MULTIMEDIA UNIVERSITY

STUDENT ID NO

SEAT NO: ____________________

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 2, 2019/2020

TCP1201 – OBJECT-ORIENTED PROGRAMMING AND DATA STRUCTURES
(All sections / Groups)

6 MARCH 2020
9:00 a.m. – 11:00 a.m.
(2 Hours)

Question	Mark
1	
2	
3	
4	
Total	

INSTRUCTIONS TO STUDENTS

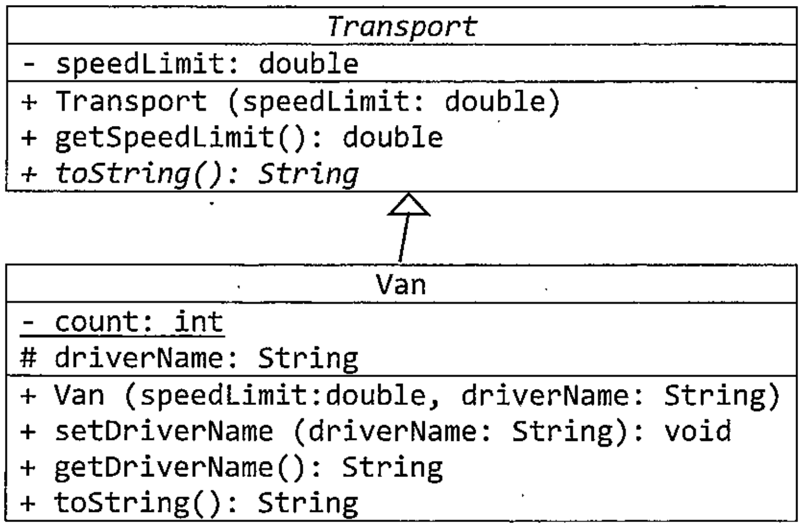
1. This Question paper consists of 12 pages with 4 Questions only.

2. Attempt all **FOUR** questions. All questions carry equal marks and the distribution of the marks for each question is given.

3. Please write all your answers in **this Question Paper**.

Question 1

The following UML Class Diagram is provided.


Transport
- speedLimit: double
+ Transport (speedLimit: double)
+ getSpeedLimit(): double
+ toString(): String
Van
- count: int
# driverName: String
+ Van (speedLimit:double, driverName: String)
+ setDriverName (driverName: String): void
+ getDriverName(): String
+ toString(): String

a. State the **relationship** between the Transport class and the Van class, and also the **superclass** and **subclass** in the UML Class Diagram above. [3 marks]

b. Write a **definition** for both the **Transport** class and the **Van** class based on the UML Class Diagram above and the main method below. [22 marks]

```
public static void main (String[] args) {
  Transport van1 = new Van (110, "Ben");
  System.out.println (van1);
  Transport van2 = new Van (90, "Cindy");
  System.out.println (van2);
}
```

Sample output:

```
Van: speedLimit = 110.0, name of driver = Ben
Van: speedLimit = 90.0, name of driver = Cindy
```

Continued...

Continued…

Question 2

a. When should **aggregation** be used instead of **inheritance**, and **vice versa**? Give an **example** for each case. [4 marks]

b. Write a **recursive** `sum` method that performs the following sum of positive odd integers. [6 marks]

```
int sum(int i)
```

where $sum(i) = 1 + 3 + 5 + 7 + \cdots + i$, if i is odd
$sum(i) = 1 + 3 + 5 + 7 + \cdots + (i - 1)$, if i is even

Sample calculations:

```
sum(5)  = 9
sum(10) = 25
```

Continued...

c. State one **benefit** of a generic type over a raw type. [1 mark]

d. Convert the following raw, non-generic version of the `min` method to a **generic** version. The `min` method finds the smallest element in the array. [7 marks]

```
static int min (int[] array) {
  int smallest = array[0];
  for (int i = 1; i < array.length; i++)
    if (array[i] < smallest)
      smallest = array[i];
  return smallest;
}
```

e. State **two differences** between **checked** and **unchecked** exceptions. [4 marks]

