Unit 1: Computer Concept & 1st Java program

**Objectives**
1. Identify problems that computers can solve
2. Describe the basic components of a computer
3. Describe the relationship between hardware and software
4. Distinguish between interpretation and compilation
5. Describe and use the Internet and World Wide Web
6. Describe the Java program development and execution model
7. Install and use the Java Software Development Kit (please refer to my notes of Tutorial-1; 201-00.ppt)

**Computer solutions vs manual solutions**

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>A computer is a machine that can calculate. Manual calculation has higher change of error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>A computer obviously performs calculations much faster than we could do manually.</td>
</tr>
<tr>
<td>Availability &amp; Convergence</td>
<td>Nowadays computer is an easy-to-access tool in our daily life.</td>
</tr>
<tr>
<td>Automation</td>
<td>A computer can be set to perform tasks automatically without human intervention. Machines that can follow a predetermined sequence of steps are known as automatons.</td>
</tr>
<tr>
<td>Cost effective</td>
<td>A computer solution may have higher cost-saving (in many senses) than a manual one.</td>
</tr>
</tbody>
</table>

**Data, Information and Algorithm**

<table>
<thead>
<tr>
<th>Data</th>
<th>The raw materials we need to carry out a task (analogue vs. digital)</th>
</tr>
</thead>
</table>
| Information | Meaningful result obtained after processing the data  
words, numbers and graphics used as the basis for human actions and decisions  
eg. flight data on the screen that a person uses it to complete actions or making decisions |
| Algorithm | In order to obtain the required result a well-defined formula or sequence of steps is applied to the data. Such a formula or sequence of steps is known as an algorithm |

**Problems that computer find it hard to solve**
- Computers need algorithm(s) to solve problems,  
  - because they are just machines for performing calculations
- Computers cannot derive algorithms themselves  
  - They must rely on human to feed in algorithms
- For some difficult problems with no efficient algorithm to solve, trial-and-error approach is necessary.  
  - e.g. traveling salesman problem  
  - Visiting all locations only ONCE and return to the starting point

**The history of computers**
1. Calculating devices - abacus, Napier Bones, Slide rule, Pascalene (mechanical adding machine)
2. Programmable computers - devices that can accept sequences of instructions (programs) and perform them  
   - Early computers: mechanical binary switches  
   - First generation modern computers: vacuum tubes  
     - unreliable, hot, huge in size  
   - Second generation: transistors (semi-conducting material)  
     - smaller, faster, consume less power  
   - Third generation: Integrated Circuit (IC)  
     - transistors and others in a circuit  
     - LSI, VLSI, ULSI
3. Personal Computer (PC): introduced by IBM

**Computer Hardware (h/w)**
- The components that you can physically touch in a computer are the hardware of the computer; there are:
  1. Special-purpose computer:  
     - a device with components that a usual computer possesses but is designed for specific purposes  
     - washing machine, microwave oven, air-conditioner, rice cooker
  2. General-purpose computer:  
     - can be used for multiple purpose  
     - desktop, notebook, handheld

<table>
<thead>
<tr>
<th>Types</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal Computers (microcomputers):</td>
<td>single-user</td>
</tr>
<tr>
<td>Workstations:</td>
<td>single-user, more powerful</td>
</tr>
<tr>
<td>Mini-computers:</td>
<td>multi-user, multi-processor</td>
</tr>
<tr>
<td>Mainframes:</td>
<td>organizational or enterprise-wise</td>
</tr>
<tr>
<td>Supercomputers:</td>
<td>extremely high speed, very expensive</td>
</tr>
</tbody>
</table>
Data Representation

- An Analog device operates on continuously varying data.
- A Digital device works with discrete number.
  - Which has only 2 digits: [0] and [1]
- "Character data" composed of letter, symbol, & numeral that will not be used in arithmetic operations.
- Digital computers represent character data using codes and "Code Set".
  - ASCII: American Standard Code for Information Interchange
  - EBCDIC: Extended Binary-Coded Decimal Interchange Code
- Most micro-computers use ASCII code to represent character data.
- Some mini, mainframe computers use EBCDIC code to represent character data.

