

Introduction to Programming and Algorithms Module 1

CS 146

Sam Houston State University
Dr. Tim McGuire

Module Objectives

To understand:

- the necessity of programming,
- differences between hardware and software,
- common computer architecture,
- the role of the operating system,
- the need for programming languages,
- a brief history of Java,
- the difference between applications and applets,

2

Module Objectives

To understand:

- elements common to all programming languages,
- the role of the Java compiler,
- the role of the Java Virtual Machine,
- the compilation and execution process,
- the fundamental concepts of object-oriented programming,
- the steps involved in creating programs, and
- the fundamental structure of UML diagrams.

3

Topics

Module 1 discusses the following main topics:

- Introduction
- Why Program?
- Computer Systems: Hardware and Software
- Programming Languages
- What Is a Program Made Of?
- The Programming Process
- Object-Oriented Programming

4

Java History

- 1991 - Green Team started by Sun Microsystems.
- *7 Handheld controller for multiple entertainment systems.
- There was a need for a programming language that would run on various devices.
- Java (first named Oak) was developed for this purpose.

5

Introduction

- Java enabled web browser (HotJava) demonstrated at 1995 Sun World conference.
- Java incorporated into Netscape shortly after.
- Java is “cross platform”, meaning that it can run on various computer operating systems.

6

Java Applications and Applets

- Java programs can be of two types:
 - Applications
 - Stand-alone programs that run without the aid of a web browser.
 - Relaxed security model since the user runs the program locally.
 - Applets
 - Small applications that require the use of a Java enabled web browser to run.
 - Enhanced security model since the user merely goes to a web page and the applet runs itself.

7

Why Program?

- Computers are tools that can be programmed to perform many functions, such as:
 - spreadsheets
 - games
 - databases
 - etc.
 - word processing
- Computers are versatile because they can be programmed.
- Computer Programmers implement programs that perform these functions.

8

Why Program?

Aspects of a computer program that must be designed:

- The logical flow of the instructions
- The mathematical procedures
- The layout of the programming statements
- The appearance of the screens
- The way information is presented to the user
- The program's "user friendliness"
- Manuals, help systems, and/or other forms of written documentation.

9

Why Program?

- Programs must be analytically correct as well.
- Programs rarely work the first time they are programmed.
- Programmers must perform the following on a continual basis:
 - analyze,
 - experiment,
 - correct, and
 - redesign.
- Programming languages have strict rules, known as *syntax*, that must be carefully followed.

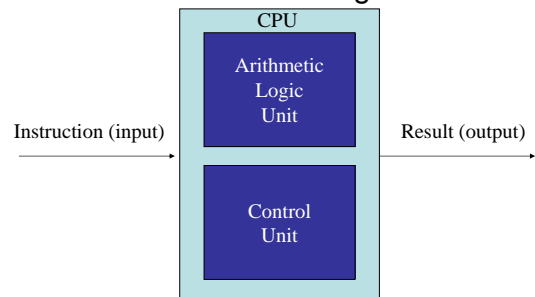
10

Computer Systems: Hardware

- Computer hardware components are the physical pieces of the computer.
- The major hardware components of a computer are:
 - The central processing unit (CPU)
 - Main memory
 - Secondary storage devices
 - Input and Output devices

11

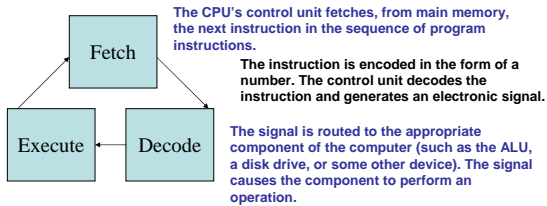
Computer Systems: Hardware Central Processing Unit



12

Computer Systems: Hardware Central Processing Unit

- The CPU performs the fetch, decode, execute cycle in order to process program information.



13

Computer Systems: Hardware Main Memory

- Commonly known as *random-access memory (RAM)*
- RAM contains:
 - currently running programs
 - data used by those programs.
- RAM is divided into units called *bytes*.
- A byte consists of eight *bits* that may be either on or off.

