1. Introduction

[ECE20016/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

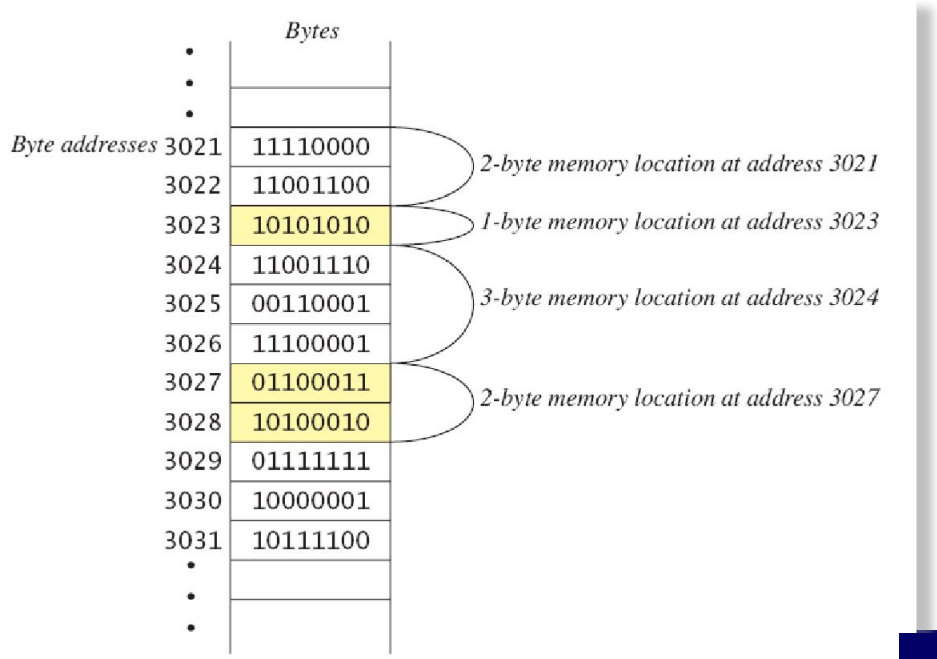
- CPU - carries out only very simple instructions
  - Moving data from one place in memory to another
  - Performing some basic arithmetic (+, -, …)

  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 that CPU can access directly.

Main Memory

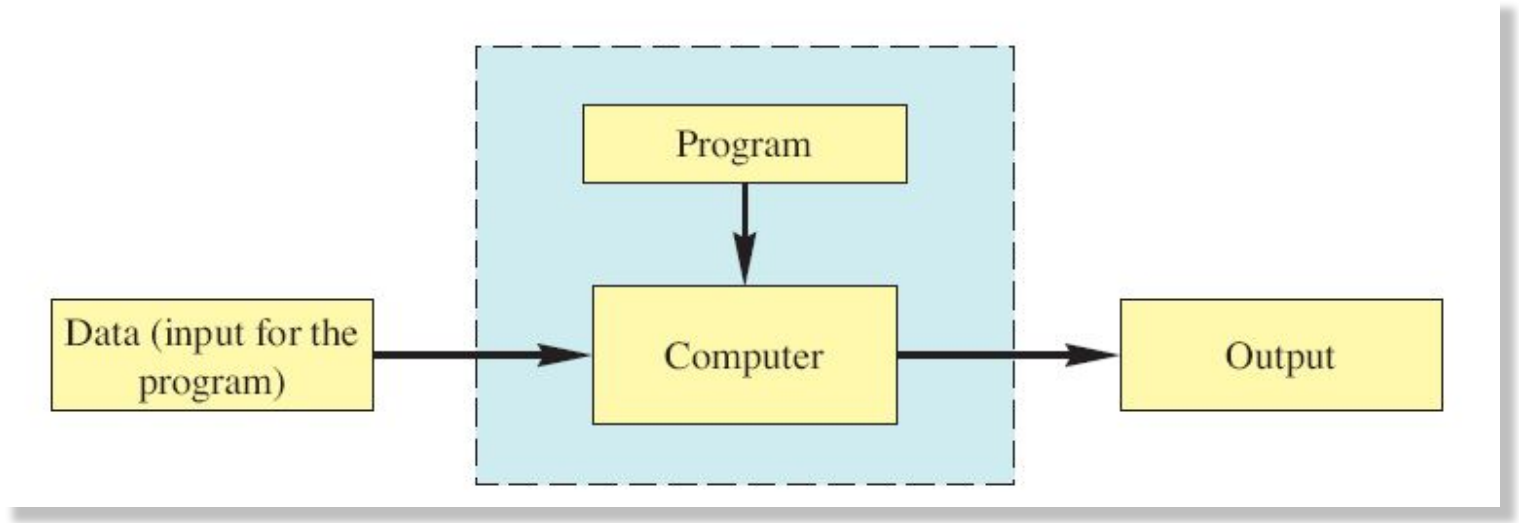
- Main memory consists of a long list of numbered **bytes**.
  - All kinds of data are stored as a series of bits or bytes.
- The location of a byte is called its **address**.
  - The address of other memory unit, i.e. WORD(2bytes) or DWORD(4bytes), is the address of the starting byte.