Number Systems

A decimal number 137 (assumed base 10) is

\[ 137_{10} = 1 \times 10^2 + 3 \times 10^1 + 7 \times 10^0 \]

\[ = 100 + 30 + 7 \]

\[ = 137 \]

A binary number 10001001₂ (base 2) is

\[ 10001001₂ = 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \]

\[ = 128 + 0 + 0 + 0 + 8 + 0 + 1 \]

\[ = 137_{10} \]

An octal number 211₈ (base 8) is

\[ 211₈ = 2 \times 8^2 + 1 \times 8^1 + 1 \times 8^0 \]

\[ = 128 + 8 + 1 \]

\[ = 137_{10} \]

A hexadecimal number 89₁₆ (base 16) is

\[ 89₁₆ = 8 \times 16^1 + 9 \times 16^0 \]

\[ = 128 + 9 \]

\[ = 137_{10} \]

Conversions

- Given a decimal number 137, after repeated division by 2, we have 10001001 by reading the digits bottom-up

Von Neumann architecture

- Fetch-Decide-Execute cycle
  - The control unit fetches one instruction (encoded in binary format) at a time from the memory; then decodes it and executes it.

Central Processing Unit (CPU)

- The circuitry in a computer that execute instruction to process data
- It takes instruction/data from RAM, process instructions then places the results back into RAM

A simplified general-purpose computer

To Octal (take 3 bits)

\[
\begin{array}{c|c|c|c}
\hline
\text{Decimal} & \text{Binary} & \text{Octal} & \text{Hexadecimal} \\
\hline
0 & 000 & 0 & 0 \\
1 & 001 & 1 & 1 \\
2 & 100 & 2 & 2 \\
3 & 101 & 3 & 3 \\
4 & 110 & 4 & 4 \\
5 & 111 & 5 & 5 \\
6 & 1000 & 6 & 6 \\
7 & 1001 & 7 & 7 \\
8 & 1010 & 10 & A \\
9 & 1011 & 11 & B \\
10 & 1100 & 12 & C \\
11 & 1101 & 13 & D \\
12 & 1110 & 14 & E \\
13 & 1111 & 15 & F \\
\hline
\end{array}
\]

- Then the decimal number 137 can be converted to any bases by grouping different number of BITs from the Right end

- To Hexadecimal (take 4 bits)

\[
\begin{array}{c|c|c|c|c}
\hline
\text{Decimal} & \text{Binary} & \text{Octal} & \text{Hexadecimal} \\
\hline
0 & 0000 & 0000 & 0000 \\
1 & 0001 & 0001 & 0001 \\
2 & 0010 & 0010 & 0010 \\
3 & 0011 & 0011 & 0011 \\
4 & 0100 & 0100 & 0100 \\
5 & 0101 & 0101 & 0101 \\
6 & 0110 & 0110 & 0110 \\
7 & 0111 & 0111 & 0111 \\
8 & 1000 & 1000 & 1000 \\
9 & 1001 & 1001 & 1001 \\
10 & 1010 & 1010 & 1010 \\
11 & 1011 & 1011 & 1011 \\
12 & 1100 & 1100 & 1100 \\
13 & 1101 & 1101 & 1101 \\
14 & 1110 & 1110 & 1110 \\
15 & 1111 & 1111 & 1111 \\
\hline
\end{array}
\]

Central Processing Unit (CPU) (out syllabus)

- The data bus transports data and instructions between RAM and the CPU.
- The CPU processes the data in RAM and places the results back into RAM.
CPU has 2 main parts:

1/. Arithmetic Logic Unit (ALU),
- performs arithmetic operations such as addition & subtraction
- performs Logical operations such as comparing 2 numbers to check if they are the same
- Use registers to hold data that is being processed
- The result of the ALU will be kept in the accumulator.

