Closed Book Examination

COMP37321

Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Modern Software Engineering Practice

Tuesday 20th January 2009
14:00 – 16:00

Please answer any THREE Questions from the FIVE questions provided

The use of electronic calculators is NOT permitted
1. a) i) Explain, briefly, what is meant by the terms *iterative development* and *incremental development*. (4 Marks)

ii) State, briefly, the kind of project and software development team that each of the approaches in Q1)a) i) is best suited to. (4 Marks)

b) A stamp machine accepts only 5p and 10p coins via its coin slot and dispenses only 20p stamps via its dispensing outlet (together with any change). The machine’s display indicates the total amount of money inserted so far. Once 20p or more is inserted, a 20p stamp is automatically dispensed together with any change. The display only shows the amounts 0p, 5p, 10p, 15p and 20p. If, for example, a 5p and a 10p have been inserted, the display indicates 15p. If the user then inserts another 10p, the stamp dispenser dispenses a 20p stamp, returns 5p and changes the display to show 0p. The machine also has a coin return button which, when pressed, causes the machine to return the amount of money indicated by its display and which sets the display to indicate 0p.

Draw a state transition diagram to represent the *actions* taken by the stamp dispenser and the *state changes* that result from the dispenser being used by customers.

Hint: The states that the machine may take include *HAS_0*, *HAS_5*, etc. (8 Marks)

c) What do RUP and eXtreme programming have in common and how do they fundamentally differ? (4 Marks)
2. a) Draw a diagram to show the processes involved in managing a PRINCE2 project and how those processes link with each other to create the usual content of a PRINCE2 project. (8 Marks)

b) Construct a *Use Case diagram* for a conventional domestic refrigerator that is the subject of a repair contact with the fridge’s manufacturer.

The diagram should depict a *user* who knows how to operate the fridge (i.e. an *actor* associated to the use case). To operate the fridge various sub-tasks have to be carried out (i.e. the use case will include other named use cases). These sub-tasks include at least the opening and closing of the fridge, and changing the fridge’s temperature setting. The repair person is based on the user (i.e. the repair person actor is a *generalization* of the user actor). Repairing the fridge is similar to operating the fridge except that additional functionality is needed (i.e. the repair use case *extends* the operates use case) and involves at least the subtask of inspecting the access log maintained for the fridge. (8 Marks)

c) What two elements do standards for software management plans explicitly NOT contain? (4 Marks)
3. a) The class diagram below depicts the relationships between the classes $X, Y, Z, P$ and $Q$.

Java implementations of the classes $X, Y, Z$ and $P$ are shown below.

```java
public class X {
    public String m1() {
        return "X.m1()";
    }
    public String m2(String s) {
        return "X.m2( " + s + " )";
    }
}

interface Y {
    String m2(String s);
    String m3();
}

public class Z extends X {
}

public class P {
    +m1(): string
    +m2(s: String): String
    +m3(): String
}

public class Q {
    +m1(): string
    +m3(): String
}
```

(Question 3 continues on the following page)
(Question 3 continues from the previous page)

```
implments Y
{
    public String m1()
    {
        return "Z.m1()";
    }
    public String m3()
    {
        return "Z.m3()";
    }
}

public class P
implments Y
{
    public String m1()
    {
        return "P.m1()";
    }
    public String m2(String s)
    {
        return "P.m2(" + s + ")";
    }
    public String m3()
    {
        return "P.m3()";
    }
}

public class Q
extends Z
{
    public String m2(String s)
    {
        return "Q.m2(" + s + ")";
    }
}
```

Given the declaration:-

```
Q q = new Q();
```

i) Explain, briefly, which class’s implementation code the method \textit{m1()} in the \textit{Q} object is mapped to. (2 marks)

ii) Explain, briefly, which method, and in which class, the implementation code from i) above overrides. (2 marks)

iii) Explain briefly, if it \textbf{is} or \textbf{is not} possible for a \textit{Q} reference variable to access the overridden \textit{m1()} implementation in the class \textit{X}. (2 marks)

iv) Explain, briefly, if it \textbf{is} or \textbf{is not} possible for the actual implementation code in class \textit{Z} to use the \textit{X} class implementation. (2 marks)

(Question 3 continues on the following page)
(Question 3 continues from the previous page)

b) Develop a BNF grammar to describe the sequences of keystrokes made by an infallible (i.e. never makes mistakes) user of the label maker whose interface is shown below.

A perfect user always creates at least one label by pressing the On key, making the text for the label visible in the text window, pressing the Print key and finally the Off key.

Making the text for the label visible in the text window involves either entering text using the text keys or retrieving a stored text using the Recall key and arrow keys to scroll through stored texts.

The text currently in the text window can be stored by pressing the Store key.

The machine is supplied with stored texts for all modules in the School of Computer Science, University of Manchester for academic year 2008-2009.

Ensure that the resulting grammar is systematically refined such that it is possible to identify each step in the refinement process. (8 Marks)

c) Why, when developing a game playing program for Noughts and Crosses, is it important to separate concrete from abstract syntax? (4 Marks)
4. a) Consider the following statement:-

“In the context of object-orientation reuse is by replacement”.

i) Explain, briefly, how replacement in pre-OO (i.e. imperative-procedural, possibly also modular) languages differs from that in object-oriented systems.  

(2 marks)

ii) Explain, briefly, what constraint applies to an object replacing another object.  

(2 marks)

iii) Explain the two forms of method replacement.  

(2 marks)

iv) Explain why the concept of self is required.  

(2 marks)

b) Given the definition of the abstract data type stack in the equational specification below:-

ADT-MODULE stack_definition;

INTERFACE

USES integer, Boolean;

FUNCTION empty: stack;
FUNCTION is_empty(s: stack): Boolean;
FUNCTION push(i: integer; s: stack): stack;
FUNCTION pop(s: stack): stack;
FUNCTION top(s: stack): integer;

SPECIFICATION

VAR i: integer;
    S: stack;

EQUATIONS

    is_empty(empty) = true;            //axiom 1
    is_empty(push(i, s)) = false;      //axiom 2
    top(push(i, s)) = i;               //axiom 3
    pop(push(i, s)) = s;               //axiom 4

END.
Evaluate the following stack expression:-

```
top(pop(push(top(push(k, push(top(push(j, push(i, empty)))))), empty
        
    ),
    
    ),
push(l,
    push(m, pop(push(n, empty)))

```

(8 marks)

c) Explain, briefly, two limitations of a module construct. (4 Marks)
5.  

a)  Explain, briefly and with reference to simple examples, what is meant by the terms *ad-hoc polymorphism* and *universal polymorphism* in the context of class-based object-oriented programming languages.  

b)  The definition of the type *time* and two variables of type *time*, *arrival_time* and *departure_time*, in an imperative-procedural language is shown below.

```plaintext
TYPE  time = RECORD
    hour: 0..23;
    mins: 0..59;
    secs: 0..59
END;

VAR arrival_time, departure_time: time; // variables of type time
```

i)  Construct an “equivalent” Java class called *time* together with two suitably initialised variables named *arrivalTime* and *departureTime*.

ii) Convert your answer from Q5)b)i) above into an implementation that exploits Java’s support for *generic classes*.

b)  What two fundamental advantages do Java’s “generics” provide?