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

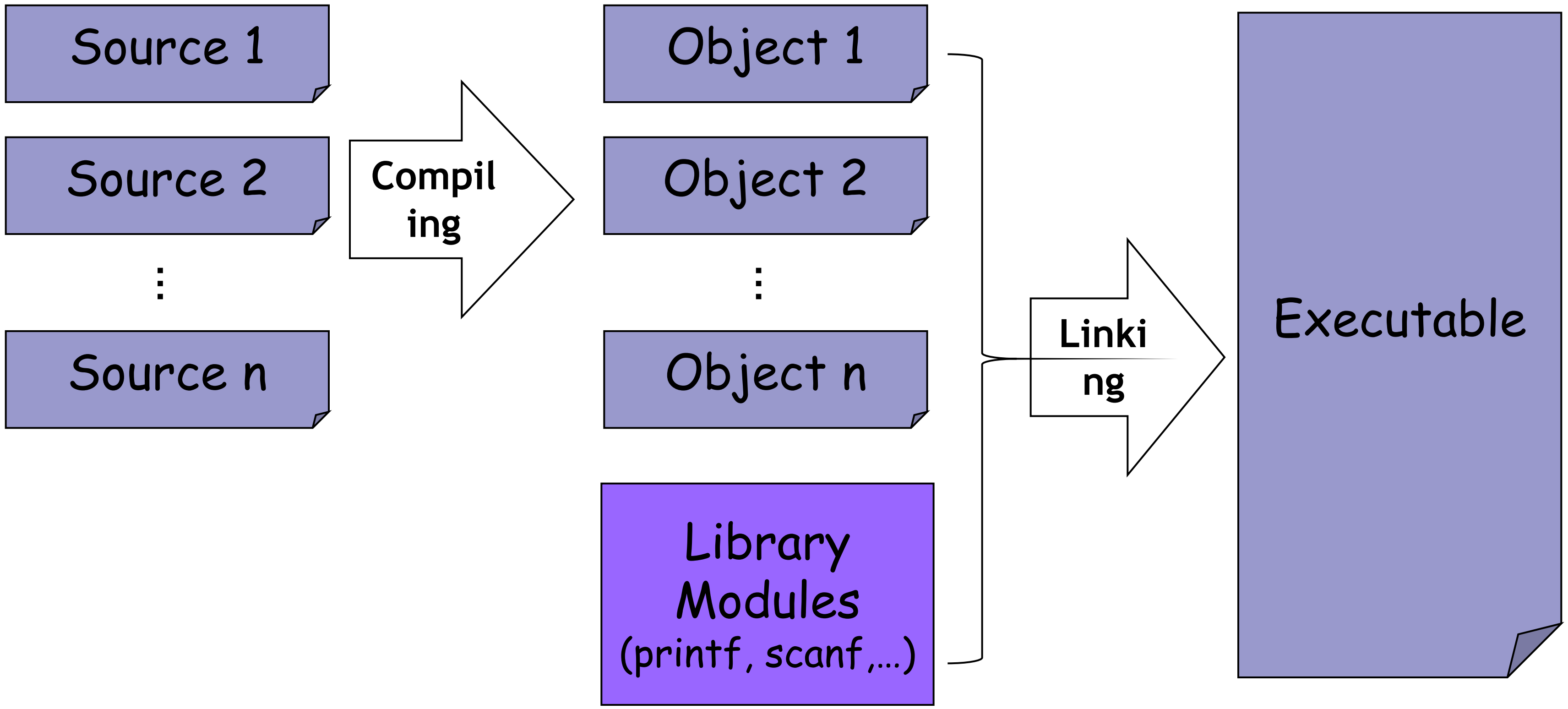
- Interpreter - translates and executes each command alternatively
  - Translates every time the program runs.
  - Interactive