2/. Control Unit (CU),
- direct & control processing
- Use instruction pointer to keep track of instruction sequence
- Retrieve each instruction in sequence from RAM and place it in instruction register
- Interpret the instruction
- and may send signal to data bus:
  - fetch data from RAM
  - to the ALU to perform a process

Instructions
- tells the computer to perform arithmetic, logical, or control operation
- It has 2 parts
  - [op code] = "operation code", a command word for an operation
    - Such as add, compare, or jump
  - [operands] specify the data or address of the data for the operation
- [Instruction cycle] as, shown in margin, refers to the process that a computer executes a single instruction.

Instruction Set Complexity
- CISC (Complex Instruction Set Computer)
  - Complex instructions are provided in this kind of machine, however
  - 20% of the instructions of a CISC machine do about 80% of the work
- RISC (Reduced Instruction Set Computer)
  - Has limited set of instructions that it can perform very quickly
  - In theory a RISC machine is faster than CISC machine for most tasks
  - Scientists believe that a balance or hybrid of CISC & RISC technologies produces the most efficient and flexible computer.

Pipelining
- To speed up the processing
  - A processor can begin executing part of the next instruction(s) before the current instruction completed, as shown below
  - Instruction-1 load process store
  - Instruction-2 load process store
  - Instruction-3 load process store

Parallel Processing
- these are Parallel Computers
- Increase the amount of processing that a computer can accomplish in a specific amount of time

Peripherals (Input/Output Devices)
- keyboard
- mouse
- monitor
- Printer

Input & Output (IO)
- Refers to collecting data for the microprocessor to manipulate, and transporting results to display, print and storage devices
  - Expansion Bus is a segment of the data bus that transports data between RAM and peripheral devices
  - IO architecture transports data to and from external components to peripheral devices.

(out syllabus)
Expansion Slots & Cards

- **Expansion Slot** is a long, narrow socket on the motherboard, where you can plug your expansion cards.
- **Expansion card / controller card** is a small circuit board that provides a computer with the ability to control a storage, input, or output device.

**Examples**
- Sound card
- Modern card
- LAN card
- Capture card
- AGP graphic card

AGP, PCI, and ISA slots are different lengths, so you can easily identify them by opening your computer’s system unit & looking at the motherboard.

Memory

- **ROM** - Read Only Memory
- **RAM** - Random Access Memory

**size measurement**
- Binary digiT (bit)
- byte=8-bit
- KB=1000 Bytes, kB=1024 Bytes,
- Kb =1000 bits, kb =1024 bits
- MB=1000 Bytes, mB=1024 Bytes
- GB
- TB

Storage Devices (Secondary Storage)

- floppy disk
- hard disk
- magnetic tape
- memory stick, compact flash, USB disk

Disks | Full name | Nature
--- | --- | ---
CD-ROM | Compact Disc Read-Only Memory | read only
CD-R | Compact Disc Recordable | write once, disallow re-write; many allow multiple sessions
CD-RW | Compact Disc Rewritable | write many; Use phase change technology
Computer Software (s/w)

Operating Systems (OS)
- manage and coordinate computer resources (examples?)
- provide security features
- support multi-users to perform different tasks concurrently
- provide GUI

Application software
- allow computer to perform specific tasks (e.g. word-processing)
- must match operating system to work properly
- cannot manipulate hardware directly and must access them using the services provided by OS (hence a 3-layer architecture)

Application Software
- allow computer to perform specific tasks (e.g. word-processing)
- must match operating system to work properly
- cannot manipulate hardware directly and must access them using the services provided by OS (hence a 3-layer architecture)

OS interacts with h/w & applications

Programming Languages (PL)

Low-level languages (LL)
- machine language - native language understood directly by computer (but language different for different computers)
- assembly language