14

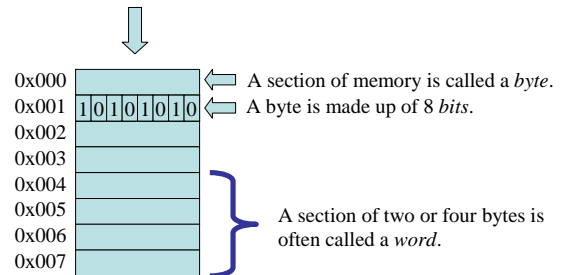
Computer Systems: Hardware Main Memory

- A bit is either on or off:
 - 1 = on
 - 0 = off
- The bits form a pattern that represents a character or a number.
- Each byte in memory is assigned a unique number known as an *address*.
- RAM is *volatile*, which means that when the computer is turned off, the contents of RAM are erased.

15

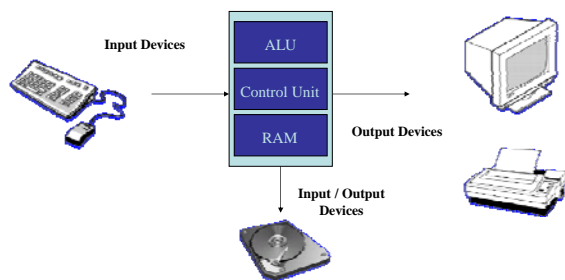
Computer Systems: Hardware Main Memory

Main memory can be visualized as a column or row of cells.



16

Computer Systems: Hardware



17

Computer Systems: Hardware Secondary Storage Devices

- Secondary storage devices are capable of storing information for longer periods of time (*non-volatile*).
- Common Secondary Storage devices:
 - Hard drive
 - Floppy drive
 - CD RW drive
 - CD ROM
 - DVD RAM drive
 - Compact Flash card

18

Computer Systems: Hardware Input Devices

- Input is any data the computer collects from the outside world.
- That data comes from devices known as *input devices*.
- Common input devices:
 - Keyboard
 - Mouse
 - Scanner
 - Digital camera

19

Computer Systems: Hardware Output Devices

- Output is any data the computer sends to the outside world.
- That data is displayed on devices known as *output devices*.
- Common output devices:
 - Monitors
 - Printers
- Some devices such as disk drives perform input and output and are called *I/O devices* (input/output).

20

Computer Systems: Software

- Software refers to the programs that run on a computer.
- There are two classifications of software:
 - Operating Systems
 - Application Software

21

Computer Systems: Software Operating Systems

- An operating system has two functions:
 - Control the system resources.
 - Provide the user with a means of interaction with the computer.
- Operating systems can be either single tasking or multi-tasking.

22

Computer Systems: Software Operating Systems

- A single tasking operating system is capable of running only one program at a time.
 - DOS
- A multitasking operating system is capable of running multiple programs at once.
 - Windows
 - Unix
 - Apple

23

Computer Systems: Software Operating Systems

- Operating systems can also be categorized as single user or multi-user.
 - A single user operating system allows only one user to operate the computer at a time.
 - Multi-user systems allow several users to run programs and operate the computer at once.

24

Computer Systems: Software Single User Systems

Examples:

- DOS
- Windows
- 95/98/ME

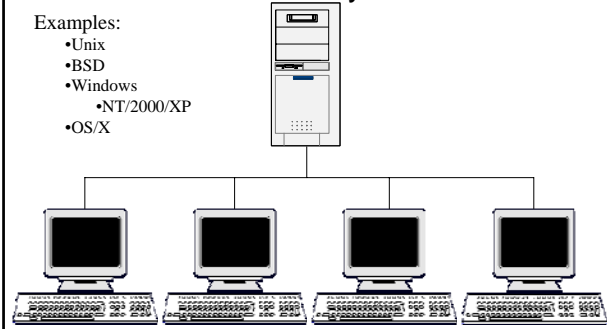


25

Computer Systems: Software Multi-User Systems

Examples:

- Unix
- BSD
- Windows
- NT/2000/XP
- OS/X



26

Computer Systems: Software Application Software

- *Application software* refers to programs that make the computer useful to the user.
- Application software provides a more specialized type of environment for the user to work in.
- Common application software:
 - Spreadsheets
 - Word processors
 - Accounting software
 - Tax software
 - Games

27

Programming Languages

- A program is a set of instructions a computer follows in order to perform a task.
- A programming language is a special language used to write computer programs.
- A computer program is a set of instructions that enable the computer to solve a problem or perform a task.
- Collectively, these instructions form an *algorithm*

28

Programming Languages

- An algorithm is a set of well defined steps to completing a task.
- The steps in an algorithm are performed sequentially.
- A computer needs the algorithm to be written in *machine language*.
- Machine language is written using *binary numbers*.
- The binary numbering system (base 2) only has two digits (0 and 1).

