1. Introduction

[ITP20003] Java Programming

Agenda

- Computer Basics
- The First Java Application
- Programming Basics
- Graphics Supplement

Computer

- Composed of …
  - Input devices (keyboards, mouse, camera, mic,…)
  - Output devices (monitor, printer, speaker, …)
  - Storages (HDD, SSD, flash memory, CD/DVD, …)
  - CPU, main memory, controller, …

CPU and Memory

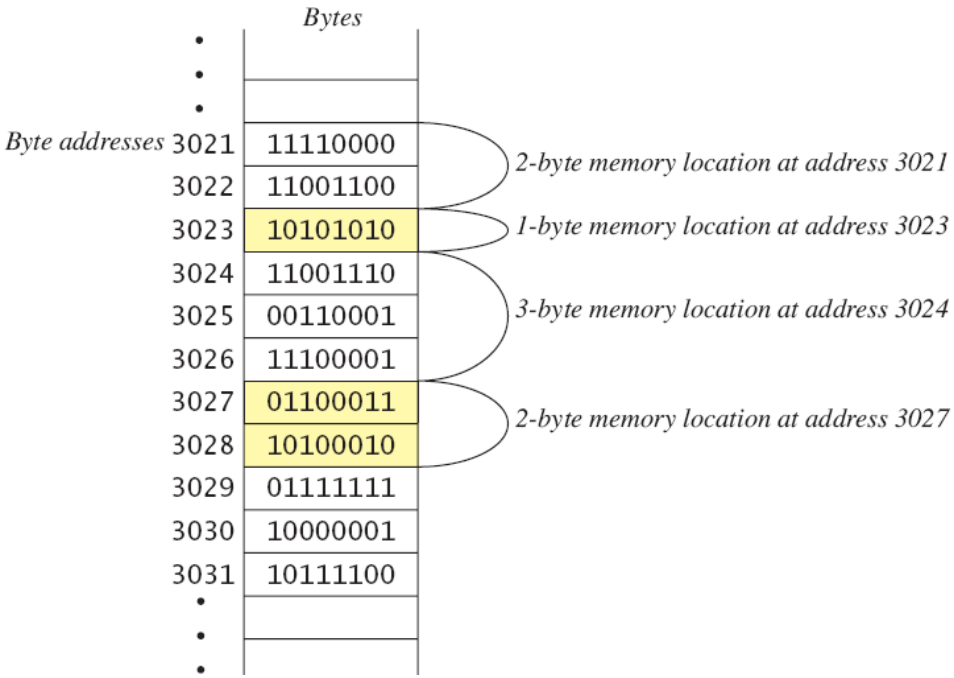
- CPU - carries out only very simple instructions
  - Moving data from one place in memory to another
  - Performing some basic arithmetic (+, -, …)

  Cf. program: a sequence of instructions to accomplish a task

- Main memory (RAM) – stores data and instructions
  - Volatile
  - Fast
  - Smaller and more expensive than auxiliary memory
  - The only storage CPU can access directly.

Main Memory

- Main memory consists of a long list of numbered **bytes**.
  - All kinds of data are stored as a series of bits or bytes.
- The location of a byte is called its **address**.
  - The address of other memory unit, i.e. WORD(2bytes) or DWORD(4bytes), is the address of the starting byte.



ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	₩	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

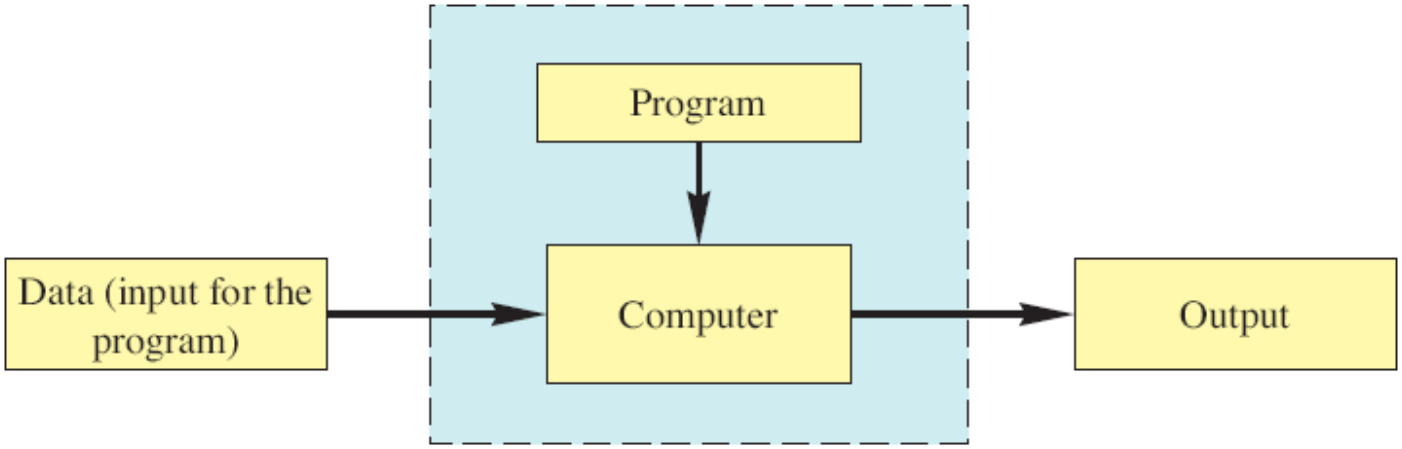
- Each character is represented by 7 bits
  - 0x00~0x1f: control characters
  - 0x20~0x7f: printable characters

