# CS101
# Computer Programming I

Chapter 1- Introduction to Computers and Java

Lecture 1

# Topics

1.1 Introduction

1.2 Computers: Hardware and Software

1.4 Computer Organization

1.5 Computer Languages

1.6 Introduction to Object Technology

1.9 Programming Languages

1.9 Java and a Typical Java Development Environment

# 1.1 Introduction

- Java, developed by Sun Microsystems, is one of today’s most popular languages for developing software.
- You’ll learn structured programming and an exciting newer methodology, **object-oriented programming**. (Why both?)
- Java has become the language of choice for implementing Internet-based applications and software for devices that communicate over a network.

- Java has two other editions:
  - The Java Enterprise Edition (Java EE)  for developing large scale distributed networking applications and web-based applicatio
  - The Java Micro Edition (Java ME)  for developing applications for small, memory constrained devices (cell phones and PDAs)

# 1.2 Computers: Hardware and So

t A **computer** is a device that can perform computations & make logical decisions faster than humans can.

t **Computer program:**

- A set of instructions for a computer to follow.

t **Computer software:**

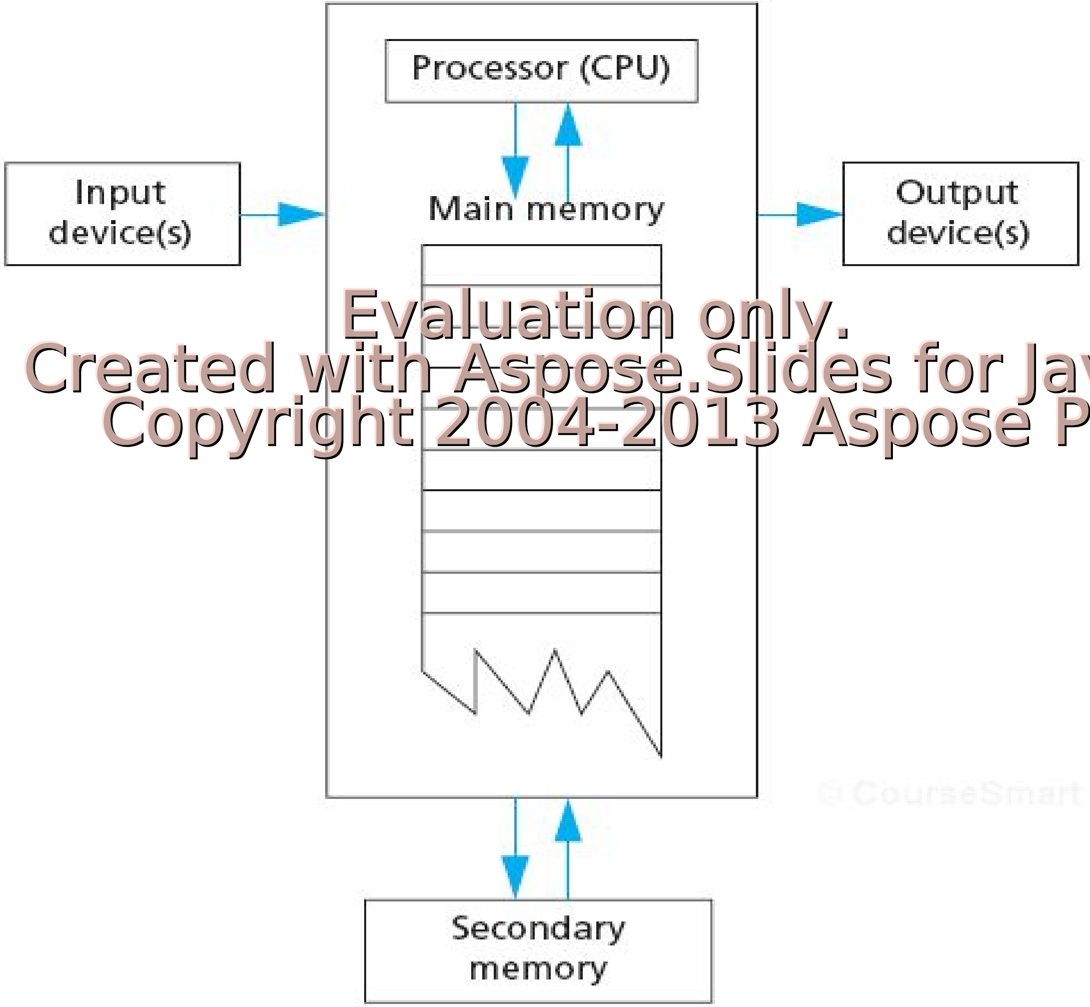
- The collection of programs used by a computer.

