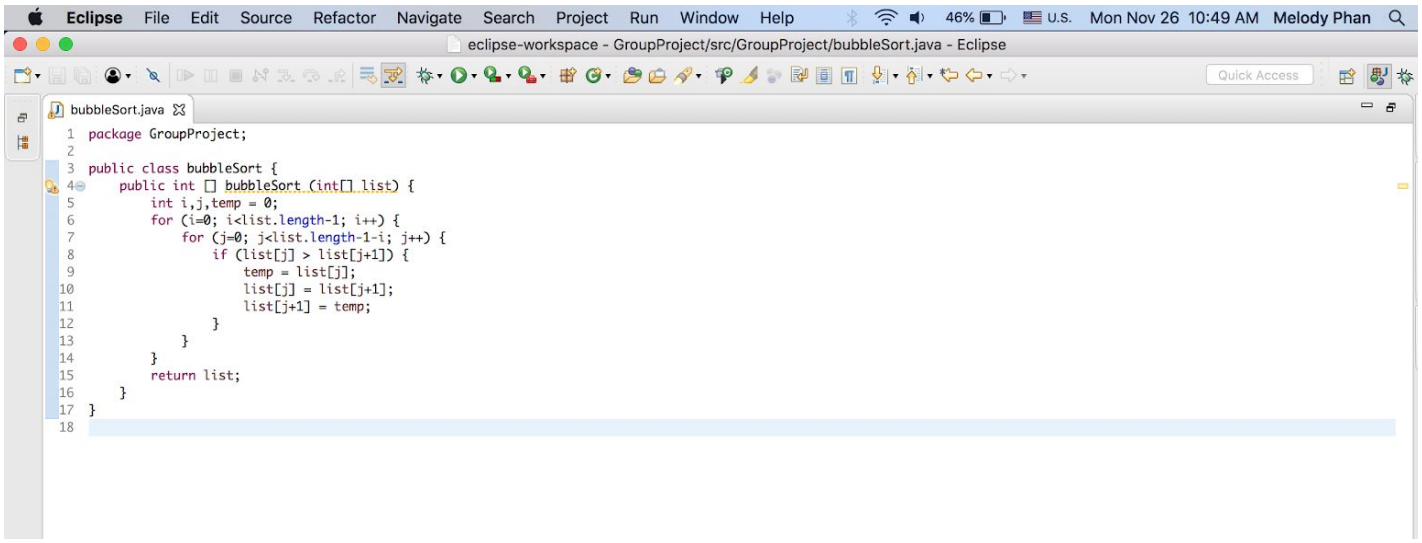
Group Project

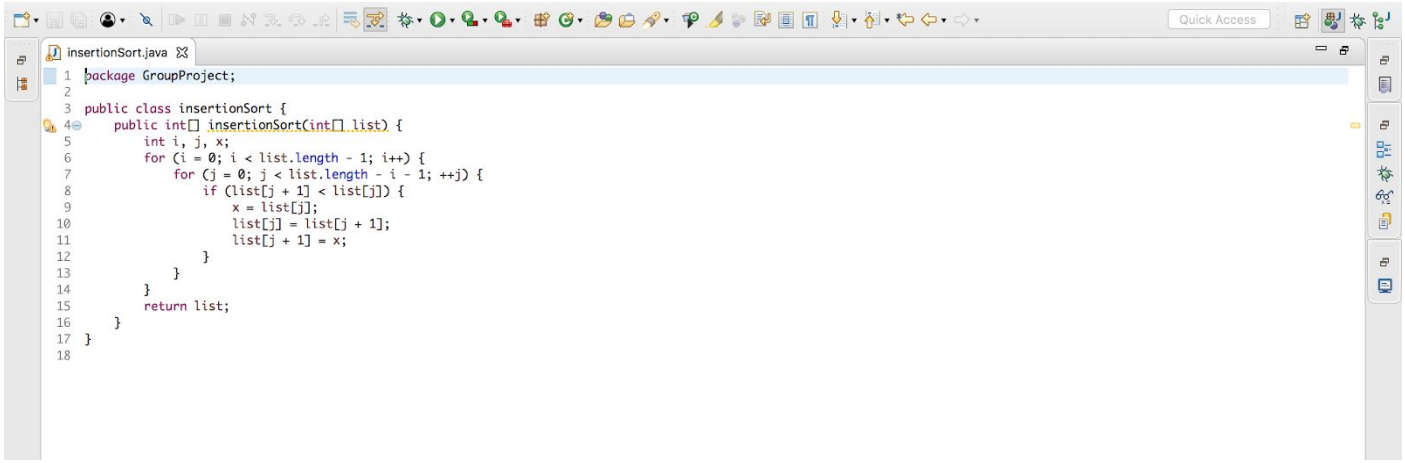
Tran Phan

Shravan Raul




```java
package GroupProject;

public class bubbleSort {
    public int [] bubbleSort (int[] list) {
        int i,j,temp = 0;
        for (i=0; i<list.length-1; i++) {
            for (j=0; j<list.length-1-i; j++) {
                if (list[j] > list[j+1]) {
                    temp = list[j];
                    list[j] = list[j+1];
                    list[j+1] = temp;
                }
            }
        }
        return list;
    }
}
```




```java
package GroupProject;

public class insertionSort {
    public int[] insertionSort(int[] list) {
        int i, j, x;
        for (i = 0; i < list.length - 1; i++) {
            for (j = 0; j < list.length - i - 1; ++j) {
                if (list[j + 1] < list[j]) {
                    x = list[j];
                    list[j] = list[j + 1];
                    list[j + 1] = x;
                }
            }
        }
        return list;
    }
}
```

mergeSort.java

```java
package GroupProject;

import java.util.Arrays;

public class mergeSort {
    public void mergeSort(int[] list) {
        if (list.length >= 2) {
            int[] left = Arrays.copyOfRange(list, 0, list.length / 2);
            int[] right = Arrays.copyOfRange(list, list.length / 2, list.length);

            mergeSort(left);
            mergeSort(right);
            merge(list, left, right);
        }
    }

    public static void merge(int[] list, int[] left, int[] right) {
        int i1 = 0;
        int i2 = 0;
        for (int i = 0; i < list.length; i++) {
            if (i2 >= right.length || (i1 < left.length && left[i1] <= right[i2])) {
                list[i] = left[i1];
                i1++;
            } else {
                list[i] = right[i2];
                i2++;

            }

        }
    }
}
```

selectionSort.java

```java
package GroupProject;

public class selectionSort {
    public int[] selectionSort(int[] list) {
        int i, j, min, x;
        for (i = 0; i < list.length - 1; i++) {
            min = i;
            for (j = i + 1; j < list.length; j++) {
                if (list[j] < list[min])
                    min = j;
            }
            x = list[min];
            list[min] = list[i];
            list[i] = x;