    Ex) ‘a’ = 0x61, ‘0’ = 0x30, ‘-’ = 0x2D

Programs

- Program: a sequence of instructions for a computer to follow.

- Execution of program
  - Program is executed by computer (+ OS)
  - Program takes input and produces output



Programming Languages

- Primitive programming languages
  - Machine language - a sequence of machine instructions
    - Machine instruction: primitive instructions CPU can run.
  - Assembly language – a sequence of assembly instruction
    - Assembly instruction: symbolic representation of machine instruction
    - Needs translation into machine language (assembler)

- High-level programming languages
  - Human-friendly language to describe the things the computer should do.
  - Only for human (cannot be executed on computer)

    ➔ Needs translation into machine language code. (interpreter/compiler)

Interpreter and Compiler

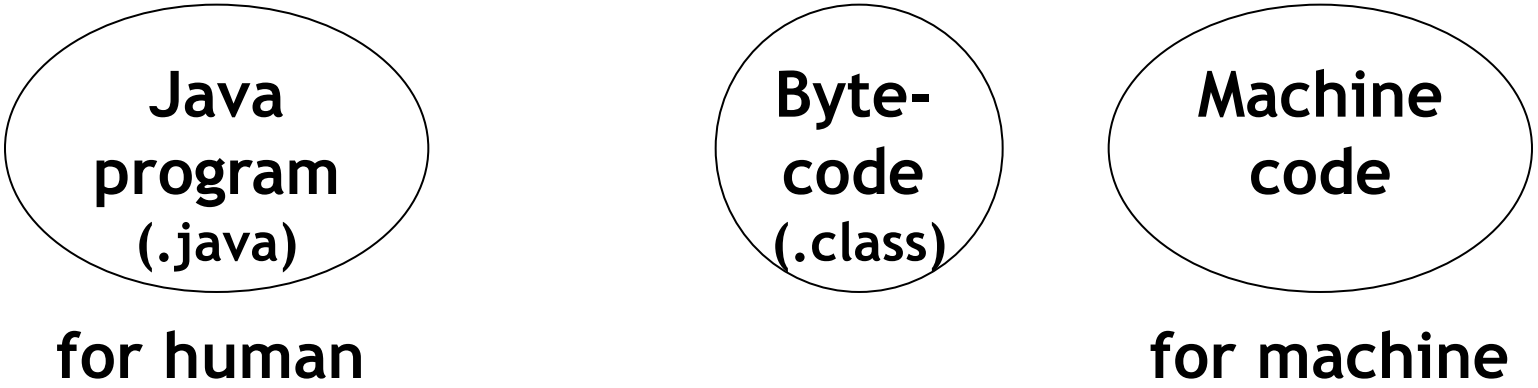
- Interpreter - translates and executes each command alternatively
  - Translates every time the program runs.
  - Interactive

- Compiler - translates the whole (or a part of) program into **machine code** (exceptions: Java, C#, …)
  - Compile once execute often.
  - Fast

