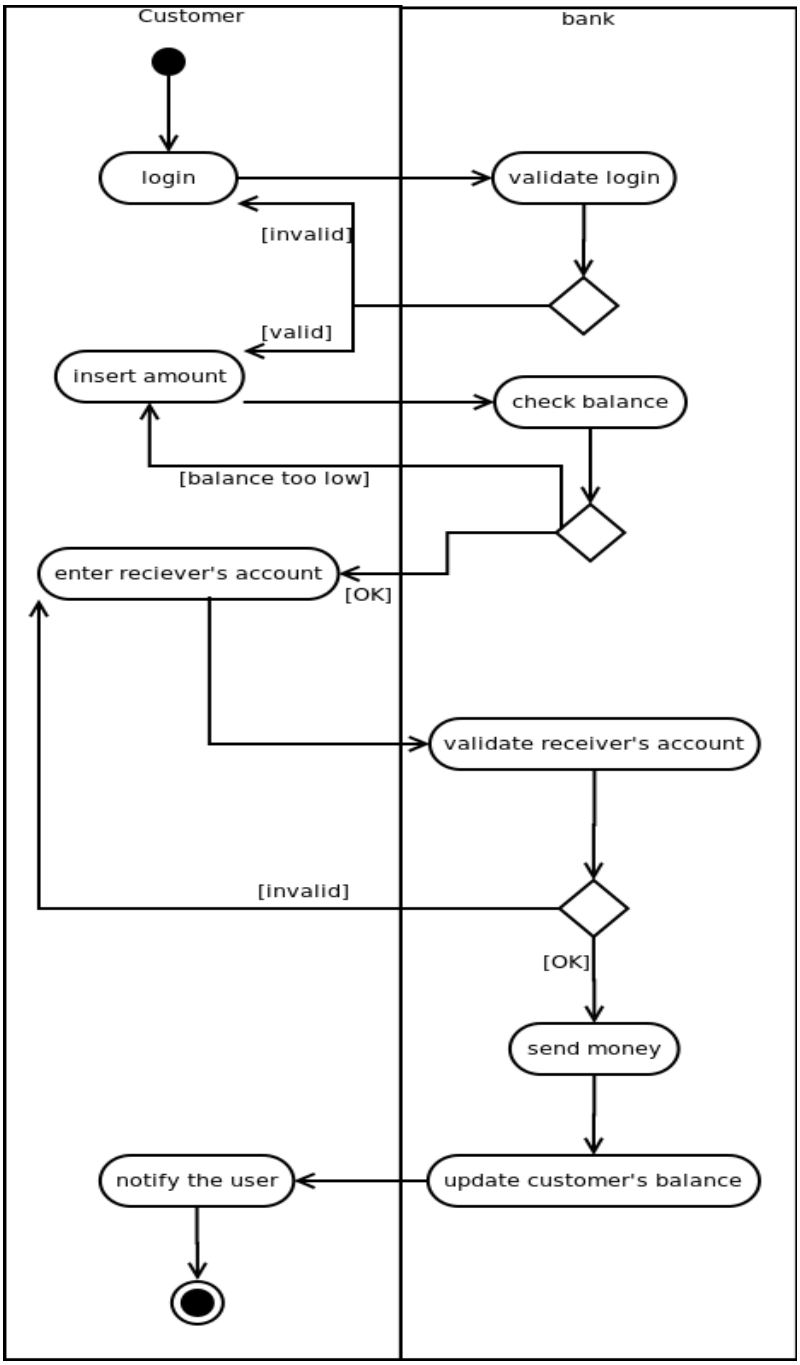
Functional and Non-functional requirements:

- Functional Requirements
  - The application should have a way to send messages to friends.
  - Its background colour should be customizable.
  - A new registered user should have a limited set of functionalities.
- Non-functional Requirements
  - Loading the web application should be blazingly fast.
  - The web application should not depend on any proprietary plug-in.

* * *\

Q3: (20 points)

Almost any bank supports transferring money from an account to another account, draw the activity diagram for this scenario.



Q4/ A: (15 points)

You are developing an application for an aircraft company, users can purchase 2nd class tickets or business class tickets. A business class ticket has all the advantages of 2nd class tickets, plus some extra features. What is the best design pattern to use here? Show it in code.

Answer

Decorator design patter is the pattern to use here:

```
public class Trip{}

public class BusinessClass{
     private Trip trip;
     public BusinessClass(Trip trip){
          this.trip = trip;
     }
}
```

B: (15 points)

The following interface violates one of the SOLID principles, identify it and re-write the code to obey the principle.

```
class PriceCalculator{
     public double totalPrice(Part[] parts){
          double total = 0.0;
          for(int i=0; i < parts.length; i++){
               if(parts[i] instanceof Motherboard)
                    total += (1.45 * parts[i].getPrice());
               else if(parts[i] instanceof Memory)
                    total += (1.27 * parts[i].getPrice());
               else
                    total += parts[i].getPrice();
          }
          return total;
     }
}
```

Answer

The above code violates the Open/Closed Principle, and the fix will be something like:

```
public class Part{
     private double price;
     public Part(double price) {this.price = price;}
     public double getPrice() {return price;}
```

```
}

public class ConcretePart extends Part{
     public double getPrice(){
          return (0.90 * price);
     }
}
```

Good Luck