Generics

1. Write the following method that returns a new **ArrayList**. The new list contains the non-duplicate elements from the original list.

   public static <E> ArrayList<E> removeDuplicates(ArrayList<E> list)

** To find duplication, comparison is to be done → compreTo() method is needed → must implement Comparable interface.*

```
import java.util.ArrayList;

public class DemoGeneric {
      public static <E extends Comparable<E>>
            ArrayList<E> removeDuplicates(ArrayList<E> list) {
            for (int i = 0; i < list.size() - 1; i++) {
                  for (int j = i + 1; j < list.size(); j++) {
                        if (list.get(i).compareTo(list.get(j)) == 0)
                              list.remove(j);
                  }
            }
            return list;
      }
}
```

2. Implement the following generic method for linear search. If key found, return the index, -1 otherwise.

public static <E extends Comparable<E>> int linearSearch(E[] list, E key)

```
public class GenericLinearSearch {

   public static <E extends Comparable<E>> int linearSearch(E[] list, E key) {
      for (int i = 0; i < list.length; i++) {
         if (key.compareTo(list[i]) == 0)
            return i;
      }
      return -1;
   }
}
```

3. Implement the following method that returns the maximum element in an array.

public static <E extends Comparable<E>> E max(E[] list)

** Assume the first element to be the maximum element.*

```
public class DemoMaxArray {
    public static <E extends Comparable<E>> E max(E[] list) {
        E max = list[0];
        for (int i = 1; i < list.length; i++) {
            if (list[i].compareTo(max) > 0)
                max = list[i];
        }
        return max;
    }
}
```

4. Write a generic method that returns the maximum element in a two-dimensional array.

public static <E extends Comparable<E>> E max(E[][] list)

** Assume the first element to be the maximum element.*

```
public class DemoMax2DArray {
      public static <E extends Comparable<E>> E max(E[][] list) {
            E max = list[0][0];
            for (int i = 0; i < list.length; i++) {
                  for (int j = 0; j < list[i].length; j++) {
                        if (list[i][j].compareTo(max) > 0)
                              max = list[i][j];
                  }
            }
            return max;
      }
}
```

5. Write the following method that shuffles an ArrayList:

public static <E> void shuffle(ArrayList<E> list)

** Math.random() returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.*

```java
import java.util.ArrayList;

public class DemoShuffleArray {
    public static <E> void shuffle(ArrayList<E> list) {
        for (int i = 0; i < list.size(); i++) {
            int index = (int)(Math.random() * list.size());
            E temp = list.get(i);
            list.set(i, list.get(index));
            list.set(index, temp);
        }
    }
}
```

6. Write the following method that returns the largest element in an ArrayList:

public static <E extends Comparable<E>> E max(ArrayList<E> list)

```java
import java.util.ArrayList;

public class DemoMaxArrayList {
    public static <E extends Comparable<E>> E max(ArrayList<E> list) {
        E max = list.get(0);
        for (int i = 1; i < list.size(); i++) {
            if (max.compareTo(list.get(i)) < 0)
                max = list.get(i);
        }
        return max;
    }
}
```

To test it:

```
import java.util.*;
public class Test {
    public static void main(String[] args) {
        Integer[] intArray = {new Integer(2), new Integer(4),
        new Integer(3)};
        ArrayList<Integer> intList = new ArrayList<>(Arrays.asList(intArray));

        Double[] doubleArray = {new Double(3.4), new Double(1.3),
        new Double(-22.1)};
        ArrayList<Double> doubleList = new ArrayList<>(Arrays.asList(doubleArray));

        Character[] charArray = {new Character('a'),
        new Character('J'), new Character('r')};
        ArrayList<Character> charList = new ArrayList<>(Arrays.asList(charArray));

        String[] stringArray = {"Tom", "Susan", "Kim"};
        ArrayList<String> stringList = new ArrayList<>(Arrays.asList(stringArray));

        System.out.println("Maximum Integer object: " + DemoMaxArrayList.max(intList));
        System.out.println("Maximum Double object: " + DemoMaxArrayList.max(doubleList));
        System.out.println("Maximum Character object: " + DemoMaxArrayList.max(charList));
        System.out.println("Maximum String object: " + DemoMaxArrayList.max(stringList));
    }
}
```

Practical 4 Exercise 3

7. Write a program that meets the following requirements:
   - Define a class named **Point** with two data fields x and y to represent a point's x- and y-coordinates. Implement the Comparable interface for comparing the points on x-coordinates. If two points have the same x-coordinates, compare their y-coordinates.
   - Define a class named **CompareY** that implements Comparator<Point>. Implement the compare method to compare two points on their y-coordinates. If two points have the same y-coordinates, compare their x-coordinates.
   - Randomly create 100 points and apply the Arrays.sort method to display the points in increasing order of their x-coordinates and in increasing order of y-coordinates, respectively.

