CSE-2102
Object Oriented Programming

Dr. Muhammad Ibrahim
Assistant Professor
Dept. of Computer Science and Engineering
University of Dhaka

Administrative Stuff

- Credit hours: 3
  - ~30 classes (1.5 hours each)
- Marks distribution
- Prerequisite
- Course materials: Google Classroom
- Evaluation: assignment, attendance, mid term exam, final exam

Why Study this Course?

- Programming is fundamental to all CSE courses.
- Increases your analytical capability that will be useful for other CSE courses.
- Opens the way to a plethora of jobs.
- If you cannot learn programming in the early stage of your 4 years course, you may never learn it!
- Understanding is important: memorization will not help you much.

Books

Textbook

Java: The Complete Reference by Herbert Schildt (10+th Ed.)

Reference books

- Java How to Program
- Think Java

Lot of other books are available!

Useful reading materials will be given throughout the course in Google Classroom Drive.

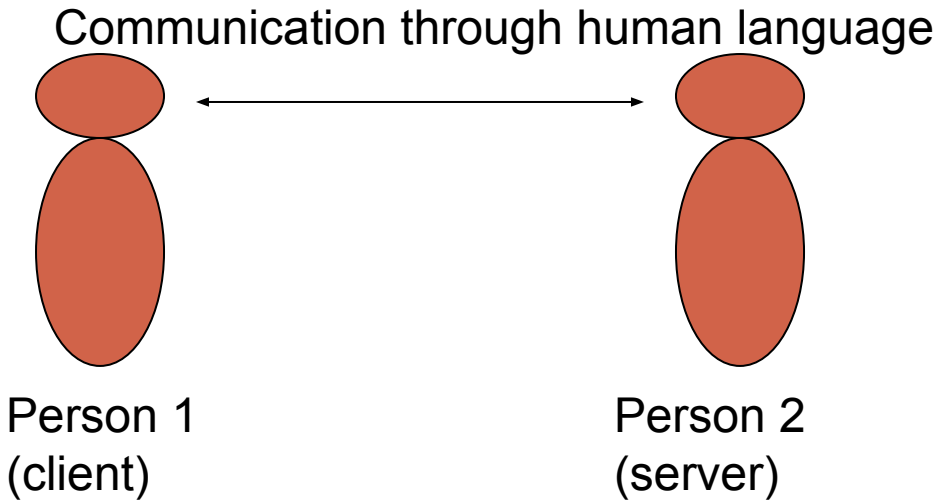
A Short Note: Typing Speed

- Important for every student in CSE.
- Must use 10 fingers for typing!
- Use software to learn *grammatical way* of typing - which fingers to use for which letters.
- Speed: should be at least 35 WPM (words per minute).
- Software: any typing tutor software.

What is Computer Programming?

Computer Programming

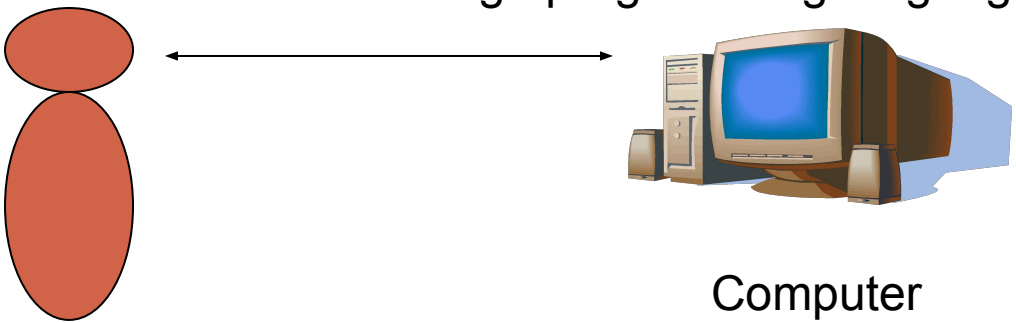
- Computer Programming broadly consists of two things
  - Programming Language
  - Algorithm
- Human language
  - To communicate between two persons



Computer Programming

- Computer Programming broadly consists of two things
  - Programming Language
  - Algorithm
- Programming language
  - To communicate between human and computer

Communication through programming language



Person 1
(client)

Computer
(server)

Computer Programming Contd.