        }
        return list;
    }
}
```

Demo.java | jumpSearch.java | linearSearch.java | binarySearch.java

```java
package GroupProject;

public class jumpSearch {
    public int jumpSearch(int[] list, int key) {
        int jump = (int) Math.sqrt(list.length);
        int left = 0;
        int right = 0;

        while (left < list.length && list[left] <= key) {
            right = Math.min(list.length - 1, left + jump);

            if (list[left] <= key && list[right] >= key) {
                break;
            }

            left += jump;
        }

        if (left >= list.length || list[left] > key) {
            return -1;
        }

        right = Math.min(list.length - 1, right);

        for (int i = left; i <= right && list[i] <= key; ++i) {
            if (list[i] == key) {
                return i;
            }
        }

        return -1;
    }

}
```

Demo.java | jumpSearch.java | linearSearch.java | binarySearch.java

```java
package GroupProject;

public class linearSearch {

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

Demo.java | jumpSearch.java | linearSearch.java | binarySearch.java

```java
package GroupProject;

public class binarySearch {
    public int binarySearch (int[] list, int key) {
        int min = 0;
        int max = list.length-1;

        while (min <= max) {
            int mid = (min +max) / 2;
            if (list[mid] < key) {
                min = mid + 1;
            }
            else if (list[mid] > key) {
                max = mid - 1;
            } else {
                return mid;
            }
        }
        return -(min + 1);
    }
}
```

Demo.java

```java
package GroupProject;
import java.util.Arrays;

public class Demo {
    public static void main(String[] args) {
        int[] list = new int[100000];
        for(int i = 0; i < list.length; i++) {
          list[i] = (int)(Math.random()*1000000 + 1);
        }
        //System.out.println(Arrays.toString(list));

        //Sort
        //Bubble sort
        bubbleSort bubbleObj = new bubbleSort();
        Instant start = Instant.now();
        bubbleObj.bubbleSort(list);
        Instant end = Instant.now();
        Duration timeElapsed = Duration.between(start, end);
        System.out.println("Bubble Sort: "+ timeElapsed.toMillis() +" milliseconds");
        System.out.println();