Java Bytecode

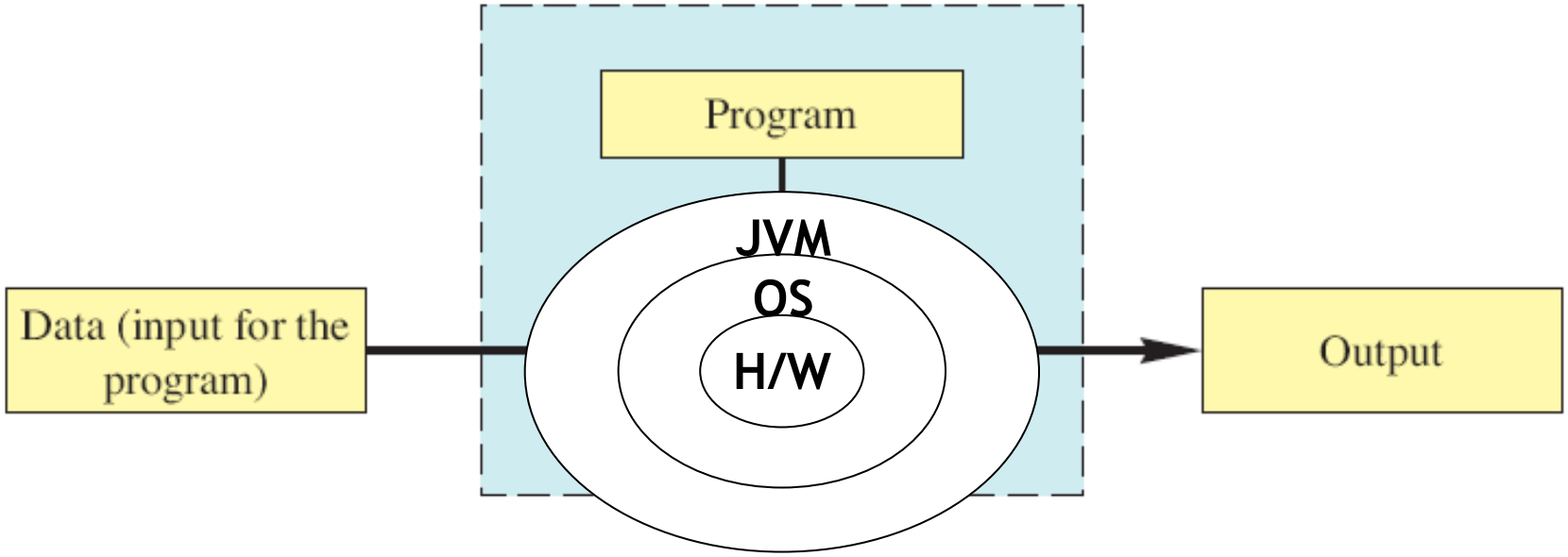
- Java compiler translates Java program into bytecode rather than machine language.

- Bytecode: machine language of a hypothetical computer known as a virtual machine, called JVM.
  - Intermediate form between Java program and machine code.
  - Easy to interpret