High-level languages (HL)
- use human-language-like statements (easy for us to read and understand)
- 4GL - usually used for accessing database (SQL)
- 5GL - support declarative programming (e.g. in AI area)

Notice: there is a gap between LL and HL languages

Difference between levels of PL

<table>
<thead>
<tr>
<th></th>
<th>Low-level language</th>
<th>High-level language</th>
</tr>
</thead>
<tbody>
<tr>
<td>speed</td>
<td>faster</td>
<td>slower (code not optimal after translation)</td>
</tr>
<tr>
<td>portability</td>
<td>impossible</td>
<td>good (with no or minor modifications)</td>
</tr>
<tr>
<td>readability</td>
<td>difficult to learn, read or modify</td>
<td>easy to learn, read or modify</td>
</tr>
<tr>
<td>selectivity</td>
<td>no choice</td>
<td>plenty of choices (e.g. C, C++, Java, etc.)</td>
</tr>
</tbody>
</table>

How programs are executed?
- How to fill up the gap between LL and HL languages? i.e. How to make the computer understand a HL language?

Compilation
- convert the HL language (source code) to machine language (object code or machine code)
- execute the machine code (in binary format)

Interpretation
- use a software interpreter to execute the source code

Hybrid
- compilation process translates the source code to an intermediate code, and the interpreter executes it (e.g. Java)

Compile & Compilation
- Compiling refers to the act of turning human-readable source code into machine-readable binary code
- Compiling includes in itself various phases.

Interpretation vs Compilation

<table>
<thead>
<tr>
<th></th>
<th>Interpretation</th>
<th>Compilation</th>
</tr>
</thead>
<tbody>
<tr>
<td>speed</td>
<td>slower (translation during execution)</td>
<td>faster (compile only once)</td>
</tr>
<tr>
<td>execution model</td>
<td>need source code for real-time translation</td>
<td>need only object code</td>
</tr>
<tr>
<td>portability</td>
<td>need an interpreter on each platform</td>
<td>need compilation once for each platform</td>
</tr>
<tr>
<td>reliability</td>
<td>less reliable (check on-the-fly)</td>
<td>more reliable (check by compiler)</td>
</tr>
<tr>
<td>intellectual property</td>
<td>weak protection</td>
<td>strong protection</td>
</tr>
</tbody>
</table>
The Internet

Local Area Network (LAN)
- Computers connected to form a network for communication and sharing some common resources (such as printers).

The Internet - network of networks
- The original aim was for academic purposes
- common services include WWW, email, FTP, telnet, ICQ, online games

World Wide Web
- Every machine on the Internet has a unique numeric network address (IP address).
- URL specifies the location of a document in the Internet (web server).
- The collection of Web pages provided by an organization or company is a website.

Object-oriented (OO) programming paradigm
- Researchers found that modelling the real world entities was a better way to develop software.
- From the perspective of writing programs, a problem can be solved by first determining the involved agents (objects) and the messages that are sent among them. Then each agent is concentrated and programs written to perform all the determined operations.
- It is a more natural methodology for solving problems. Software developed using object-oriented programming is more robust, easier to debug and maintain, and it is easier to reuse the components built.

Hyper-Links & Websites

<table>
<thead>
<tr>
<th>Description</th>
<th>Hyperlinks</th>
</tr>
</thead>
<tbody>
<tr>
<td>My testing Servers</td>
<td>http:// KY.GotDNS.Com</td>
</tr>
<tr>
<td></td>
<td>http:// KY.GotDNS.Com/index.htm</td>
</tr>
<tr>
<td></td>
<td>http:// KY.GotDNS.Com/040110_Moscow%20Circus/index.htm</td>
</tr>
<tr>
<td></td>
<td>FTP:// KY.ServeFTP.Org</td>
</tr>
<tr>
<td>MT-201 Tutorial Page</td>
<td>http:// learn.ouhk.edu.hk/~t441063</td>
</tr>
<tr>
<td>Personal Home Page</td>
<td><a href="http://slinux.ouhk.edu.hk/~itkyuen">http://slinux.ouhk.edu.hk/~itkyuen</a></td>
</tr>
</tbody>
</table>

Java History
- 1991 - programming language Oak (OO) was invented for handheld devices in home entertainment
- The direction was shifted to Internet and the name changed to Java
- 1995 - a Java applet was written to demonstrate its capability of running an application within a Web page
- Java is a hybrid language involving both the object-oriented and imperative features.
- It is hence up to the developers to write programs with object-oriented methodology or not.

