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 30 marks
Section B carries 70 marks
All programs to be written in Java

Section A: Theory

a) 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? **(12 marks)**

b) The Java keyword `synchronized` can be applied to a method or a block of code.

i. The keyword `synchronized` was introduced to Java to allow programmers to solve a particular kind of problem that may arise in multi-threaded computation. Briefly describe what this problem is, and how it might arise. **(6 marks)**

ii. Describe how `synchronized` solves this problem. **(6 marks)**

iii. Using `synchronized` may lead to *deadlock*. Briefly describe what is meant by deadlock, and describe how it may be avoided. **(6 marks)**

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

In this question, your task is to design a Java class, called Book, with the following requirements:

- Every book has ISBN, called isbn, with the type of String.
- Every book has a title, called title, with the type of String.
- Every book has a list of authors, called authors, with the type of ArrayList of String.
- Every book has an edition number, called edition, with the type of integer.
- Every book has a number of pages, called pages, with the type of integer.
- Every book has a price, called price, with the type of double.
- Provide the default **constructor** for the class, which only initializes the list of the authors with an empty ArrayList, and also initializes the title of the book with "No Title Yet".
- Provide a second **constructor** for the class that initializes the isbn, title and edition with
- three corresponding parameters, as well as the empty ArrayList of the authors.
- For every property, isbn, title, edition, pages, and price, create public getter (accessor) and setter (mutator) methods.
- Create a public method, addAuthor, to add one author to a book.
- Create a public method, removeAuthor, to remove one author from the list of authors of a book. This method should receive the name of an author to remove it from the list of the current authors, if any.
- Create a public method, changeAuthor, to change one author in the list of authors of a book. This method should receive two parameters, the name of one current author, and the new name. To find the index of an object inside an

ArrayList, you can call the indexOf method, which has the object as its parameter. If the object does not exist in the list, the method returns -1.

- Override the toString method to represent a book, with its isbn, title, list of authors, edition number, number of pages, and price.

Completely develop the Book class using the above description. **(20 marks)**

Question 4

Write a complete Java program that contains a static method named mostUnique that accepts as its parameter a Scanner for an input file. The data in the Scanner represents integer quiz scores separated by spaces. Each class period is on its own line and contains a different number of students. Your method should return the highest number of unique scores given in a single class period. The method should also print the number of unique scores given in each period. On a given line, repeated scores are always next to each other. For example, given a Scanner named input referring to an input file that contains the following text in figure 1:

```
10 10 10 9 9 8 3
3 3 8 10 9 7 7 6 6
4 1 9 9 10 7 7
10 10 10 10
```

Figure 1: Text from file

The call on mostUnique(input) should return 6 and generate the following output:

Period 1: 4 unique scores
Period 2: 6 unique scores
Period 3: 5 unique scores
Period 4: 1 unique scores

On the first line, there are 4 unique scores: 10, 9, 8 and 3. The second line contains 6 unique scores: 3, 8, 10, 9, 7 and 6. The third line contains 5 unique scores: 4, 1, 9, 10

and 7. The fourth line only has one unique score: 10. The value returned is 6 because it is the highest number of unique scores given in a class period.

Assume that the file exists, that it contains at least one line of data and that each line contains at least one score, but not make any assumptions about the size of the file. You may use any other data structures from the Java Collections Framework to simplify your work. **(25 marks)**

Question 5

Write complete Java program that contains a static method named **evenBeforeOdd** that accepts a LinkedList of Integers as a parameter and rearranges its elements so that all even values appear before all odds. For example, if the following LinkedList is passed to your method:

LinkedList numbers = {5, **2, 4**, 9, 3, **6, 2**, 1, 11, 1, **10, 4**, 7, 3};

Then after the method has been called, **an example of one acceptable ordering** of the elements would be:

{**4, 2, 4, 10, 2, 6**, 3, 1, 11, 1, 9, 5, 7, 3}

The exact order of the elements does not matter, so long as all even values appear before all odd values. For example, the following would also be an acceptable ordering:

{**2, 2, 4, 4, 6, 10**, 1, 1, 3, 3, 5, 7, 9, 11}

Do not make any assumptions about the length of the LinkedList or the range of values it might contain. For example, the LinkedList might contain no even elements or no odd elements. You may assume that the LinkedList is not null.

You may not use any temporary LinkedLists **to help you solve this problem.** You **also may not use any other data structures** from the Java Collections Framework.

(25 marks)

END OF PAPER