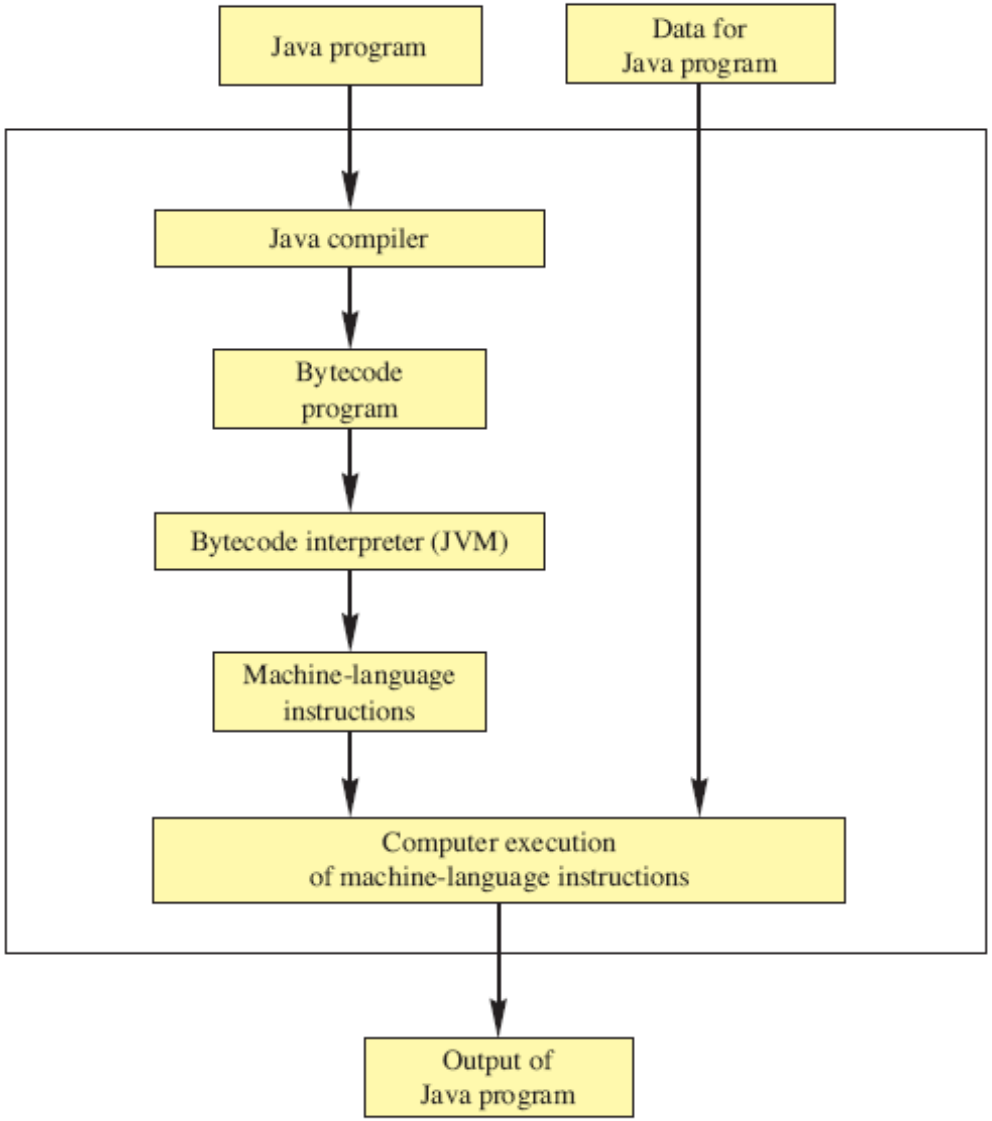


Java Virtual Machine (JVM)

- JVM interprets bytecode (translation + execution)
- JVM provides platform-independent environment.
  - There exists JVMs for various H/W's and OS's
  - Java bytecode can run on any JVM.



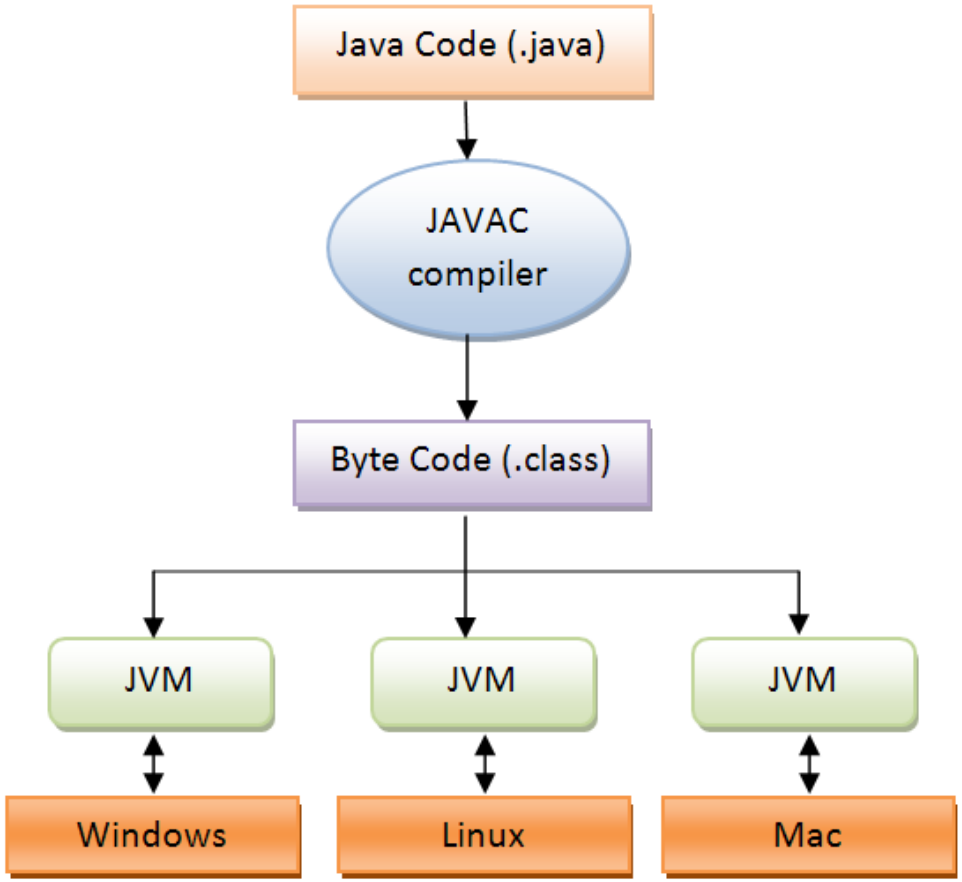
Compiling and Running Java



Java Virtual Machine (JVM)

- JVM provides great portability.

  "Compile once, run everywhere!"



