# Name: Tạ Thị Thùy Dương \_ Student’s ID: ITCSIU21053

# 10 LABS x 2 hours 15 minutes

* Lab 1: OOP Reviews & Arrays
* Lab 2: Simple sorting
* Lab 3: Stacks & Queues
* Lab 4: Linked List
* Lab 5: Recursion
* Lab 6: Trees
* Lab 7: Hash Tables
* Lab 8: Graph
* Lab 9: Exam
* Lab 10: Project Presentation

### There are 8 practical labs (30%):

* Select 3 random submissions to mark
* If you miss a lab or a submission: that lab will be selected to mark

### Lab 9 will be a practical exam (35%)

* You can use your laptop to code
* You are only allow to use the following IDE:
  + NetBeans
  + VS Code
  + BlueJ
  + IntelliJ
* You must DISCONNECT your laptop from the Internet

### Lab 10 is the project presentation (35%)

**Deadline** to submit your work on Blackboard: 3 days from the lab day

* i.e., Lab day is Monday => deadline is Wednesday (mid-night)

### Assignments submission guide

* Create the folder with a name like: ***StudentID\_Name\_Lab#***, (e.g. ***01245\_VCThanh\_Lab1***) to contain your assignment with subfolders:
  + Problem\_01 (sometimes Problem\_i or Problem\_Array)
  + Problem\_02 (sometimes Problem\_ii or Problem\_Queue)
  + etc.
* Compress (.zip) and Submit the whole folder with the same name (i.e., ***01245\_ VCThanh\_Lab1.zip***) to Blackboard
* Students **not** following this rule **will get their marks deducted**

# Lab 2: Simple sorting

## Objectives

* + 1. Know how, in reality, three simple sorting methods work.
    2. Know how to use analysis tool to compare performance of sorting algorithms

## Problem 1: BubbleSortApp.java

* Trace the algorithm (display the array inside after inner or outer loop)
* Display the number of swaps after the inner loop
* Display the number of comparisons after the inner loop and the total number of comparisons, and estimate the algorithms' complexity (n\*(n-1)/2, O(n^2))

## Problem 2: SelectSortApp.java

* Trace the algorithm (display the array after the inner loop)
* Print the items that are swapped. Are swaps always needed?
* Display the number of comparisons after the inner loop and the total number of comparisons, and estimate the algorithms' complexity (n\*(n-1)/2, O(n^2))

## Problem 3: InsertSortApp.java

* Trace the algorithm (display the array after each pass of the outer loop)
* Display the number of passes of the inner loop and total number of passes, and estimate the algorithms' complexity (n\*(n-1)/4, O(n^2))

## Problem 4

Create an array of integer numbers, fill the array with random data and print the number of **comparisons, copies, and swaps** made for sorting 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000 and 50000 items and fill in the table below. Analyze the trend for the three different algorithms.

|  |  |  |  |
| --- | --- | --- | --- |
| **COPIES/ COMPARISONS/ SWAPS** | | | |
|  | **Bubble Sort** | **Selection Sort** | **Insertion Sort** |
| **10000** | Number of Comparisons: 49995000  Number of Swaps: 24708330  Number of Copies: 74124990 | Count Comparison: 49995000  Count Swap: 10000  Count Copies: 51359 | Number of Comparisons: 24704950  Number of Swaps: 9999  Number of Copies: 24714949 |
| **15000** | Number of Comparisons: 112492500  Number of Swaps: 55832018  Number of Copies: 167496054 | Count Comparison: 112492500  Count Swap: 15000  Count Copies: 76777 | Number of Comparisons: 55368821  Number of Swaps: 14999  Number of Copies: 55383820 |
| **20000** | Number of Comparisons: 199990000  Number of Swaps: 99546341  Number of Copies: 298639023 | Count Comparison: 199990000  Count Swap: 20000  Count Copies: 104437 | Number of Comparisons: 98748466  Number of Swaps: 19999  Number of Copies: 98768465 |
| **25000** | Number of Comparisons: 312487500  Number of Swaps: 154519209  Number of Copies: 463557627 | Count Comparison: 312487500  Count Swap: 25000  Count Copies: 129813 | Number of Comparisons: 153299660  Number of Swaps: 24999  Number of Copies: 153324659 |
| **30000** | Number of Comparisons: 449985000  Number of Swaps: 222770017  Number of Copies: 668310051 | Count Comparison: 449985000  Count Swap: 30000  Count Copies: 154843 | Number of Comparisons: 221080580  Number of Swaps: 29999  Number of Copies: 221110579 |
| **35000** | Number of Comparisons: 612482500  Number of Swaps: 302121228  Number of Copies: 906363684 | Count Comparison: 612482500  Count Swap: 35000  Count Copies: 182306 | Number of Comparisons: 303360776  Number of Swaps: 34999  Number of Copies: 303395775 |
| **40000** | Number of Comparisons: 799980000  Number of Swaps: 393090850  Number of Copies: 1179272550 | Count Comparison: 799980000  Count Swap: 40000  Count Copies: 206756 | Number of Comparisons: 397044119  Number of Swaps: 39999  Number of Copies: 397084118 |
| **45000** | Number of Comparisons: 1012477500  Number of Swaps: 500077284  Number of Copies: 1500231852 | Count Comparison: 1012477500  Count Swap: 45000  Count Copies: 232241 | Number of Comparisons: 498989830  Number of Swaps: 44999  Number of Copies: 499034829 |
| **50000** | Number of Comparisons: 1249975000  Number of Swaps: 620658299  Number of Copies: 1861974897 | Count Comparison: 1249975000  Count Swap: 50000  Count Copies: 258080 | Number of Comparisons: 617382166  Number of Swaps: 49999  Number of Copies: 617432165 |

## Problem 5: ObjectSortApp.java (sort the array by first name or by age)

**(Option 1) Given the class Student.java** that has variables of first name, last name, grade

* Add a main() method and add create an array of 10 students
* Add methods to sort the array by first name, last name, and by grade.