COMP 103 : Test 1 **Page 1 of 12**

Family Name: . . . . . .

First Name: . . . . . .

Student ID: . . . . . .

COMP 103 : Test 1

September 6, 2023

Test Instructions

- **Time Limit:** 60 Minutes
- Write your **Full Name** and **Student ID** at the top of the **first page** of the exam paper. For all subsequent pages, include your **Student ID** at the top.
- **Attempt all questions** in the exam paper; the total marks available are 50.
- Write your answers on the provided test paper in their desinated spaces.
- Upon completing the test, please ensure that you hand in **all sheets** of the test paper.
- If you encounter a question that appears unclear, feel free to request **clarification** from the invigilator.
- A concise **Java documentation for Collections** is made available with the test to assist you during the test.
- The marks obtained in this test will contribute **20%** towards your final grade for the course.
- You may use **dictionaries** during the exam. Computer, phone and other smart devices **are not allowed**.
- Ensure that your final answers are presented clearly.
- You can assume that **all libraries required for programs are imported**.

Question	Max mark	Stu. Mark
1. Properties of Collections	**9**	
2. Using Collections	**8**	
3. More on using Collections	**17**	
4. Cost of Algorithms	**10**	
5. Recursion	**6**	
TOTAL	**50**	

COMP 103 : Test 1 **Page 2 of 12** **Student ID:** . . . . . .

Question 1. Properties of Collections [9 marks]

For these questions, **circle** the right answer from the list.

1.a) [3 marks]

You need to store a list of student names and their corresponding IDs in a way that allows for fast retrieval of a student's ID given their name. Which collection would you use for this purpose?

1. ArrayList 2. HashMap 3. TreeSet 4. LinkedList

In [1]:
```
// Answer: HashMap
```

1.b) [3 marks]

In a game application, you want to maintain a list of players with their scores in descending order, where the highest score comes first. Additionally, you need to efficiently retrieve and remove the highest-scoring player. Which collection would you use for this purpose?

1. HashSet 2. PriorityQueue 3. TreeMap 4. LinkedHashSet

In [2]:
```
// Answer: PriorityQueue
```

1.c) [3 marks]

You are building a spell-checking feature for a text editor. You need to store a dictionary of words in a way that allows for efficient lookup of words (i.e., check words are contained in the dictionary) and doesn't store duplicate words. Which collection would you use for this purpose?

1. ArrayDeque 2. HashMap 3. HashSet 4. TreeMap

In [3]:
```
// Answer: HashSet
```

Question 2. Using Collections [8 marks]

Complete the method `filterDuplicates` that takes a `List<String>` as input and returns a `List<String>` containing only the unique elements from the input list, with the following two properties:

1. maintains the original order of appearance of elements in the input list, and
2. ignores the case of letters when checking for uniqueness. For example, `red` and `Red` are considered the same.

Each element in `inputList` contains only one word. For tracking whether an element has been encountered before, you must use a `Set`. Otherwise, you will loose some marks.

```
public class FilterDuplicatesSol {

    public static List<String> filterDuplicates(List<String> inputList) {
        List<String> resultList = new ArrayList<>();
        Set<String> seenWords = new HashSet<>();

        for (String word : inputList) {
            String lowerCaseWord = word.toLowerCase();
            if (!seenWords.contains(lowerCaseWord)) {
                resultList.add(word);
                seenWords.add(lowerCaseWord);
            }
        }
        return resultList;
    }

    public static void main(String[] args) {
        List<String> words = Arrays.asList(
            "apple", "Banana", "APPLE", "Grape", "banana", "Pear");
        List<String> uniqueWords = filterDuplicates(words);
        System.out.println("Unique Words (ignoring case): "
                           + uniqueWords);
    }
}
```

Question 3. More on using Collections [17 marks]

You are working on a project to manage a library's book collection. The library wants to enhance its system by incorporating sorting and mapping functionalities for books. You have a class named `Book` that needs to be modified to implement the `Comparable` interface for natural ordering based on book titles and their publication years. Additionally, you have a class named `LibraryBookManager` to be modified by adding various methods.

3.a) For the `Book` class: [6 marks]

1. Modify the `Book` class to implement the `Comparable` interface. The natural order should arrange books in ascending alphabetical order of titles, and in cases where titles match, it should then sort them in ascending order of publication years, from the oldest to the newest.
2. Add a `toString()` method to the `Book` class to print the title, author and publication year of the book.

Provide your code solution for each task in the given spaces and wherever it is required on page 5:

```java
class Book implements Comparable<Book> {
    private String title;
    private String author;
    private int publicationYear;
    // Constructor
    public Book(String title, String author, int publicationYear) {
        this.title = title;
        this.author = author;
        this.publicationYear = publicationYear;
    }
    public String getTitle() {
        return title;
    }
    public String getAuthor() {
        return author;
    }
    public int getPublicationYear() {
        return publicationYear;
    }
    //compareTo method
    public int compareTo(Book otherBook) {
        int titleComparison = this.title.compareTo(otherBook.getTitle());
        if (titleComparison == 0) {
            return Integer.compare(this.publicationYear,
otherBook.getPublicationYear());
            // or: return this.publicationYear - otherBook.getPublicationYear() //
        }
        return titleComparison;
    }
    //toString method
    public String toString() {
        return "Title: " + title + ", Author: " + author + ", Publication Year: "
+ publicationYear;
    }
}
```