29

Programming Languages

- The binary numbers are encoded as a machine language .
- Each CPU has its own machine language.
 - Motorola 68000 series processors
 - Intel x86 series processors
 - DEC Alpha processors, etc.
- Example of a machine language instruction:

101101000000101

30

Programming Languages

- In the distant past, programmers wrote programs in machine language.
- Programmers developed higher level programming languages to make things easier.
- The first of these was *assembler*.
- Assembler made things easier but was also processor dependent.

31

Programming Languages

- High level programming languages followed that were not processor dependent.
- Common programming languages:
 - BASIC
 - COBOL
 - Pascal
 - C
 - C++
 - Java

32

Programming Languages Common Language Elements

- There are some concepts that are common to virtually all programming languages.
- Common concepts:
 - Key words
 - Operators
 - Punctuation
 - Programmer-defined identifiers
 - Strict syntactic rules.

33

Programming Languages Sample Program

```
public class HelloWorld
{
    public static void main(String[] args)
    {
        String message = "Hello World";
        System.out.println(message);
    }
}
```

34

Programming Languages Sample Program

- Key words in the sample program are:
 - public
 - void
 - class
 - String
 - static

String is not really a key word but is the name of a predefined class in Java.
- Key words are lower case (Java is a case sensitive language).
- Key words cannot be used as a programmer-defined identifier.

35

Programming Languages

- Some Java key words have no meaning but are reserved to prevent their use. (ex. goto, const, include)
- Semi-colons are used to end Java statements; however, not all lines of a Java program end a statement.
- Part of learning Java is to learn where to properly use the punctuation.

36

Programming Languages Lines vs Statements

- There is a difference between lines and statements when discussing source code.

```
System.out.println(  
    message);
```

- This is one Java statement written using two lines. Do you see the difference?
- A statement is a complete Java instruction that causes the computer to perform an action.

37

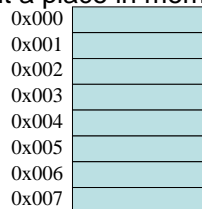
Programming Languages Variables

- Information in a Java program is stored in memory.
- Variable names represent a location in memory.
- Variables in Java are sometimes called fields.
- Variables are created by the programmer who assigns it a programmer-defined identifier.
ex: `int hours = 40;`
- In this example, the variable *hours* is created as an integer (more on this later) and assigned the value of 40.

38

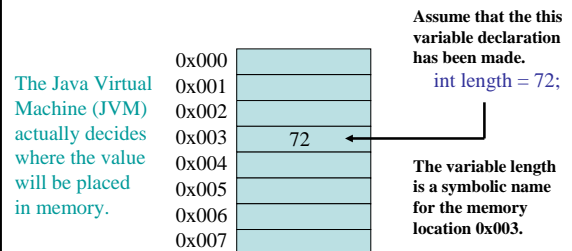
Programming Languages Variables

- Variables are simply a name given to represent a place in memory.



39

Programming Languages Variables



40

The Compiler and the Java Virtual Machine

- A programmer writes Java programming statements for a program.
- These statements are known as *source code*.
- A *text editor* is used to edit and save a Java *source code file*.
- Source code files have a *.java* file extension.
- A *compiler* is a program that translates source code into an executable form.

41

The Compiler and the Java Virtual Machine

- A compiler is run using a source code file as input.
- Syntax errors that may be in the program will be discovered during compilation.
- *Syntax errors* are mistakes that the programmer has made that violate the rules of the programming language.
- The compiler creates another file that holds the translated instructions.

42

The Compiler and the Java Virtual Machine

- Most compilers translate source code into *executable* files containing *machine code*.
- The Java compiler translates a Java source file into a file that contains *byte code* instructions.
- Byte code instructions are the machine language of the *Java Virtual Machine (JVM)* and cannot be directly executed directly by the CPU.