        //Merge sort
        mergeSort mergeObj = new mergeSort();
        Instant start1 = Instant.now();
        mergeObj.mergeSort(list);
        Instant end1 = Instant.now();
        Duration timeElapsed1 = Duration.between(start1, end1);
        System.out.println("Merge Sort: "+ timeElapsed1.toMillis() +" milliseconds");
        System.out.println();

        //Selection sort
        selectionSort selectObj = new selectionSort();
        Instant start2 = Instant.now();
        selectObj.selectionSort(list);
        Instant end2 = Instant.now();
        Duration timeElapsed2 = Duration.between(start2, end2);
        System.out.println("Selection Sort: "+ timeElapsed2.toMillis() +" milliseconds");
        System.out.println();

        //Insertion sort
        insertionSort insertObj = new insertionSort();
        Instant start3 = Instant.now();
        insertObj.insertionSort(list);
        Instant end3 = Instant.now();
        Duration timeElapsed3 = Duration.between(start3, end3);
        System.out.println("Insertion Sort: "+ timeElapsed3.toMillis() +" milliseconds");
        System.out.println();

        //Sort the array
        Arrays.sort(list);
        //Search
        Scanner input = new Scanner (System.in);
        System.out.print("Enter the number to search: ");
        int numb = input.nextInt();
        System.out.println();

        //Binary search
        System.out.print("Binary Search: ");
        binarySearch binaryObj = new binarySearch();
        Instant start4 = Instant.now();
        int n = binaryObj.binarySearch(list, numb);
        String result = n > 1 ? "Number found at index: " + n : "Number not found";
        System.out.println(result);
        Instant end4 = Instant.now();
        Duration timeElapsed4 = Duration.between(start4, end4);
        System.out.println(timeElapsed4.toMillis() +" milliseconds");
        System.out.println();
```

```java
        //Jump search
        System.out.print("Jump Search: ");
        jumpSearch jumpObj = new jumpSearch();
        Instant start5 = Instant.now();
        int n2 = jumpObj.jumpSearch(list, numb);
        String result2 = n2 > 1 ? "Number found at index: " + n2: "Number not found";
        System.out.println(result2);
        Instant end5 = Instant.now();
        Duration timeElapsed5 = Duration.between(start5, end5);
        System.out.println(timeElapsed5.toMillis() +" milliseconds");
        System.out.println();

        //Linear search
        System.out.print("Linear Search: ");
        linearSearch linearObj = new linearSearch();
        Instant start6 = Instant.now();
        int n3 = linearObj.linearSearch(list, numb);
        String result3 = n3 > 1 ? "Number found at index: " + n3: "Number not found";
        System.out.println(result3);
        Instant end6 = Instant.now();
        Duration timeElapsed6 = Duration.between(start6, end6);
        System.out.println(timeElapsed6.toMillis() +" milliseconds");
        System.out.println();
    }
}
```

Eclipse File Edit Source Refactor Navigate Search Project Run Window Help 44% U.S. Mon Nov 26 10:51 AM Melody Phan

eclipse-workspace - GroupProject/src/GroupProject/Demo.java - Eclipse

Console

```
<terminated> Demo (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_162.jdk/Contents/Home/bin/java (Nov 26, 2018, 10:50:29 AM)
Bubble Sort: 16503 milliseconds

Merge Sort: 19 milliseconds

Selection Sort: 1193 milliseconds

Insertion Sort: 1457 milliseconds

Enter the number to search: 873453

Binary Search: Number not found
0 milliseconds

Jump Search: Number not found
0 milliseconds

Linear Search: Number not found
3 milliseconds
```

Console

```
<terminated> Demo (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_162.jdk/Contents/Home/bin/java (Nov 26, 2018, 10:55:07 AM)
Bubble Sort: 16561 milliseconds

Merge Sort: 19 milliseconds

Selection Sort: 1209 milliseconds

Insertion Sort: 1506 milliseconds

Enter the number to search: 999999

Binary Search: Number found at index: 99997
0 milliseconds

Jump Search: Number found at index: 99997
0 milliseconds

Linear Search: Number found at index: 99997
1 milliseconds
```

Conclusion:

The best sorting method is Merge Sort

The best searching methods are Binary and Jump Search

Step 1: Select any four sorting algorithm and three searching algorithms

Step 2: Understand the logic of all the algorithms

Step 3: Create java program and use your sorting/searching source codes and integrate it into your main java project.

Step 4: Create a separate java class for each algorithm

Step 5: Create a random function that generates at least 100000 random integer numbers from 1 to 1 million(No need to print out or store the numbers)

Step 6: Insert start transaction and end transaction for each sorting and searching methods

Step 7: Calculate the time in milliseconds for each sorting and searching class

Step 8: Compare the performance of each algorithm