t **Computer hardware:**

- The actual physical machines that make up a computer.

| Software | Hardware |
|---|---|
| •The parts that can’t be touched are called “Software”.<br>Characteristics: It has no physical existence.<br>•They can be realized through *only* mind and feeling<br>•E.g. translators, editors, programs. | •Things that can be touched are called “Hard<br>•Characteristics: They have physical existence, we can be damaged or broken.<br>•E.g. monitor, mouse, printer. |

# 1.4 Computer Organization



# 1.4 Computer Organization (Co

## Five main components

1. Input unit
   - Allows a person to communicate data to the computer.
   - Obtains information from input devices for other units so that it can be proce
   - Examples?

2. Output unit
   - Allows the computer to communicate information to the user.
   - Takes information that the computer has processed and places it on various
3. Memory (Main Memory/ Primary Memory)
   - Often referred to as RAM (Random Access Memory) since the computer can immediately access the data in any memory location.
   - Contains information that has been entered through the input unit to make it available for processing when needed
   - Also retains processed information until it can be placed on output devices b
   - Volatile  information is lost when power is off.

# 1.4 Computer Organization (Co

## 4. Processor (CPU)

t The CPU (Central Processing Unit) is the brain of the computer.

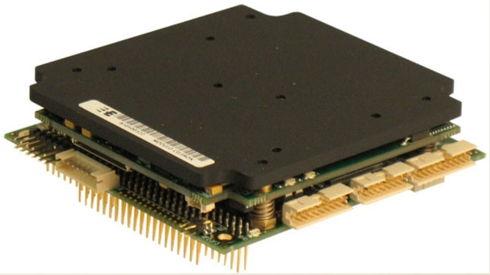
t Coordinates and supervises the operation of other sections.

t The CPU follows the instructions in a program and performs the calculations specified by the program.

t The processor of a modern computer can have as many as several hundred availa

t Consists of two main units:

- The Arithmetic and Logic Unit (ALU)
- The Control Unit (CU)



# 1.4 Computer Organization (Co

**5. Secondary Memory/Storage**

t Main memory is only used while the computer is actually following the instruction

t **Secondary memory** is the memory that is used for keeping a permanent record of info

t Information on secondary storage devices is said to be persistent, it is preserved even when the computer's power is turned off.

t Alternative terms: secondary storage, auxiliary memory/ storage, external memo

t Information in secondary storage takes much longer to access than information in primary memory, but the cost per unit of secondary storage is much less than that of primary mem

t Examples: hard disk, CD's, DVD's, Flash memory drives, etc.

# 1.5 Computer Languages

t There are many languages for writing programs. The different categories include:

- Machine Languages
- Assembly Languages
- High-level Languages

# 1.5 Computer Languages: Machine Languages

- t Programs written in the form of 0's and 1's are said to be written in **machine language**
- t This is because it is the version of the program that the computer can directly read and follow.

- t Machine languages are *machine dependent* (what does this mea
- t The difference between assembly and machine language is not important, they are almost the same.
- t However the important distinction is between machine-languag and high-level language like Java

# 1.5 Computer Languages: Assembly Languages

t Instead of using strings of numbers, programmers began using English-like abbreviations to represent elementary oper

t These abbreviations formed the basis of assembly language

t A low-level command such as: ADD X, Y, Z

means add the values found at x and y in memory, and store the result in location z.

t It must be translated to machine language (zeros and ones) before the computer can understand it.

0110 1001 1010 1011

# 1.5 Computer Languages: High-level Languages

- High-level languages were developed in which single statements could be written to perform a substantial task.
- Common programming languages include: C, C++, Java, Visual Basic, etc.

- Characteristics
  - Resemble human languages.
  - Designed to be easy to read and write.
  - Use more complicated instructions than the CPU can follo
  - Must be translated to zeros and ones for the CPU to execute a program (by a *compiler*).