- Suppose somehow the computer understands a limited form of human instructions, which is called Programming Language.
  - *How* - will be discussed in various courses of your entire degree.
- Is only Programming Language enough for the machine to work “intelligently”?
  - A machine (computer) cannot do anything by itself. It can only perform some basic actions prescribed by a human (programmer).
    - Basic actions: addition, subtraction, compare two numbers etc.
  - So how to make a machine to perform a particular work as per the these basic instructions given by a human?

Computer Programming Contd.

- Take an analogy:
  - Suppose you (client) want a cup of tea from a person (server).
  - You say: “Bring me a cup of tea”.
  - If the person (1) understands your language, (2) knows how to make a cup of tea and (3) is not handicapped/insane, he will be able to serve you as per your instruction.
  - However, suppose that the requirement No. 2 is absent in the serving person i.e. the person doesn’t know how to make a cup of tea.
  - So in the first place someone needs to train the person as to how to make a cup of tea. And this is Programming!
    - A programmer trains/teaches a computer as to how to solve a given problem.
    - More specifically, a programmer gives a clearly specified set of instructions to a computer to solve a particular problem.

Computer Programming Contd.

- Analogy Contd.
  - Now, how to train the person to make a cup of tea?
  - Someone (programmer) writes some instructions to the person as follows:
    - Procedure: Making a cup of tea.

      Step 1: Take a cup of water.

      Step 2: Pour the water into a kettle.

      Step 3: Heat the kettle.

      Step 4: If the water is boiled, then pour some tea.

      Step 5: After 2 minutes, bring the kettle back from the stove.

      Step 6: Pour the water into a cup.

      Step 7: Pour some sugar and milk into the water.

      Step 8: Stir the mixture of the cup with a spoon.

      Step 9: Give the cup to the client.

Computer Programming Contd.

- Now can the person (server) make a cup of tea?
  - Yes, but only when the ambiguity present in some of the instructions are eliminated because a machine is stupid - it can only do something if it is told to do so.
  - The corrected form of the procedure is as follows:
    - Procedure: Making a cup of tea.

      Step 1: Take a cup of water.
      Step 2: Pour the water into a kettle.
      Step 3: Heat the kettle.
      Step 4: If the water is boiled, then pour 1 spoon of tea.
      Step 5: After 2 minutes, bring the kettle back from the stove.
      Step 6: Pour the water into a cup.
      Step 7: Pour 1 spoon of sugar and 3 spoon of milk into the water.
      Step 8: Stir the mixture of the cup with a spoon for 30 seconds.
      Step 9: Give the cup to the client.
  - This type of a set of clearly specified instructions to solve a particular problem is called an Algorithm.

Computer Programming Contd.

- Now from the perspective of a machine.
  - A machine understands a programming language. (*how* is not part of this course). No. 1 requirement of slide 12 is fulfilled.
  - A machine can do some basic operations (add, subtract, multiplication, division, comparison, taking input from keyboard, printing output to the screen, storing in and retrieving results from computer memory etc.) No. 3 requirement of slide 12 is fulfilled.
  - The only thing is to fulfill requirement No. 2 of slide 12 i.e. to train the machine to solve a particular problem.
  - Example: next slide…

Computer Programming Contd.

- Suppose we want a machine to perform addition, subtraction, multiplication and division of two numbers. (A simple calculator)
- Procedure: A simple Calculator

  Step 1: Take a number.
  Step 2: Take a sign (+, -, * or /).
  Step 3: Take another number.
  Step 4: If the sign is '+', then add the two numbers and store the result.
  Step 5: If the sign is '-', then subtract the two numbers and store the result.
  Step 6: If the sign is '*', then multiply the two numbers and store the result.
  Step 7: If the sign is '/', then divide the two numbers and store the result.
  Step 8: Display the result.

A machine can take a number and sign but it will not take unless it is instructed.

A machine can add two numbers but it will not add unless it is instructed.

Computer Programming Contd.

Exercise for novice students

- Add from 1 up to some given positive integer *n*.

Solution hint: Use this instruction: repeat from 1 to n

Problem-Solving Steps

How do human solve problems?

In particular, what are the major steps a human takes?