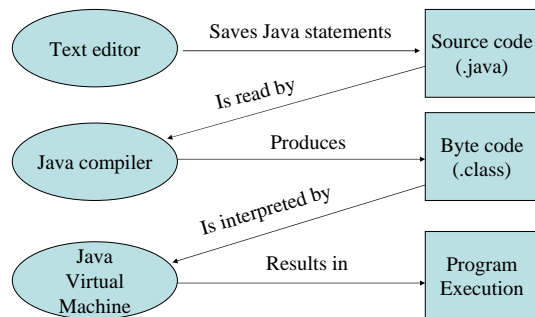
43

The Compiler and the Java Virtual Machine

- Byte code files end with the *.class* file extension.
- The JVM is a program that *emulates* a micro-processor.
- The JVM executes instructions as they are read.
- JVM is often called an *interpreter*.
- Java is often referred to as an *interpreted language*.

44

Program Development Process



45

Portability

- *Portable* means that a program may be written on one type of computer and then run on a wide variety of computers, with little or no modification.
- Java byte code runs on the JVM and not on any particular CPU; therefore, compiled Java programs are highly portable.
- JVMs exist on many platforms:
 - Windows
 - Unix
 - Macintosh
 - BSD
 - Linux
 - Etc.

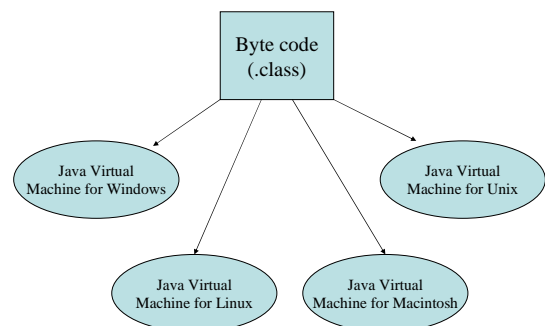
46

Portability

- With most programming languages, portability is achieved by compiling a program for each CPU it will run on.
- Java provides an JVM for each platform so that programmers do not have to recompile for different platforms.

47

Portability



48

Java Versions

- Java began at version 1.0 and is now at version 5.0 (Sun skipped from 1.4 to 5.0).
- With the advent of version 1.2, Java became Java2 because it provided much more functionality.
- Java2 version 5.0 can still compile Java 1.0 programs as long as no features of any other version of Java are present.
- This is called backwards compatibility.

49

Java Versions

- Java began as the Java Development Kit (JDK).
- With the advent of Java2, through version 1.4 it changed to the Java Software Development Kit (SDK)
- In Java 5, JDK is back
- There are different editions of Java:
 - J2SE - Java2 *Standard Edition*.
 - J2EE - Java2 *Enterprise Edition*.
 - J2ME - Java2 *Micro Edition*.

50

Compiling a Java Program

- The Java compiler is a *command line* utility.
- The command to compile a program is:
`javac filename.java`
- javac is the Java compiler.
- The .java file extension must be used.

Example: To compile a java source code file named Payroll.java you would use the command:
`javac Payroll.java`

51

The Programming Process

1. Clearly define what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools to create a model of the program.
4. Check the model for logical errors.

52

The Programming Process

5. Enter the code and compile it.
6. Correct any errors found during compilation.
Repeat Steps 5 and 6 as many times as necessary.
7. Run the program with test data for input.
8. Correct any runtime errors found while running the program.
Repeat Steps 5 through 8 as many times as necessary.
9. Validate the results of the program.

53

Software Engineering

- Encompasses the whole process of crafting computer software.
- Software engineers perform several tasks in the development of complex software projects.
 - designing,
 - writing,
 - testing,
 - debugging,
 - documenting,
 - modifying, and
 - maintaining.

54

Software Engineering

- Software engineers develop:
 - program specifications,
 - diagrams of screen output,
 - diagrams representing the program components and the flow of data,
 - pseudocode,
 - examples of expected input and desired output.

55

Software Engineering

- Software engineers also use special software designed for testing programs.
- Most commercial software applications are large and complex.
- Usually a team of programmers, not a single individual, develops them.
- Program requirements are thoroughly analyzed and divided into subtasks that are handled by
 - individual teams
 - individuals within a team.