# An Overview Of Computer Languages

| Machine Language | Assembly Language | High Level Language |
|---|---|---|
| Collection of binary numbers | Symbolic form of machine language (I.e. symbolic names are used to represent operations, registers & memory locations | Combines algebraic expressions & symbols taken from English language (ex. C++, Pascal, FORTRAN, etc) |
| Ex.<br>10100001 00000000 00000000<br>00000101 00000100 00000000<br>10100011 00000000 00000000 | Ex.<br>MOV AX, A<br>ADD AX, 4<br>MOV A, AX | Ex.<br>A = A + 4 |
| Directly understood by a computer but cumbersome for humans | **Assembler** → converts to machine language | **Compiler** (or interpreter) → converts to machine language |

Easier to use

More Powerful

# 1.6 Introduction to Object Technol

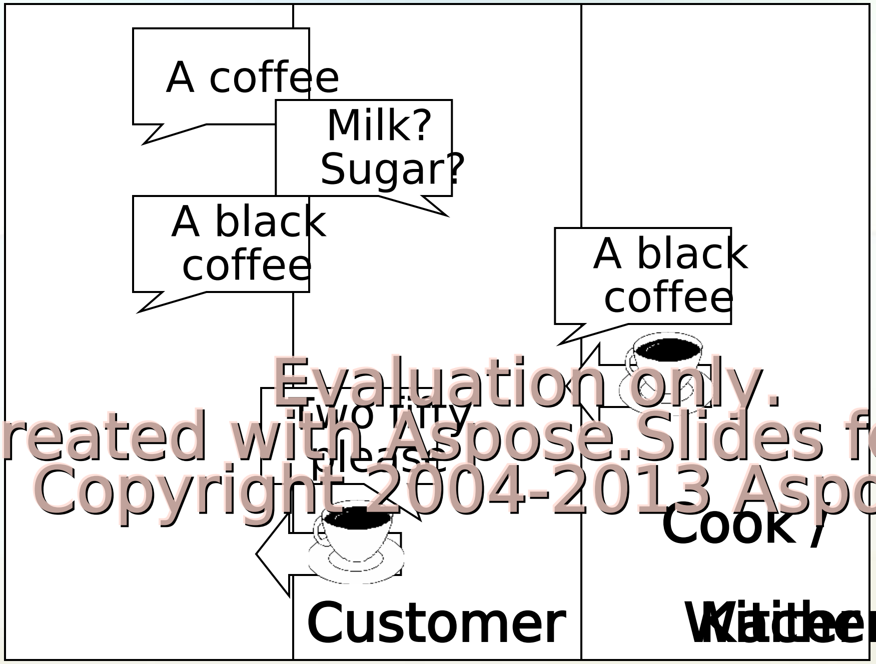
t Object Oriented Programming OOP has taken the best ideas of procedural (structured) programming and has combined them with several powerful concepts that allows us to organize our programs more effectively.

t The benefits of OOP are higher for complex programs

t All OOP languages have three characteristics:

- Encapsulation.
- Polymorphism.
- Inheritance.

# 1.6 Introduction to Object Technol
# The 'Cup Of Coffee' Example



t Customer and kitchen/cook don't know each other. The waiter is intermediary. (**encapsulation**)

t Waiter and kitchen/cook act differently to the request 'a black coffee'. (**polymorphism**)

t Both waiter and kitchen/cook supply coffee. (**inheritance**)

# 1.6 Introduction to Object Technolo
# Encapsulation

- Encapsulation means ‘hiding’ information.
- Objects contain their own data and algorithms.
- Encapsulation keeps program flexible. Suppose the waiter will brew the coffee himself. The customer won’t notice ar