- Step 1: Problem definition
  - Eg. we need an automated grading system.
- Step 2: Precise specification of the problem
  - If marks is more than 80, then we should give grade ‘A’. …. If marks is less than 40, give ‘F’.
- Step 3: Designing the algorithm
  - Eg. refer to the solutions of the previous slides.
- Step 4: Coding
  - According to the algorithm and using a programming language.
- Step 5: Validation and maintenance
  - Try different test case scenarios to check if the system works as expected.
  - In future if new requirements come in, incorporate those requirements.

Concepts Need to be Known

- History of computing and computers
- History of algorithms
- History of programming languages and paradigms
- Compilers, interpreters, linkers

Algorithm: Basic Constructs

- Taking input
- Displaying output
- Storing to and retrieving numbers from memory locations (i.e., variables)
- Basic arithmetic operations like addition, multiplication, comparison etc.
- Conditional statement (JUMP IF value of something is equal/greater/less etc. than something)
- (Unconditional) jump to a specific line of code

It's amazing and surprising that using only these primitive operations, very complex problems can be solved!

- You'll get some taste of the power of these rather primitive instructions when you will use assembly language.
- The real power lies in designing clever algorithms.

History of Algorithms

- Definition: a set of clearly specified instructions to solve a particular problem.
- Named after Muhammad ibn Musa al-Khwarizmi, from Khowarezm (now Khiva in Uzbekistan) in Abbasid empire.
  - A nice description of his work by Uni of Kentucky, USA http://www.ms.uky.edu/~carl/ma330/project2/al-khwa21.html
  - 780-850 C.E. (Common Era)
  - Wrote various books on arithmetic
    - Eg. book on algebra: Hisab al-jabr w'al-muqabala

Medieval Period [edit]

- Before – writing about "recipes" (on cooking, rituals, agric
- c. 1700–2000 BC – Egyptians develop earliest known alg
- c. 1600 BC – Babylonians develop earliest known algorith
- c. 300 BC – Euclid's algorithm
- c. 200 BC – the Sieve of Eratosthenes
- 263 AD – Gaussian elimination described by Liu Hui
- 628 – Chakravala method described by Brahmagupta
- c. 820 – Al-Khawarizmi described algorithms for solving li
- 825 – Al-Khawarizmi described the algorism, algorithms f
  the author's name gave rise to the word algorithm (Latin a
- c. 850 – cryptanalysis and frequency analysis algorithms
- c. 1025 – Ibn al-Haytham (Alhazen), was the first mathem
  development of integral calculus[2]
- c. 1400 – Ahmad al-Qalqashandi gives a list of ciphers in
  cryptanalysis, including the use of tables of letter frequen

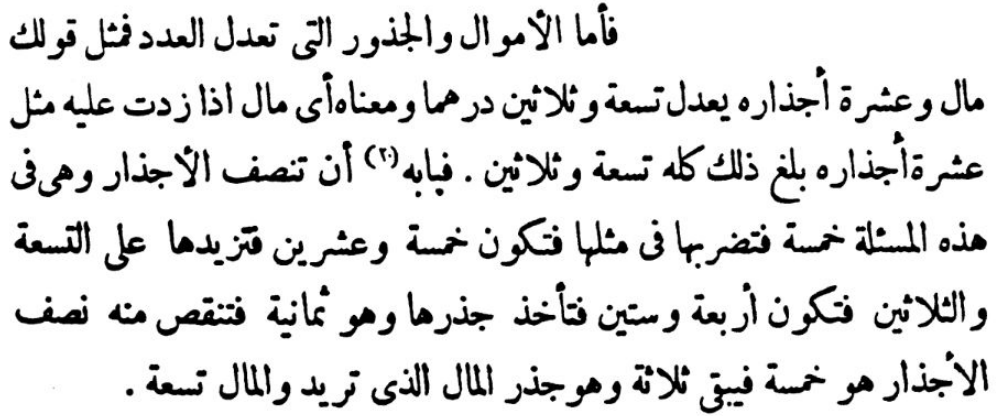
A nice (authentic?) timeline of algorithms from medieval period up to this date: https://en.wikipedia.org/wiki/Timeline_of_algorithms

History of Algorithms

Snippet source: A History of Algorithms by Chabert, Springer, 1999

https://archive.org/details/isbn_0116404090940

Do have a look at this book! Some largely unknown and undiscussed story of algorithms covering thousands of years!