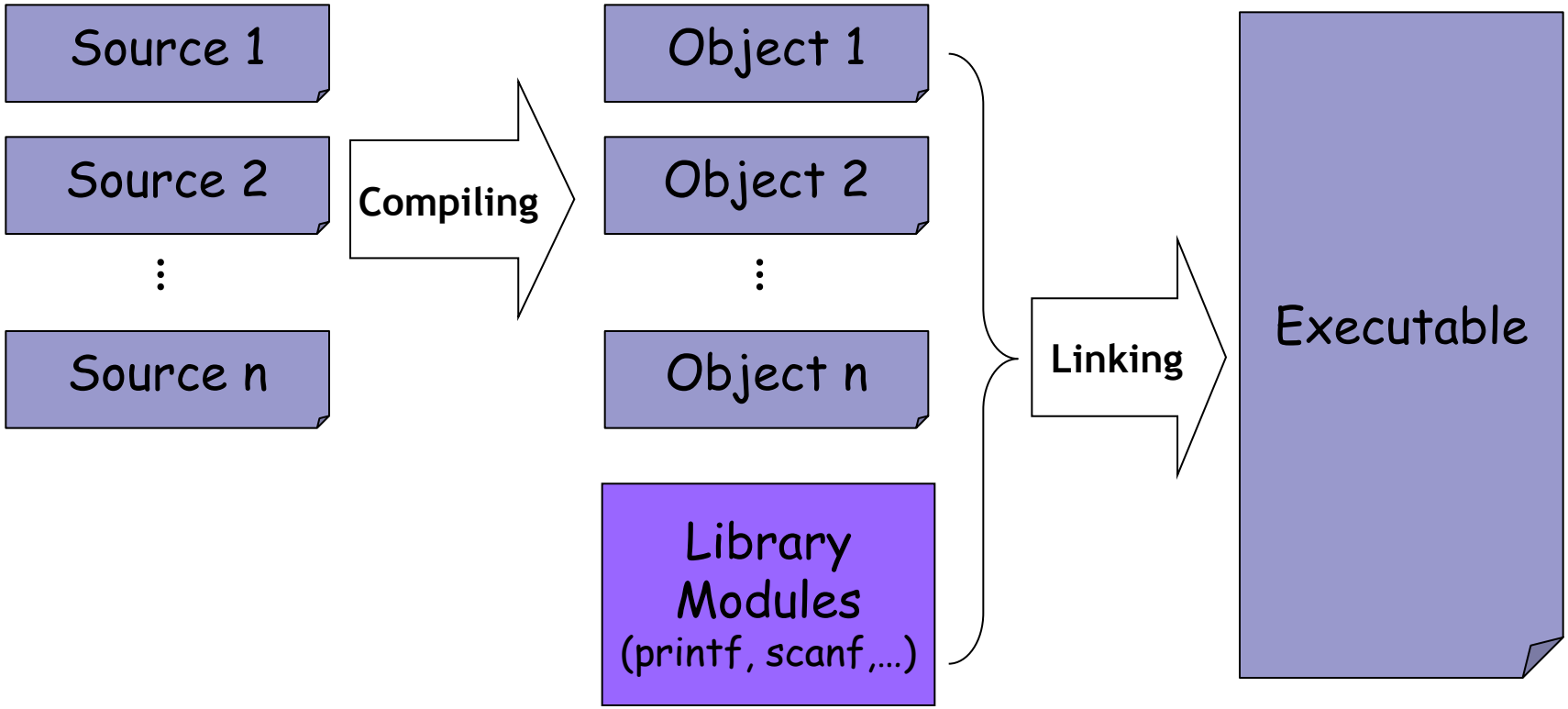
Class Loader

- A Java program is seldom written as one piece of code all in one file.
- Instead, it typically consists of different pieces, known as classes.
- Class Loader connects the classes to run the program.
  - This connecting is typically done automatically.
  - Class loader corresponds to the linker of other programming language.

Creating and Running C Programs

- Link
  - Integrating objects and library modules required to execute.

Notice! A program can be distributed in multiple source files.