3.b) For the `LibraryBookManager` class: [11 marks]

- Add the following two fields:
  - an ArrayList named `sortedBooks` to store a collection of `Book` objects.
  - a TreeMap named `bookAuthorMap` to associate book titles with their respective authors.
- Complete the `addBook` method which adds `Book` objects to the ArrayList, *ensuring that they are sorted based on their natural ordering*. Enforce natural ordering after adding a `Book`. Do not worry about big-O cost.
- Complete `addBooksToMap` method which adds books to the TreeMap `bookAuthorMap`.
- Complete `printBookAuthors` to print `bookAuthorMap` elemetns in the following format `book title -> book author`
- Complete `printBooks` method to print `Book` objects from the ArrayList `sortedBooks`.

```
class LibraryBookManager {
    // Adding an ArrayList and a TreeMap fields

    private ArrayList<Book> sortedBooks;
    private TreeMap<String, String> bookAuthorMap;


    // Constructor (written for you)
    public LibraryBookManager() {
        sortedBooks = new ArrayList<>();
        bookAuthorMap = new TreeMap<>();
    }
    // Adding books to the ArrayList and sorting
    public void addBook(Book book) {

        sortedBooks.add(book);
        Collections.sort(sortedBooks);

    }

    // Adding books to the TreeMap
    public void addBooksToMap(Book book) {

        bookAuthorMap.put(book.getTitle(), book.getAuthor());


    }// Continue on next page
```

```
// Printing the TreeMap elements as per instructions.
    public void printBookAuthors() {

        for (Map.Entry<String, String> entry : bookAuthorMap.entrySet()) {
            System.out.println(entry.getKey() + " -> " + entry.getValue());
        }









    }
    // Printing the ArrayList elements
    // Note that the Book class has a toString() method
    public void printBooks(){

        for (Book bk : sortedBooks) {
            System.out.println(bk);




    }
}
```

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked. Specify the question number for work that you do want marked.

Question 4. Cost of Algorithms [10 marks]

4.a) [5 marks]

What is the big-O cost of the following method, which takes a list of numbers and replaces each digit that is 8 or less by 0. Suppose the list of numbers, called `values`, contains `n` numbers.

- write the big-O cost of performing each line with a comment
- write the (total) number of times each of the lines will be performed
- write the total cost of the algorithm (big-O) at the bottom of the box

```
public void putInZeros(List<Integer> values){
   for (int k=8; k>0; k--) {                 // cost= O(  1  ), times= 8

      for (int i=0; i<values.size(); i++) {  // cost= O(  1  ), times= 8n

         if (i % 8 == 0)                     // cost= O(  1  ), times= 8n

            values.set(i,k-1);               // cost= O(  1  ), times= n

      }
   }
}                                          // TOTAL COST = O(   n   )
```

4.b) [5 marks]

Do the same for the following method, which is rather different. For the fifth line, you can give the *worst case* (i.e. maximum) number of times.

```
public void takeOut(List<String> values, String s){

    Collections.sort(values);              // cost= O(nlog(n))   times= 1

    for (int i=0; i<values.size(); i++) { // cost= O(   1   )   times= n

        if (values.get(i) == s)            // cost= O(   1   )   times= n

            values.remove(i);              // cost= O(   n   )   times(max)= n

   }
}                                          // TOTAL COST = O(   n^2   )
```

Question 5. Recursion [6 marks]

5.a) [4 marks]

Complete the following recursive `drawCastle(left, y, width)` method to draw a castle, having several levels. Each level is a rectangle, with two others above it, each taking up one third of the width of the level. The height of each rectangle should be 20, and each level should be 25 above the one below, that is, the horizontal gap between rectangles should be 5.

- You must use a *first-and-rest* recursion: `drawCastle` should draw the first level and then call itself recursively to draw the others.
- `(left, y)` is the **top left** of the first rectangle where first rectangle is base, and `width` is its **width**. Remember you can use the method `UI.fillRect(left, top, width, height)` to draw a rectangle.
- Your method should continue to draw rectangles as long as the width is at least 10.

For example, `drawCastle(20, 400, 600)` should output the following:

The dashed arrow line and the numbers are for indication, you do not need to draw them.

```
public class DrawCastle{
    public static void drawCastle(double left, double y, double width) {
        // YOUR CODE HERE

            if (width < 10) return;
            UI.printf("left: %.0f \t %.1f \t %.1f\n",left,width, y);

            UI.fillRect(left, y, width, 20);          // one level

            drawCastle(left, y-25, width/3);           // level's left  "turret"
            drawCastle(left+2*width/3, y-25, width/3); // level's right "turret"
    }
}
```

```
// OR.........
public class DrawCastle{
    public static void drawCastle(double left, double y, double width) {
        // YOUR CODE HERE
            if (width >= 10) {
                    UI.fillRect(left, y, width, 20);

                    drawCastle(left, y-25, width/3);
                    drawCastle(left+2*width/3, y-25, width/3);
            }
    }
}
```

5.b) [2 marks]

Consider the method `alterString(String)` given below.

```
public static String alterString(String str) {
        if (str.isEmpty()) {
            return "Done";
        } else {
            char firstChar = str.charAt(0);
            String remainder = str.substring(1);
            return alterString(remainder) + firstChar;
        }
    }
```

What string would be returned by this method, if it is called with the argument `lag` ?

```
// Your answer to 5.b

-->  Donegal

//
```

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked. Specify the question number for work that you do want marked.