فأما الأموال والجذور التى تعدل العدد فمثل قولك
مال وعشرة أجذاره يعدل تسعة وثلاثين درهما ومعناه أى مال اذا زدت عليه مثل
عشرة أجذاره بلغ ذلك كله تسعة وثلاثين . فبابه[٢٠] أن تنصف الأجذار وهى فى
هذه المسئلة خمسة فتضربها فى مثلها فتكون خمسة وعشرين فتزيدها على التسعة
والثلاثين فتكون أربعة وستين فتأخذ جذرها وهو ثمانية فتنقص منه نصف
الأجذار هو خمسة فيبقى ثلاثة وهو جذر المال الذى تريد والمال تسعة .

As for squares and roots equal to a number, it is as when you say this: a square and ten of its roots equal thirty-nine dirhams.
Its meaning is that the square, if you add to it the equivalent of ten of its roots [is such that] it will become thirty-nine.
Its method [of solution] consists in dividing the roots by two, and that is five in this problem. You multiply it by itself and this will be twenty-five. You add it to thirty-nine. This will give sixty-four. You then take its square root which is eight and you subtract from it half [the number] of the roots and that is five. There remains three and that is the root that you are seeking and the square is nine.

from Al-Khwārizmī, *al-Jabr wa l-Muqābala* (*c.* 860).
The text describes an algorithm for solving certain types of quadratic equations through using the example $x^2 + 10x = 39$.

Algorithm Design: Top-Down Approach

- Start with general steps.
- Gradually elaborate the steps until they become pretty straightforward
  - Recall the slide when we talked about the basic operations a computer can perform.
- Example:
  - “Bring me a cup of tea”
    - Step 1: organize the kitchen
    - Step 2: prepare the tea
    - Step 3: serve.
  - Now break down each of the above tasks
    - Step 1.1: clean kitchen surface
    - Step 1.2: locate cups, stove etc.
    - Step 2.1: boil water
    - Step 2.2: pour tea.
    - And so on.