Agenda

- Computer Basics
- **The First Java Application**
- Programming Basics
- Graphics Supplement

Applications and Applets

- Application: regular program.
  - Run on your computer
    - H/W + OS + VM
- Applet
  - Sent to another location on the Internet and run there.
    - Web browser + VM

The First Java Application Program

```
import java.util.Scanner;

public class FirstProgram
{
    public static void main(String[] args)
    {
        System.out.println("Hello out there.");
        System.out.println("I will add two numbers for you.");
        System.out.println("Enter two whole numbers on a line:");

        int n1, n2;

        Scanner keyboard = new Scanner(System.in);
        n1 = keyboard.nextInt();
        n2 = keyboard.nextInt();

        System.out.println("The sum of those two numbers is");
        System.out.println(n1 + n2);
    }
}
```

Gets the `Scanner` *class from the package (library)* `java.util`

Name of the class—your choice

Sends output to screen

Says that `n1` *and* `n2` *are variables that hold integers (whole numbers)*

Readies the program for keyboard input

Reads one whole number from the keyboard

The First Java Application Program

- Result

```
Hello out there.
I will add two numbers for you.
Enter two whole numbers on a line:
12 30
The sum of those two numbers is
42
```

Here, the text typed in by the user is shown in blue

The First Java Application Program

- import java.util.Scanner;
  - Tells the compiler that this program uses the class Scanner.

- class FirstProgram // see OOP

```
public class FirstProgram
{
. . .
}
```

- The main method

```
public static void main(String[] args)
{
. . .
}
```

The First Java Application Program

- System.out.println()
  - Displays what is shown in parentheses
  - System.out is an object used to send output to the screen
  - println is the method that performs this action for the object System.out.

- int n1, n2; // variable declaration
  - variable: a memory space with a name to store a piece of data.
  - int: data type (integer)
  - n1, n2: variable names

The First Java Application Program

- Scanner keyboard = new Scanner(System.in);
  - Prepares to read from the keyboard
  - System.in is an object used to read input to the keyboard

- n1 = keyboard.nextInt();
- n2 = keyboard.nextInt();
  - Reads integer numbers from the keyboard

Writing a Java Program

- A Java program is composed of smaller parts, called classes
  - List 1.1 uses three classes: FirstProgram, System, Scanner
  - Each class should be in a separate file with the same filename.

    Ex) FirstProgram.java

- Writing a Java program = writing classes
  - Design the whole program
  - Decompose it into classes
  - Implement each class

Compile and Running a Java Program

- Compile and Running with JDK (Java Development Toolkit)
  - Compiler + JRE (incl. JVM)

  cf: JRE: Java Runtime Environment (JVM + built-in classes + α)

  - Compile: javac FirstProgram.java
  - Run: java FirstProgram

  ➔ JDK should be installed, and its *bin* directory should be in PATH.

- IDE (Integrated Development Environment)
  - Editor + compiler + runtime + debugger + …

  Ex) Eclipse, NetBeans, …

  - Background compile
  - Run
    - Menu->Run->Run As->Java Application
    - Menu->Run->Run
    - CTRL-F11

Agenda

- Computer Basics
- The First Java Application
- **Programming Basics**
- Graphics Supplement

Object-Oriented Programming

- Java is an object-oriented programming language, abbreviated OOP.
  - OOP is a technique that experienced programmers have found to be extremely helpful.

- The world is made up of objects.

  Ex) people, automobiles, buildings, …

- Object-oriented programming (OOP) treats a program as **a collection of objects** that **interact by means of actions**.

Object-Oriented Programming

- Objects, appropriately, are called **objects**.
- Actions are called **methods**.
- Objects of the same kind have the same type and belong to the same **class**.
  - Objects within a class have **a common set of methods** and **the same kinds of data**
  - But each object can have **it's own data values**.

Class, Object, and Methods

- **Class**: a type of entities

  Ex) Sonata, Genesis, Galaxy Note, i-Pad…

- **Object**: a specific entity

  Ex) my Sonata (with a specific VIN and plate number)

- **Method**: an action that an object can perform

  Ex) Sonata has *go*, *stop*, *left_turn*, *right_turn*, …

- **Attribute**: component that constructs an object
  - Also called fields, member variable, data member, …

  Ex) body, engine, wheel, tire, chair, door, trunk, …

OOP Design Principles

- OOP adheres to three primary design principles:
  - Encapsulation
  - Polymorphism
  - Inheritance

Encapsulation

- The data and methods associated with any particular class are encapsulated ("put together in a capsule"), but only part of the contents is made accessible.
  - Encapsulation provides a means of using the class, but it omits the details of how the class works.

    Ex) accelerator pedal, brake pedal, steering wheel, …
  - Encapsulation often is called information hiding.