Continued...

f. The following main method is provided. It reads and prints the content of a specified file. [3 marks]

```
public static void main(String[] args) {
  String data = new String(
    Files.readAllBytes(Paths.get("data.txt")));
  System.out.println(data);
}
```

During compilation, the following error was produced.

```
OpenFile.java:x: error: unreported exception IOException;
must be caught or declared to be thrown
        Files.readAllBytes(Paths.get("data.txt")));
                          ^
1 error
```

Update the code to **handle the exception** stated in the error.

Continued...

Question 3

a. In terms of performance, **explain briefly** when an ArrayList is more efficient than a LinkedList and **vice versa**. [4 marks]

b. State the **output** of the following main method. There is no error in the method. [2 marks]

```
public static void main(String[] args) {
  PriorityQueue<Integer> queue1 =
    new PriorityQueue<>(Collections.reverseOrder());
  queue1.offer(5);
  queue1.offer(7);
  queue1.offer(2);
  queue1.offer(4);

  while (queue1.size() > 0)
    System.out.print(queue1.remove() + " ");
}
```

Continued...

c. State the main property of a **Set** when compared to List. Among **HashSet**, **LinkedHashSet**, and **TreeSet**, state when a particular set is preferred over the others. [7 marks]

d. An incomplete `MyStack` class is provided below. Assume there is no error in the class except the methods `push`, `peek`, `pop`, and `isEmpty` are incomplete. Complete the definition for the four (4) methods. Do not define other method in your solution. [12 marks]

```
public class MyStack<E> {
  private ArrayList<E> data = new ArrayList<>();
  public int getSize() {
    return data.size();
  }
  public E peek() {
    // To be completed
  }
  public void push(E o) {
    // To be completed
  }
  public E pop() {
    // To be completed
  }
  public boolean isEmpty() {
    // To be completed
  }
}
```

Continued...

Continued...

Question 4

a. An incomplete MyArrayList class is provided below. Assume there is no error in the class except the method indexOf is incomplete. Complete the definition for the indexOf method. Do not define other method in your solution. [5 marks]

```
public class MyArrayList<E> {
  public static final int INITIAL_CAPACITY = 16;
  private E[] data = (E[])new Object[INITIAL_CAPACITY];
  private int size = 0; // Number of elements in the list
  ...

  /** Return the index of the first matching element
   *  in this list. Return -1 if no match. */
  public int indexOf(Object e) {
    // To be completed
  }
}
```

Continued...

b. Construct a **binary search tree** (BST) based on the following sequence of numbers. Then, state the result of **in-order** and **post-order** traversals on the BST. [6 marks]
10, 7, 8, 4, 15, 12, 1, 3

c. A `TreeNode` definition and an incomplete `postorder` method are provided below. Complete the `postorder` method which performs recursive post-order traversal on a binary search tree. Do not define other method in your solution.
[4 marks]

```
class TreeNode<E> {
  E element;
  TreeNode<E> left;
  TreeNode<E> right;
}
...
void postorder(TreeNode<E> root) {
  // To be completed
}
```

Continued...

d. A Node class and a MyLinkedList class are provided below. Assume there is no error in the classes except the last else block in the removeLast method is incomplete. Complete the last else block. Do not define other method in your solution. [10 marks]

```
class Node<E> {
  E element;
  Node<E> next;
}

public class MyLinkedList<E> {
  private Node<E> head, tail;
  private int size = 0; // Number of elements in the list
  ...
  /** Remove the last node and
   *  return the object that is contained in the removed node.
   */
  public E removeLast() {
    if (size == 0) {
      return null;
    }
    else if (size == 1) {
      E temp = head.element;
      head = tail = null;
      size = 0;
      return temp;
    }
    else {
      // To be completed.
    }
  }
}
```

End of Paper