     [Note that in Java, there is a method Math.random(), which returns a double value between 0.0 and 1.0. And there is another method Random.nextInt(int n), which returns a random value in the range of 0 (inclusive) and n (exclusive).]

```
import java.util.Comparator;

public class CompareY implements Comparator<Point> {

   public int compare(Point p1, Point p2) {
      double x1 = p1.getX();
      double y1 = p1.getY();
      double x2 = p2.getX();
      double y2 = p2.getY();

      if (y1 == y2) {
         if (x1 < x2)
            return -1;
         else if (x1 == x2)
            return 0;
         else
            return 1;
      }
      else if (y1 < y2)
         return -1;
      else
         return 1;

   }
}
```

```java
public class Point implements Comparable<Point> {
   private double x;
   private double y;

   Point() {}

   Point(double x, double y) {
      this.x = x;
      this.y = y;
   }

   public void setX(double x) {
      this.x = x;
   }

   public void setY(double y) {
      this.y = y;
   }

   public double getX() {
      return x;
   }

   public double getY() {
      return y;
   }

   @Override // Override the compareTo method in the Comparable class
   public int compareTo(Point point) {
      if (x == point.getX()) { // same x-coordinates
         if (y > point.getY())  // compare their y-coordinates
            return 1;
         else if (y < point.getY())
            return -1;
         else
            return 0;
      }
      else if (x > point.getX())
         return 1;
      else
         return -1;
   }

   @Override // Override the toString method in the Object class
   public String toString() {
      return "(" + String.format("%.2f", x) + ", "
         + String.format("%.2f", y) + ")";
   }
}
```

```
import java.util.*;

public class TestCompareCoordinates {
   public static void main(String[] args) {

      // Randomly create 100 points
      Point[] points = new Point[100];
      for (int i = 0; i < points.length; i++) {
         points[i] = new Point((double)(Math.random() * 5),
            (double)(Math.random() * 5));
      }

      // Display the points in increasing order of their x-coordinates
      Arrays.sort(points);
      List<Point> list1 = Arrays.asList(points);
      System.out.println("\nPoints in increasing order x-coordinates:");
      System.out.println(list1);


      // Display the points in increasing order of their y-coordinates
      Arrays.sort(points, new CompareY());
      List<Point> list2 = Arrays.asList(points);
      System.out.println("\nPoints in increasing order y-coordinates:");
      System.out.println(list2);
   }
}
```

8. A Java program contains various pairs of grouping symbols, such as:

   - Parentheses: (and)
   - Braces: { and }
   - Brackets: [and]

   Note that the grouping symbols cannot overlap. For example, `(a{b)}` is illegal. Write a program to check whether a Java source-code file has correct pairs of grouping symbols. Pass the source-code file name as a command-line argument.

```java
public class DemoStack {
   public static void main(String[] args) throws IOException {
      if (args.length != 1) {
         System.out.println("Usage: Java Exercise_20_11 Source-codeFileName");
         System.exit(0);
      }

      File file = new File(args[0]);
      if (!file.exists()) {
         System.out.println("The file " + args[0] + " does not exist!");
         System.exit(1);
      }

      Stack<Character> symbols = new Stack<>();  // Create a stack

      try ( // Create an input stream for file
            Scanner input = new Scanner(file);
      ) {
         // Continuously read chars from input
         while (input.hasNext()) {
            String line = input.nextLine();
            for (int i = 0; i < line.length(); i++) {
               char ch = line.charAt(i);
               // Push symbols (, {, and [ on to the stack
               if (ch == '(' || ch == '{' || ch == '[') {
                     symbols.push(ch);
               } // Process stack
               else if (ch == ')' || ch == '}' || ch == ']') {
                     processSymbols(symbols, ch);
               }
            }
         }
      }
      System.out.println("The Java source-code " +
         (symbols.isEmpty() ? "has" : "does not have") + " correct pairs.");
   }

   private static void processSymbols(Stack<Character> stack, Character ch) {
      // Remove matching symbols from stack
      if ((stack.peek() == '(' && ch == ')') ||
          (stack.peek() == '[' && ch == ']') ||
          (stack.peek() == '{' && ch == '}')) {
         stack.pop();
      }
      else if ((stack.peek() != '(' && ch == ')') ||
          (stack.peek() != '[' && ch == ']') ||
          (stack.peek() != '{' && ch == '}')) {
         System.out.println("The Java source-code does not have"
            + " correct pairs.");
         System.exit(1);
      }
   }
}
```