    Ex) fuel injectors, automatic braking control system, power steering pump, …

Polymorphism

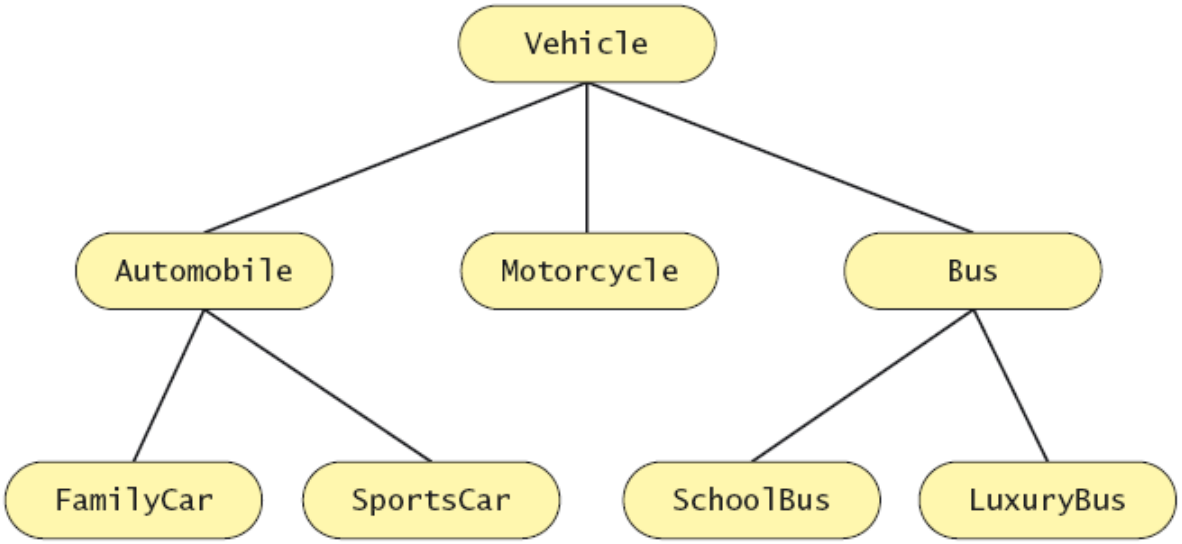
- From the Greek meaning “many forms”

- The same program instruction adapts to mean different things in different contexts.
  - A method name produces results that depend on the class of the object that used the method.

  Ex) ‘go’ method of an automobile vs. ‘go’ method of an airplane.

Inheritance

- Classes can be organized using inheritance.
  - 'is a' relation
- A class at lower levels inherits all the characteristics of classes above it in the hierarchy.
  - Inherited characteristics do not need to be repeated.
  - New characteristics are added.



Inheritance in Java

- Used to organize classes
- New characteristics are added.

Algorithms

- An **algorithm** describes a means of performing an action.
  - Algorithm = a series of actions
    cf. program = a series of instructions (or commands)
  - An abstracted form of program.
  - For human, not machine

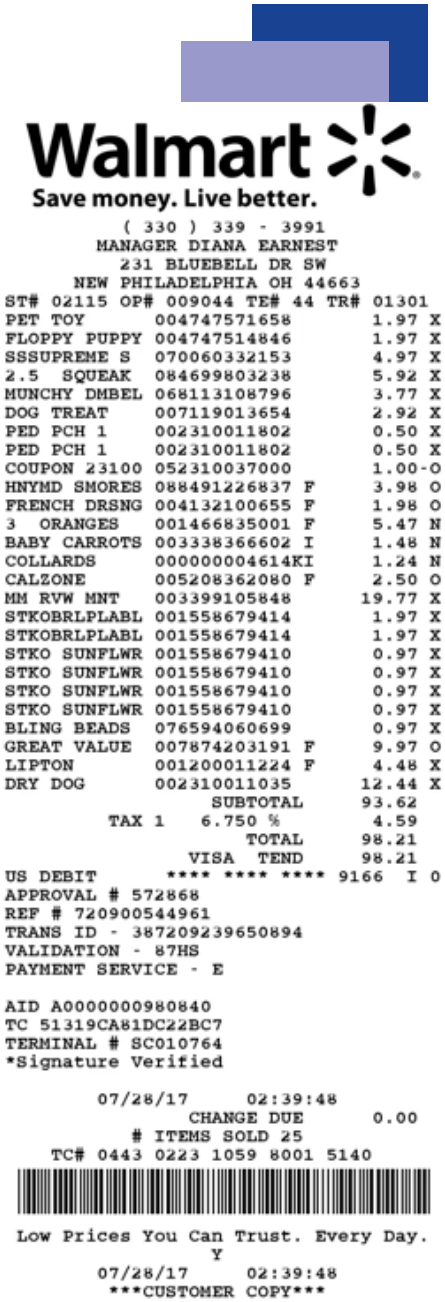
- Once an algorithm is defined, expressing it in Java (or in another programming language) usually is easy.

- An algorithm must be expressed completely and precisely.

- Algorithms usually are expressed in pseudocode.

