

# Practical-05: Sorting Algorithms

**Due** 13 Oct by 22:00      **Points** 100      **Available** after 9 Sep at 0:00

This practical will focus on using class and algorithms to solve some real world problems.

You should use JAVA to complete this practical.

## Submission

Note: all class work must be submitted to your SVN repository. For this practice, you shall use:

```
1 | https://version-control.adelaide.edu.au/svn/<YOUR-ID>/2019/s2/fcs/week-07/practical-05

...

save this project according to the requirements of each part in this practical

add to your svn repository and submit it to your web submission

...
```

## Part 1 - Build Your Own Sorting Algorithms

**Question 1:** Set up a new project in your IDE with a Main class. Your project name should be MySortAlg and it should be saved under practical-05.

**Question 2:** You need to build a base class for all your sorting algorithms. You should name this base class as **MySortAlg**. Since we can not implement the sort method in base class, you should declare it as an [abstract class](https://docs.oracle.com/javase/tutorial/java/land/abstract.html) [\\_](https://docs.oracle.com/javase/tutorial/java/land/abstract.html)(<https://docs.oracle.com/javase/tutorial/java/land/abstract.html>)[\\_](https://docs.oracle.com/javase/tutorial/java/land/abstract.html).

```
public abstract class MySortAlg
Methods:
abstract int[] sort(int[] array);
```

**Question 3:** Define an **InsertionSort** class. This class should **extend the** MySortAlg class. In this class, your sort() method should take a sequence of integer numbers as input and return a sorted array **in descending order**.

```
public class InsertionSort
Methods:
public int[] sort(int[] array);
```

**Question 4:** Define a **MergeSort** class. This class should **extend the** MySortAlg class. In this class, your sort() method should take a sequence of integer numbers as input and return a sorted array **in descending order**. You can add other methods to help you to implement merge sort algorithms. Please note that for all other methods you create in this class, you can design the signature by your self. But these methods should be declared as **private**.

```
public class MergeSort
Methods:
public int[] sort(int[] array);
```

**Question 5:** Define a **QuickSort** class. This class should **extend the** MySortAlg class. In this class, your sort() method should take a sequence of integer numbers as input and return a sorted array **in descending order**. You can add other methods to help you to implement merge sort algorithms. Please note that for all other methods you create in this class should be declared as **private**.

```
public class QuickSort
Methods:
public int[] sort(int[] array);
```

**Question 6:** In your **Main** class, build a test method to test your code.

```
public static boolean test(int[] result, int[] ans);
```

The test method should take two arrays as input. Pass the sorted array from your sorting function into 'result' and pass an array that you have sorted by hand to 'ans'. Your test method should check if these two arrays are exactly the same, return true when they are the same and false when they not. You should generate at least 3 test cases and use these test cases to test all sorting algorithms above.

For example:

```
int[] result = sortObject.sort( {4,2,3,1} );
int[] ans = {1,2,3,4};
test( result, ans );
```

Would return true if the sorting algorithm is working correctly.

**Question 7:** Hint: the sorting algorithm you just built will help you to solve this question. In your **Main** class, build a compare method.

```
public static boolean compare(int[] arr1, int[] arr2);
```

This method take two arrays as input and return true when they are equal and return false otherwise. Two arrays are said to be equal if both of them contain same set of elements, order of elements may be different though. You can assume that in one array, one element will only appear one time. For example,

```
Input:  arr1[] = {2,4,1,3,5}
        arr2[] = {2,4,1,3,5}
Output: True
```

```
Input:  arr1[] = {1,3,5,2,4}
        arr2[] = {2,4,1,3,5}
Output: True
```

```
Input:  arr1[] = {1,7,2,8,5}
        arr2[] = {2,4,1,3,5}
Output: False
```

**Question 8:** Hint: the sorting algorithm you just built will help you to solve this question. In your **Main** class, build a findSmallestSum method.


```
public static int findSmallestSum(int[] array);
```

This method will take an array as input and return the smallest sum of two elements in this array. You can assume that the length of input array is always  $\geq 2$ . For example,

```
Input:  array = {4,7,0,0,2,9,10}
Output: 0
```

## Part 2 - Marks of Class

**Question 9:** Set up a new project in your IDE with a Main class. Your project name should be **MarksOfClass** and it should be saved under practical-05

**Question 10:** You are provided a [students.txt](#)  file with marks of N students. For each student, the student's marks in Physic, Chemistry and Maths (the same order in txt file) are provided to you. You should define a **Student** class to save these data.

```
1 | John 95 75 88
2 | Alice 88 95 75
3 | Johnson 95 75 88
4 | Dennis 60 100 100
5 | Jack 77 84 93
6 | Tod 84 86 80
7 | Tom 68 70 75
8 | Dave 90 90 92
9 | David 99 70 87
10 | Trent 89 77 90
11 | Bob 100 67 89
12 | Fiona 77 89 90
13 | Peter 80 88 82
14 | Amy 85 95 78
15 | Nancy 83 93 82
16 | Richard 81 91 86
17 | Daniel 77 78 79
18 | James 80 90 85
19 | Cathy 95 74 89
20 | Paul 84 87 79
```

Define a **Student** class.

```
public class Student
Attributes: name, physic, chemistry, math, average
Methods:
//average score should be calculated in Constructor
public Student(string name, int physic, int chemistry, int math);
//mutator and accessor for each attributes
...
```

**Question 11:** In your **Main** class, define a readData method, the input of this method should be a file name and the output should be an array of students.

```
private static Student[] readData(string filename);
```

**Question 12:** You want to sort the Student's Numbers in Descending order of their average marks for three courses. In your **Student** class, define a sortStudents method, the input of this method should be an array of students and the output should be an array of sorted students.

```
public Student[] sortStudents(Student[] students);
```

Now, once this is done, only those students who have the same average marks have to be sorted in the descending order of their Physics marks.

If their Physics marks are also the same, you need to sort them in the descending order of their Chemistry marks.

If their Chemistry marks are the same, you need to sort them in the descending order of their Maths marks.

If their Math marks are the same, you need to sort them in the ascending order of their name.

Note: You have to implement the sorting algorithm by yourself. Try to add some other **private methods** in Student class so that you can **reuse your code** for sorting.

**Question 13:** In your **Main** class, define a printStudents method, the input of this method should be an array of students and it returns void.

```
public static void printStudents(Student[] students);
```

In this method, you should print the sorted students in console by the following format:

```
Student AverageScore Physic Chemistry Maths
John    90           88     90         92
Trent   88           87     88         89
...
```

## Part 3 - Recursive

**Question 14:** Set up a new project in your IDE with a Main class. Your project name should be **Recursive** and it should be saved under practical-05

**Question 15:** In your Main class, create a **nextHappyNum** method.

```
int nextHappyNum(int num)
```

For a given non-negative integer N, find the next smallest Happy Number. A number is called happy if it leads to 1 after a sequence of steps where in each step number is replaced by sum of squares of its digit that is if we start with Happy Number and keep replacing it with digits square sum, we reach 1. For example,

```
Input:  8
Output: 10

Input:  3
Output: 7

Input:  0
Output: 1

Input:  10
Output: 13
```

**Question 16:** In your Main class, create a **decodeString** method.

```
String decodeString(String s)
```

An encoded string (s) is given, the task is to decode it. The pattern in which the strings were encoded were as follows

**original string:** abbbababbbababbbab

**encoded string :** "3[a3[b]1[ab]]".

For example:

Input: 1[b]

Output: b

Input: 3[b2[ca]1[d]]

Output: bcacadbacadbacadb