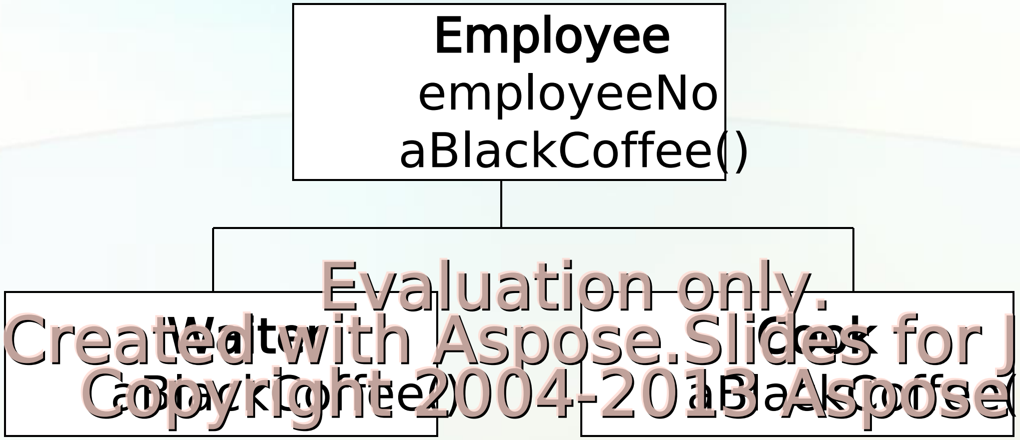
# 1.6 Introduction to Object Technolo
# Polymorphism

t A single name with multiple meanings (depen on its context), this is polymorphism.

t Polymorphism reduces complexity by allowing the same name to be used to specify multiple meanings. It is the compiler's job to select the specific action as it applies to each situation. The programmer need not do this selection manually.

# 1.6 Introduction to Object Technolo
# Inheritance

t Objects can inherit characteristics from other objects.



Employee
employeeNo
aBlackCoffee()

Waiter
aBlackCoffee()

Cook
aBlackCoffee()

t Both waiter and cook are employees. So they both have an employee number. (inherits it from Employee)

t Both return a cup of coffee to the request ‘a black coffee’. However, There are some exceptions. Waiter and cook have different methods to get a cup of coffee.

t Without the use of hierarchies, each object would have to explicitly define all of its characteristics.

t Using inheritance, an object needs to define only those qualities that make it unique within its class.

# 1.8 Programming Languages: History of C and C++

- There was a B programming language derived from B
- The C language was derived from the B language.
- The C++ language was derived from the C language.

# 1.9 Java and a Typical Java Develop Environment

t In 1991, Sun Microsystems funded an internal corporate research project which resulted in a C++-based language.

t Its creator, James Gosling, called Oak after an oak tree outside his window at Sun.

t The name Oak was already in use so the name Java was suggested by a group of Sun people.

t Sun saw the potential of using Java to add dynamic content to web pages(interactivity and animation) with the web popu

# Characteristics of Java

- t Java is simple
- t Java is object-oriented
- t Java is distributed
- t Java is interpreted
- t Java is robust
- t Java is secure
- t Java is architecture-neutral
- t Java is portable
- t Java’s performance
- t Java is multithreaded
- t Java is dynamic

# 1.9 Java and a Typical Java Develop
# Environment: Java Class Libraries

t Java programs consist of pieces called **classes**.

t Classes include pieces called **methods**.

t Methods perform tasks and return information when the tasks complete.

t Java class libraries are rich collections of already existing classes. Also called **Java APIs (Application Programming Interface**.

t You have to learn both the Java language itself and the classes in the Java class libraries.

# 1.9 Java and a Typical Java Develop Environment (Cont.)

t Java programs normally undergo five phases

1. Edit
   - Programmer writes program using an editor program (and stores program on disk)
   - A file name ending with .java extension indicates that the files contains Java code.
2. Compile
   - Compiler creates bytecodes from Java source code
   - Bytecodes are executed by the Java Virtual Machine (JVM) a part of the JDK.
   - Java's bytecodes are portable, without recompiling the source code, the same bytecodes can execute on any platform containing a JVM that understands the version of Java on which the bytcodes were compiled.
3. Load
   - Class loader stores the .class file containg the bytecodes in primary memory.

# 1.9 Java and a Typical Java Develop Environment (Cont.)

4. Verify
   - Bytecode Verifier confirms bytecodes are valid and do not violate Java’s security restrictions.
   - Java enforces strong security to make sure that Java programs arriving over the network do not damage your files or your system.

5. Execute
   - JVM translates bytecodes into machine language in order to perform the actions specified by the program.

# Compilers

t A **compiler** is a program that translates a high-level language program to a machine language program that the computer can directly understand and execute.

- Source code/program
  - The original program in a high level language
- Object code
  - The translated version in machine language

t The word *code* refers to a program or part of a program.