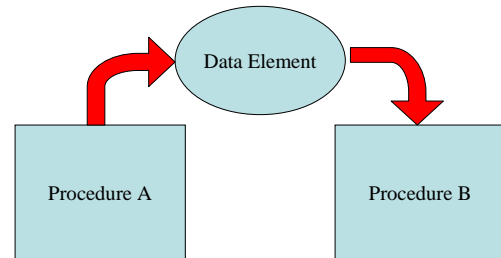
56

Procedural Programming

- Older programming languages were procedural.
- A *procedure* is a set of programming language statements that, together, perform a specific task.
- Procedures typically operate on data items that are separate from the procedures.
- In a procedural program, the data items are commonly passed from one procedure to another.

57

Procedural Programming



58

Procedural Programming

- In procedural programming, procedures are developed to operate on the program's data.
- Data in the program tends to be global to the entire program.
- Data formats might change and thus, the procedures that operate on that data must change.

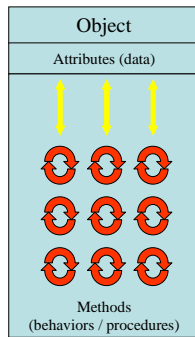
59

Object-Oriented Programming

- Object-oriented programming is centered on creating objects rather than procedures.
- Objects are a melding of data and procedures that manipulate that data.
- Data in an object are known as *attributes*.
- Procedures in an object are known as *methods*.

60

Object-Oriented Programming



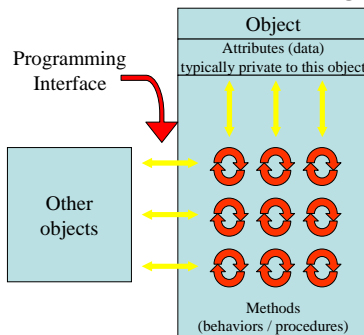
61

Object-Oriented Programming

- Object-oriented programming combines data and behavior via *encapsulation*.
- *Data hiding* is the ability of an object to hide data from other objects in the program.
- Only an objects methods should be able to directly manipulate its attributes.
- Other objects are allowed manipulate an object's attributes via the object's methods.
- This indirect access is known as a *programming interface*.

62

Object-Oriented Programming



63

Object-Oriented Programming Data Hiding

- Data hiding is important for several reasons.
 - It protects of attributes from accidental corruption by outside objects.
 - It hides the details of how an object works, so the programmer can concentrate on using it.
 - It allows the maintainer of the object to have the ability to modify the internal functioning of the object without “breaking” someone else's code.

64

Object-Oriented Programming Code Reusability

- Object-Oriented Programming (OOP) has encouraged component reusability.
- A component is a software object contains data and methods that represents a specific concept or service.
- Components typically are not stand-alone programs.
- Components can be used by programs that need the component's service.
- Reuse of code promotes the rapid development of larger software projects.

65

Classes and Objects

- Components are objects.
- The programmer determines the attributes and methods needed, and then creates a class.
- A *class* is a collection of programming statements that define the required object
- A class as a “blueprint” that objects may be created from.
- An object is the realization (instantiation) of a class in memory.

66

Classes and Objects

- Classes can be used to instantiate as many objects as are needed.
- Each object that is created from a class is called an *instance* of the class.
- A program is simply a collection of objects that interact with each other to accomplish a goal.

67

Classes and Objects

The *Insect* class defines the attributes and methods that will exist in all objects that are instances of the *Insect* class.

Insect class

housefly object

The housefly object is an instance of the *Insect* class.

The mosquito object is an instance of the *Insect* class.

mosquito object

68

Inheritance

- *Inheritance* is the ability of one class to extend the capabilities of another.
- Consider the class *Car*.
- A *Car* is a specialized form of the *Vehicle* class.
- So, it is said that the *Vehicle* class is the base or parent class of the *Car* class.
- The *Car* class is the derived or child class of the *Vehicle* class.

69

Inheritance

Vehicle represents all of the generic attributes and methods of a vehicle.

Vehicle

Vehicle is the parent class.

is-a relationship

Car and Truck are child classes of Vehicle.

Car

Truck

Car and Truck are Specialized versions of a Vehicle.

70