Data Structures

Objective:
The objective of this lab exercise is to analyze and compare the time complexity of different algorithms in Java. You will implement and test three different sorting algorithms and measure their performance on various input sizes.

Algorithms to Implement:
1. Bubble Sort: A simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order.
2. Selection Sort: A sorting algorithm that divides the input list into two parts: the left sub-list of items already sorted and the right sub-list of items to be sorted.
3. Merge Sort: A divide-and-conquer algorithm that divides the unsorted list into n sub-lists, each containing one element, and then repeatedly merges sub-lists to produce new sorted sub-lists until there is only one sub-list remaining.

Instructions:
1. Create a Java class that contains the three sorting algorithms (Bubble Sort, Selection Sort, and Merge Sort).
2. Implement each sorting algorithm in separate methods within the class.
3. Write a method to generate random integer arrays of various sizes.
4. Implement a method that calculates and prints the time taken to sort an array using each of the sorting algorithms.
5. Run the sorting algorithms on arrays of different sizes (e.g., 100, 1000, 10000, 100000 elements).
6. Record and analyze the time taken for sorting on each input size.
7. Compare the time complexity of the sorting algorithms and discuss your findings.

Hints:
- You can use the `System.currentTimeMillis()` or `System.nanoTime()` methods to measure the execution time of your sorting algorithms.
- Use the System.arraycopy() method to efficiently copy arrays.
- Make sure to run each sorting algorithm on the same input array for fair comparisons.

Sample Code:

```
import java.util.Random;

public class SortingAnalysisLab {
   public static void main(String[] args) {
      SortingAnalysisLab lab = new SortingAnalysisLab();
      lab.runSortingAnalysis();
   }

   public void bubbleSort(int[] arr) {
      // Implement Bubble Sort here
   }

   public void selectionSort(int[] arr) {
      // Implement Selection Sort here
   }

   public void mergeSort(int[] arr) {
      // Implement Merge Sort here
   }

   public int[] generateRandomArray(int size) {
      Random random = new Random();
      int[] arr = new int[size];
      for (int i = 0; i < size; i++) {
         arr[i] = random.nextInt(1000);
      }
      return arr;
   }

   public void runSortingAnalysis() {
      // Perform sorting analysis here
      // Measure and compare the time taken for sorting different input sizes
   }
}
```