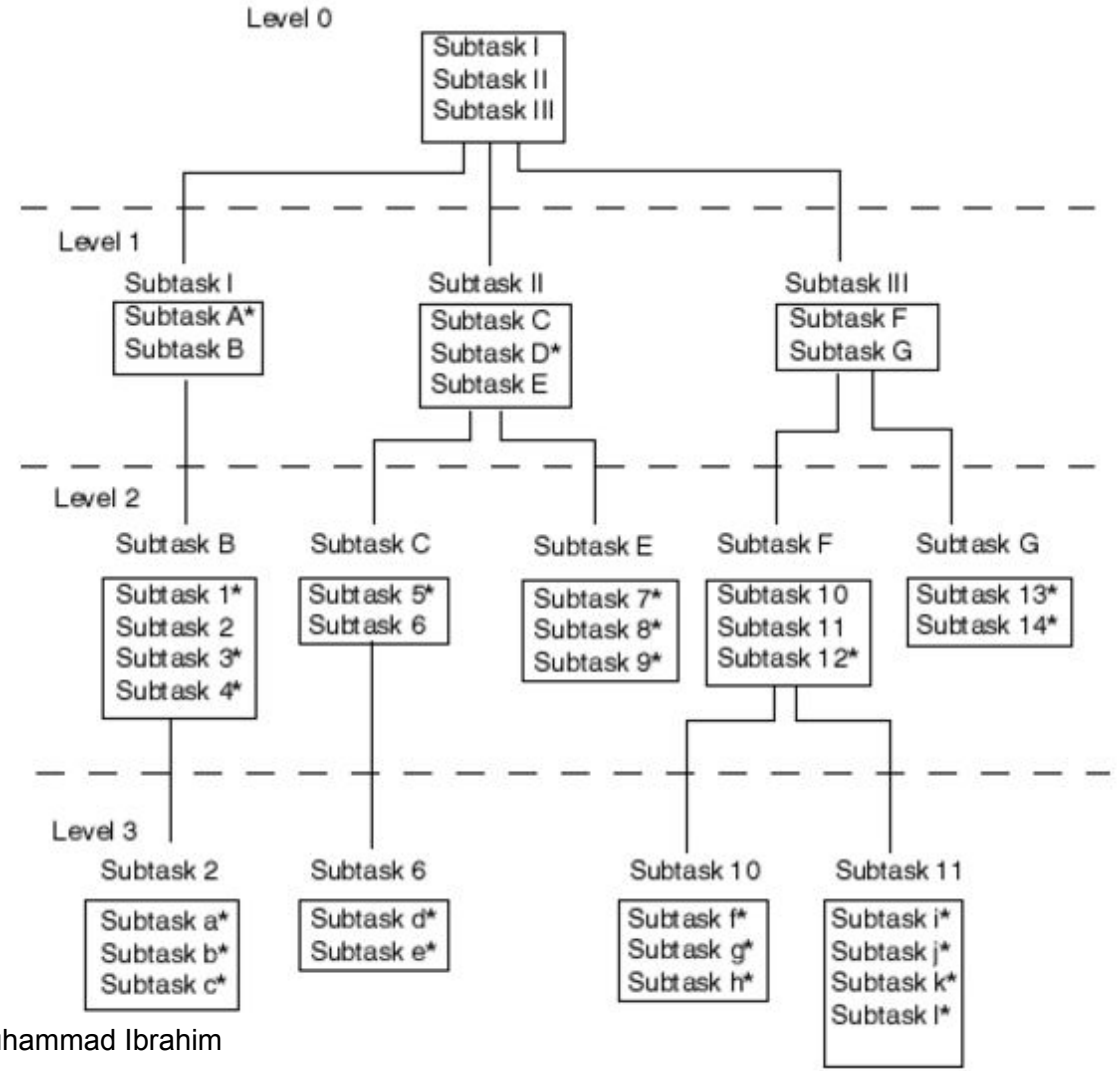


Image source: http://www.eecs.wsu.edu/~cs150/tdd.htm

- Problem: Preparing an automated admission system for a university.
  - Step 1: collect candidate students' data
  - Step 2: list the eligible candidates
  - Step 3: organize admission test
  - Step 4: select the top students
- One level down:
  - Step 2.1: check if the student's marks in the previous exam surpass the threshold
  - Step 2.2: check if the student has paid the required fee
  - Step 3.1: arrange seat plan
  - Step 3.2: prepare question papers
  - Step 3.3: take the test
  - Step 3.4: evaluate answer scripts.
  - Step 4.1: get the number of available seats in the university.
  - Step 4.2: select the top students according to the available seats.
  - Step 4.3: announce the result.

Concepts Need to be Known

- How a digital computer works
  - Motivation: need to have an all-purpose/universal (in today's term "programmable") machine
    - Implies intelligent machine because resembles human
  - Principles
    - Algorithm: step by step breaking down of a problem
    - Abstraction/modularity: build the current module assuming something underneath working perfectly.
  - Tools
    - Boolean logic
    - hardwares (switching circuit for processor and electrical/magnetic devices for memory)
    - and eventually software
- History of programming languages
  - Different programming paradigms
    - Need, market
  - Chronology of programming languages
  - Why some languages stand out while others fade away
- How interpreter, compiler, linker and loader works
  - Toggle-switch based programming -> machine code but manual -> machine code automated (batch) -> translator/interpreter/compiler and mid-level language -> high-level language (and of course, compiler).

Common Programming Paradigms

- Procedural
- Logic
- Functional
- Object oriented

Can be structured or unstructured

How do we get to OOP?

- Evolution of programming
  - In the beginning, writing codes using machine language directly.
    - Very small programs.
  - Then, assembly language (1940s)
    - Moderate sized programs.
  - Then, high-level language. Eg. Fortran. (mid 1950s)
    - Moderately complex programs. Thousands of lines.
  - However, even with high-level languages, larger programs became unmanageable.
    - E.g. “Spaghetti code” problem due to the use of GOTO statement.
  - As a result, structured programming paradigm was invented.
    - Relies on well-defined control structure, code blocks, and almost absence of GOTO statement, modular design (with reusable function modules).
    - Algol, Pascal, C etc. in late 1960s.
    - Very complex, large programs.

How do we get to OOP? (Contd.)

- So why do we need yet another paradigm (OOP in late 1970s)?
  - Because the required softwares grew even bigger in terms of size and complexity, which once again became unmanageable using the languages like C.
  - OOP takes the best of the ideas of structured programming, and adds more.
- In OOP, the world is seen as a number of objects, where each object is a self-constrained entity with its own data and actions/functions.
  - Thus programmers can manage the complexity of large programs in an easier way.
- Three main concepts of OOP are: (these concepts will be thoroughly discussed as we go through the course)
  - Encapsulation
  - Polymorphism
  - Inheritance.