BINDURA UNIVERSITY OF SCIENCE EDUCATION
FACULTY OF SCIENCE AND ENGINEERING
DEPARTMENT COMPUTER SCIENCE
BSc HONS DEGREE IN COMPUTER SCIENCE/INFORMATION TECHNOLOGY

CS203/_SWE211: OBJECT ORIENTED PROGRAMMING II

DURATION:  3 HOURS                            TOTAL MARKS: 100

INSTRUCTIONS TO CANDIDATES

Answer all questions.
The paper consists of Section A (Theory) and Section B (Practical)
Section A carries 40 marks
Section B carries 60 marks
All programs to be written in Java

## Section A: Theory

a) Describe the Java naming conventions for naming classes, interfaces, methods and variables. **[12]**

b) Using sample code, demonstrate the different ways of creating java threads. **[6]**

c) The method start() in class Thread will place a new thread in a pool of threads that are 'ready' to run under time-slicing. The Java interpreter will select a thread from this pool to 'run' (i.e., execute the thread). 'Ready' and 'running' are two of the possible states that threads may be in, and threads move between these states under the control of the Java interpreter's time-slicing mechanism. What other states can threads be in, and how do they change from one state to another? **[8]**

d) Explain any two ways of dealing with thread deadlocks. **[4]**

## Section B: Practical

Create a folder on the desktop and name it using your registration number and course code. Save all you work in this folder.

## Question 3

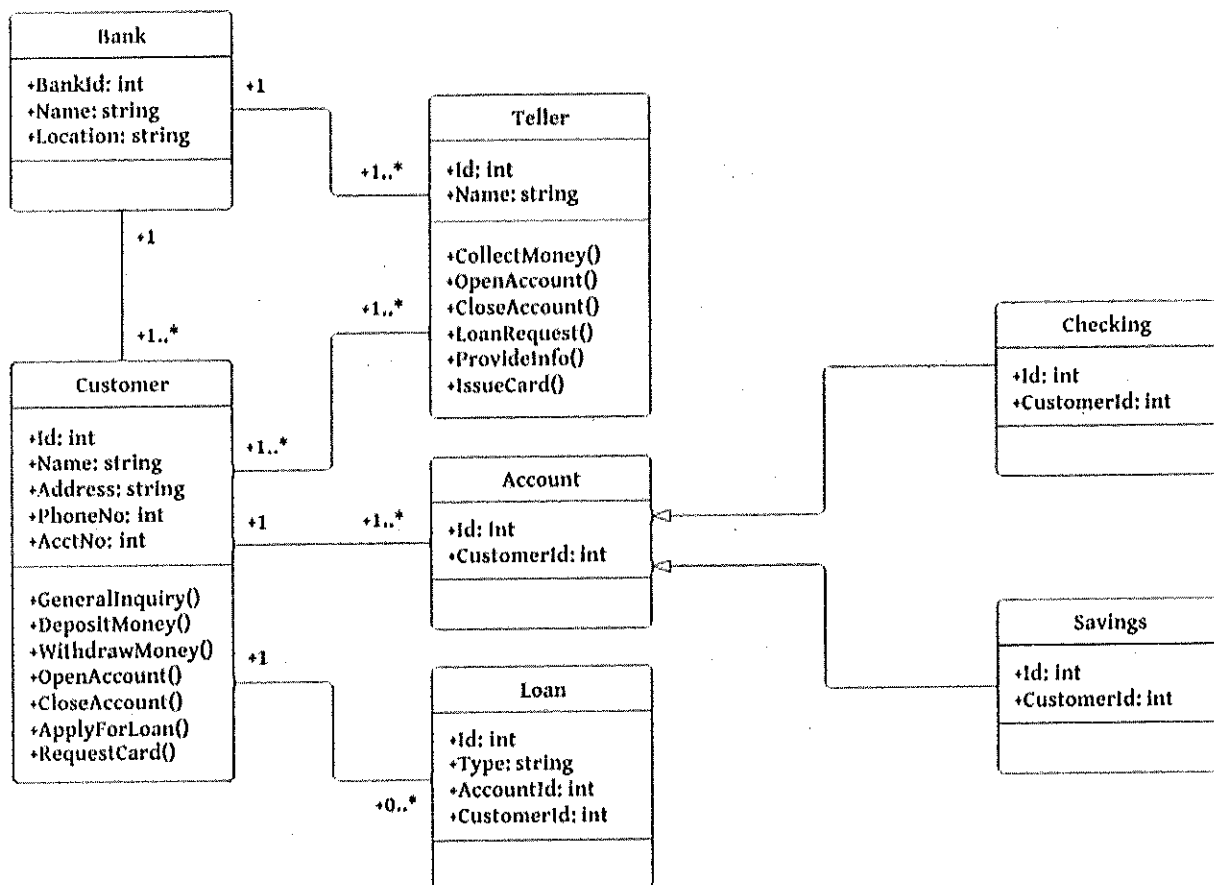The following class diagram in figure 1 represents a partial design for a major bank.



**Figure 1: UML diagram for a bank**

Write Java versions of the all the classes in the diagram, implementing the variables, methods and relationships shown in figure 1. Include a suitable constructor in each class. [20]

## Question 4

Write a complete Java program which contains a static method named **countCoins** that accepts as its parameter a Scanner for an input file whose data represents a person's money grouped into stacks of coins. Your method should add up the cash values of all the coins and print the total money at the end. The input consists of a series of pairs of tokens, where each pair begins with an integer and is followed by the type of coin, which will be either "pennies" (1 cent each), "nickels" (5 cents each), "dimes" (10 cents each), or "quarters" (25 cents each), case-insensitively. A given coin might appear more than once on the same line.

For example, if the input file contains the following text:

**3 pennies 2 quarters 1 pennies 3 nickels 4 dimes**

3 pennies are worth 3 cents, and 2 quarters are worth 50 cents, and 1 penny is worth 1 cent, and 3 nickels are worth 15 cents, and 4 dimes are worth 40 cents. The total of these is 1 dollar and 9 cents, therefore your method would produce the following output if passed this input data. Notice that it says 09 for 9 cents.

**Total money: $1.09**

You may assume that the file contains exactly 5 pairs of tokens. You may also assume that the input is valid; that the input has an even number of tokens, that every other token is an integer, and that the others are valid coin types.

You may assume that the input file exists and has the format described above. Do not make any assumptions about the length of the file, but you can assume that it is formatted correctly.                    **[25]**

## Question 5

Write a complete java program that contains a static method named "locationOfLargest" that takes as argument a LinkedList of Integer values. The method should return as its value the subscript of the cell containing the largest of the values in the LinkedList. Thus, for example, if the LinkedList that is passed to the method looks like the one in figure 2:

```
 0     1     2     3     4
58  |  26 |  90 |  34 |  71
```

**Figure 2: Linked list**

then the method should return the integer 2 as its value. If there is more than one cell containing the largest of the values in the LinkedList, then the method should return all the subscripts of the cells containing the largest values. For example, if the LinkedList that is passed to the method is as shown in figure 2.1

```
 0     1     2     3     4     5     6
58  |  26 |  91 |  34 |  70 |  91 |  88
```

**Figure 2.1: Linked list**

then the largest value occurs in cells 2 and 5, so the method should return the integer value 2 and 5.

You may assume that the list passed is not null. You may use any other generic collections from the Java Collections Framework to help come up with your solution. Do not make any assumptions about the length of the LinkedList or the number of the largest values that can be found in a LinkedList. The passed LinkedList should not be mutated.                                                    [25]

### END OF PAPER