WIA1002/WIB1002 Data Structure

Lab : Priority Queue

1. Given an integer array which consists of {4, 8, 1, 2, 9, 6, 3, 7}. Insert these integers into a priority queue using its ADT. Then, perform the following operations to the priority queue:

   - toString() - Display all the elements inside this priority queue.
   - poll() – retrieve and remove the first element in this priority queue.
   - toArray() – convert the priority queue into an array and display.
   - peek() – retrieve the first element in the priority queue.
   - contains() – check if the priority queue consists of element “1”.
   - size() – get the current size of the priority queue.
   - isEmpty() – display while removing the elements in the queue until it is empty.

```
import java.util.PriorityQueue;
public class IntArrayPQ {
   public static void main(String [] args)
   {
      PriorityQueue<Integer> q = new PriorityQueue<Integer>();
      int[] arr = {4, 8, 1, 2, 9, 6, 3, 7};

      q.add(4);
      q.add(8);
      q.add(1);
      q.add(2);
      q.add(9);
      q.add(6);
      q.add(3);
      q.add(7);

      System.out.println(q.toString());
      System.out.println("First element (and remove): " + q.poll());
      Object[] array = q.toArray();
      for (int i = 0; i<array.length; i++)
      {
         System.out.println(array[i]);
      }
      System.out.println("First element (and NOT remove): " + q.peek());
      System.out.println("Element 1 is inside the PQ: " + q.contains(1));
      System.out.println("Size of the PQ: " + q.size());
      while(!q.isEmpty())
      {
```

```
            System.out.println("Current element to be removed: " + q.poll());
            System.out.println("Remaining element(s): " + q.toString());
        }
    }
}
```

2. Create two priority queues with the following elements: {"George", "Jim", "John", "Blake", "Kevin", "Michael"} and {"George", "Katie", "Kevin", "Michelle", "Ryan"}. Find their union, difference, and intersection.

```
import java.util.PriorityQueue;
public class Compare2PQ {
   public static void main(String[] args) {
      PriorityQueue<String> pq1 = new PriorityQueue<>();
      pq1.add("George");
      pq1.add("Jim");
      pq1.add("John");
      pq1.add("Blake");
      pq1.add("Kevin");
      pq1.add("Michael");
      System.out.println("First Priority Queue: "+pq1);
      PriorityQueue<String> pq2 = new PriorityQueue<>();
      pq2.add("George");
      pq2.add("Katie");
      pq2.add("Kevin");
      pq2.add("Michelle");
      pq2.add("Ryan");
      System.out.println("Second Priority Queue: "+pq2);
      System.out.println("Intersection for 2 PQs:");
      for (String element : pq1){
         if (pq2.contains(element))
            System.out.print(element + " ");
      }
      System.out.println("\nDifference for 2 PQs:");
      for (String element : pq1){
         if (!pq2.contains(element))
            System.out.print(element + " ");
      }
      System.out.println("\nUnion for 2 PQs:");
      for (String element : pq1){
         System.out.print(element + " ");
      }
      for (String element : pq2){
```

```
            if (!pq1.contains(element))
                System.out.print(element + " ");
        }
        System.out.println();
    }
}
```

3. Given following books information and the main class:

```
import java.util.Queue;
public class TestComparableBook {
    public static void main(String[] args) {
    Queue<ComparableBook> BookQueue = new java.util.PriorityQueue<>();
    BookQueue.add(new ComparableBook(1065, "Effective Java: Third Edition"));
    BookQueue.add(new ComparableBook(3012, "Java: A Beginner Guide Seventh Edition"));
    BookQueue.add(new ComparableBook(1097, "Learn Java in One Day and Learn It Well"));
    BookQueue.add(new ComparableBook(7063, "Beginning Programming with Java
(Dummies)"));
    BookQueue.add(new ComparableBook(6481, "Java: Programming Basic for Absolute
Beginner"));

    System.out.println(BookQueue);
    while (BookQueue.peek() != null) {
      System.out.println("Head Element: " + BookQueue.peek());
      BookQueue.remove();
      System.out.println("Priority queue: " + BookQueue);
    }
  }
}
```

Write the code for ComparableBook class using Comparable.

```
package priorityqueue;

public class ComparableBook implements Comparable<ComparableBook>
{
    private int BookID;
    private String BookName;
```

```
public ComparableBook(int id, String name) {
   this.BookID = id;
   this.BookName = name;
}

public int getId() {
   return BookID;
}

public void setId(int id) {
   this.BookID = id;
}

public String getName() {
   return BookName;
}

public void setName(String name) {
   this.BookName = name;
}

@Override
public boolean equals(Object o) {
   if (!(o instanceof ComparableBook)) {
      return false;
   }
   ComparableBook p = (ComparableBook) o;
   if (this.BookID == p.getId()) {
      return true;
   }

   return false;
}

@Override
public int hashCode() {
   return this.BookID;
}

@Override
public String toString() {
   return "(" + BookID + ", " + BookName + ")";
```

```
    }

    @Override
    public int compareTo(ComparableBook cp) {

        int cpId = cp.getId();
        String cpName = cp.getName();

        if (this.getId() < cpId) {
            return -1;
        }

        if (this.getId() > cpId) {
            return 1;
        }

        if (this.getId() == cpId) {
            return this.getName().compareTo(cpName);
        }
        return 0;
    }
}
```