Advantages of using Java
- It is a real programming language and software development kits (SDKs) have been made available free of charge.
- It is platform independent (for platforms having SDK or Java runtime environment JRE - which include a Java interpreter). It features "write once, run everywhere".
- Plenty of development tools are available and most of them are open source or free of charge.
- It has a large user community and is hence enhanced rapidly.
- Its syntax is more restricted and the resulting program is more reliable and robust.
- It was developed for use in the Internet and hence naturally supports security by various means (i.e. much safer to use).

Imperative (procedural) programming paradigm
- Program statements/instructions are executed one-by-one and applied with the data.
- Computer hardware implementations are basically imperative (which is the execution model of von Neumann architecture), and so are their low-level programming languages.
- As a consequence, most early high-level programming languages were also imperative.
- Programming languages like Basic, Pascal, C, etc. are typical imperative languages.

Pitfalls of imperative programming paradigm
- It emphasizes the detailed steps (algorithms) for solving a problem. Programmers are distracted because they have to design the solution as a high-level abstraction and low-level implementation at the same time.
- There is little support for programming-in-the-large and team development. This makes system development and maintenance difficult, especially for large-scale and complex software.
- Programs written are mostly problem specific. There are no facilities for enhancing software reuse, and reusing written programs is not an easy task.

Java execution models

Application
- execute as a standalone software application
- need to install JRE and the application in bytecode (intermediate object code)

- pros: no operational restrictions (i.e. a Java application can access any resource in the machine)
- cons: need to reinstall the application to user’s machine every time the software is updated
Applet
- execute inside a Web browser (which is equipped with JRE)
- when the browser is used to view a Web page with embedded Java applet, the browser will download the bytecode into user’s machine and run it immediately
- **pros:** when the Java applet is updated, the browser will detect the changes and download the modified files automatically
- **cons:** JRE for applet is customized not to allow operations that are potentially a security threat (because the software is downloaded from Internet)
- Java Web Start - start a Java application by clicking a hyperlink in a Web page or an icon on the desktop

Differences among Java packages

<table>
<thead>
<tr>
<th></th>
<th>JVM</th>
<th>JRE</th>
<th>JDK/SDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>JVM emulator</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>The standard Application Programming Interface (API library)</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>compiler &amp; other supporting tools</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

JavaScript Is NOT Java
- Java applications can stand alone.
- **JavaScript is interpreted** while **Java is compiled.**
- Their development tools are different, and they have a surprisingly different audience
- Please refer to the table (next page) for their differences

JavaScript and Java

<table>
<thead>
<tr>
<th><strong>JAVASCRIPT</strong></th>
<th><strong>JAVA</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpreted by client</td>
<td>Compiled by the author, run on client</td>
</tr>
<tr>
<td>Code integrated in HTML documents</td>
<td>Applets distinct from HTML document</td>
</tr>
<tr>
<td>Loose typing of data types</td>
<td>Strong typing of data types</td>
</tr>
<tr>
<td>Dynamic binding</td>
<td>Static binding</td>
</tr>
<tr>
<td>Script limited to browser functions</td>
<td>Stand-alone applications</td>
</tr>
<tr>
<td>Works with HTML elements</td>
<td>Goes beyond HTML (for example, multimedia)</td>
</tr>
<tr>
<td>Access browser objects and functionality</td>
<td>Limited access to browser objects or functionality</td>
</tr>
</tbody>
</table>