Example: Total Cost of All Items

- Write the number 0 on the whiteboard.
- For each item on the list
  - Add the cost of the item to the number on the whiteboard
  - Replace the number on the whiteboard with the result of this addition.
- Announce that the answer is the number written on the whiteboard.



Walmart
Save money. Live better.

```
          ( 330 ) 339 - 3991
        MANAGER DIANA EARNEST
          231 BLUEBELL DR SW
      NEW PHILADELPHIA OH 44663
ST# 02115 OP# 009044 TE# 44 TR# 01301
PET TOY        004747571658        1.97 X
FLOPPY PUPPY   004747514846        1.97 X
SSSUPREME S    070060332153        4.97 X
2.5  SQUEAK    084699803238        5.92 X
MUNCHY DMBEL   068113108796        3.77 X
DOG TREAT      007119013654        2.92 X
PED PCH 1      002310011802        0.50 X
PED PCH 1      002310011802        0.50 X
COUPON 23100   052310037000        1.00-O
HNYMD SMORES   088491226837 F      3.98 O
FRENCH DRSNG   004132100655 F      1.98 O
3  ORANGES     001466835001 F      5.47 N
BABY CARROTS   003338366602 I      1.48 N
COLLARDS       000000004614KI      1.24 N
CALZONE        005208362080 F      2.50 O
MM RVW MNT     003399105848       19.77 X
STKOBRLPLABL   001558679414        1.97 X
STKOBRLPLABL   001558679414        1.97 X
STKO SUNFLWR   001558679410        0.97 X
STKO SUNFLWR   001558679410        0.97 X
STKO SUNFLWR   001558679410        0.97 X
STKO SUNFLWR   001558679410        0.97 X
BLING BEADS    076594060699        0.97 X
GREAT VALUE    007874203191 F      9.97 O
LIPTON         001200011224 F      4.48 X
DRY DOG        002310011035       12.44 X
                    SUBTOTAL      93.62
          TAX 1   6.750 %          4.59
                      TOTAL      98.21
                  VISA  TEND      98.21
US DEBIT       **** **** **** 9166  I 0
APPROVAL # 572868
REF # 720900544961
TRANS ID - 387209239650894
VALIDATION - 87HS
PAYMENT SERVICE - E

AID A0000000980840
TC 51319CA81DC22BC7
TERMINAL # SC010764
*Signature Verified

        07/28/17      02:39:48
              CHANGE DUE        0.00
          # ITEMS SOLD 25
    TC# 0443 0223 1059 8001 5140
```

Low Prices You Can Trust. Every Day.

```
                Y
        07/28/17      02:39:48
         ***CUSTOMER COPY***
```

Reusable Components

- Most programs are created by combining existing components.
  - Programs are NOT usually created entirely from scratch.
- Reusing components saves time and money.
- Reused components are likely to be better developed, and more reliable.
- New components should be designed to be reusable by other applications.
- Java provides many classes

  https://docs.oracle.com/en/java/javase/21/docs/api/index.html

Java Platform API

https://docs.oracle.com/en/java/javase/21/docs/api/index.html

Java® Platform, Standard Edition & Java Development Kit Version 21 API Specification

This document is divided into two sections:

Java SE

The Java Platform, Standard Edition (Java SE) APIs define the core Java platform for general-purpose computing. These APIs are in modules whose names start with `java`.

JDK

The Java Development Kit (JDK) APIs are specific to the JDK and will not necessarily be available in all implementations of the Java SE Platform. These APIs are in modules whose names start with `jdk`.

All Modules | Java SE | JDK | Other Modules

Module	Description
java.base	Defines the foundational APIs of the Java SE Platform.
java.compiler	Defines the Language Model, Annotation Processing, and Java Compiler APIs.
java.datatransfer	Defines the API for transferring data between and within applications.
java.desktop	Defines the AWT and Swing user interface toolkits, plus APIs for accessibility, audio, imaging, printing, and JavaBeans.
java.instrument	Defines services that allow agents to instrument programs running on the JVM.
java.logging	Defines the Java Logging API.
java.management	Defines the Java Management Extensions (JMX) API.
java.management.rmi	Defines the RMI connector for the Java Management Extensions (JMX) Remote API.
java.naming	Defines the Java Naming and Directory Interface (JNDI) API.
java.net.http	Defines the HTTP Client and WebSocket APIs.
java.prefs	Defines the Preferences API.
java.rmi	Defines the Remote Method Invocation (RMI) API.
java.scripting	Defines the Scripting API.
java.se	Defines the API of the Java SE Platform.
java.security.jgss	Defines the Java binding of the IETF Generic Security Services API (GSS-API).

questions or comments?

hchoi@handong.edu