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

Creating and Running C Programs

- Link
  - Integrating objects and library modules required to execute

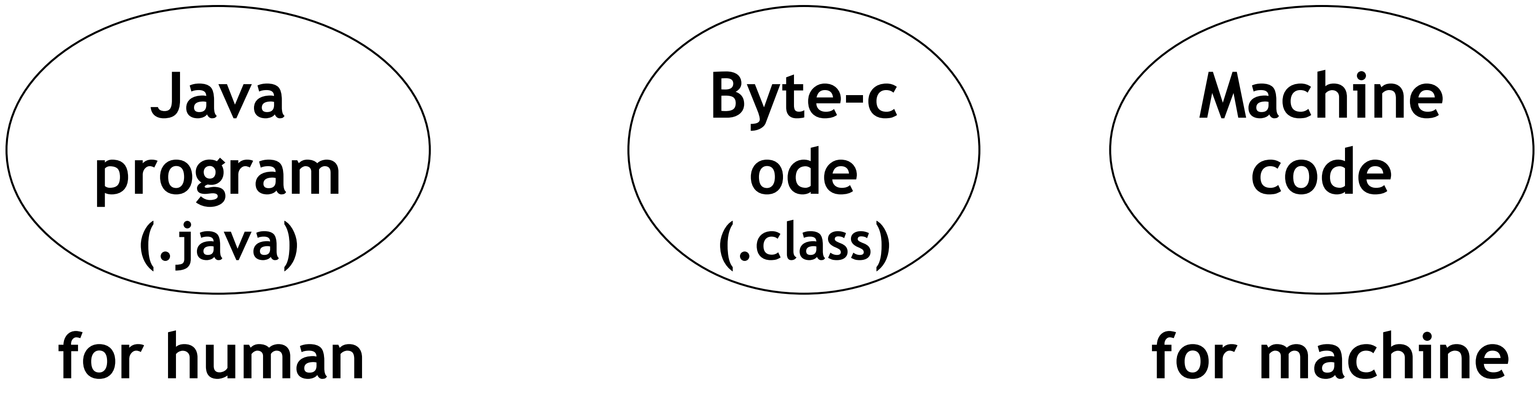
Notice! a program can be distributed in multiple source files.



Java Bytecode

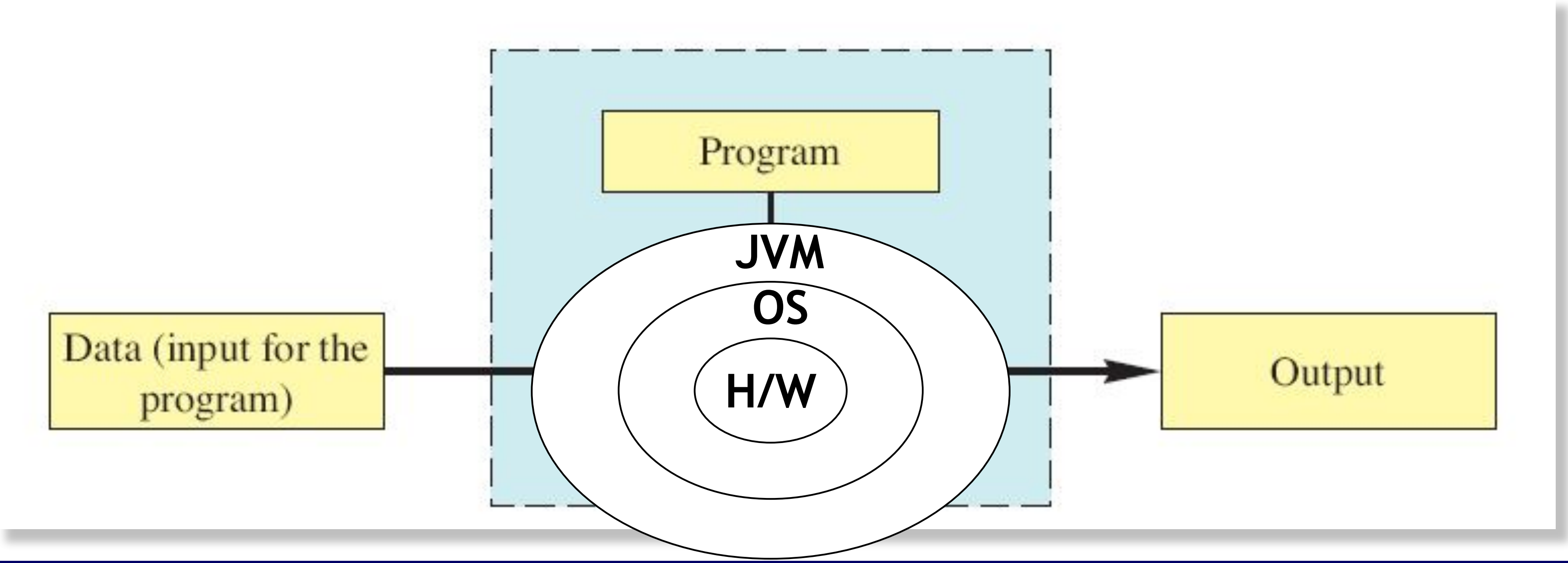
- Java compiler translates Java program into bytecode rather than machine language.

- 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

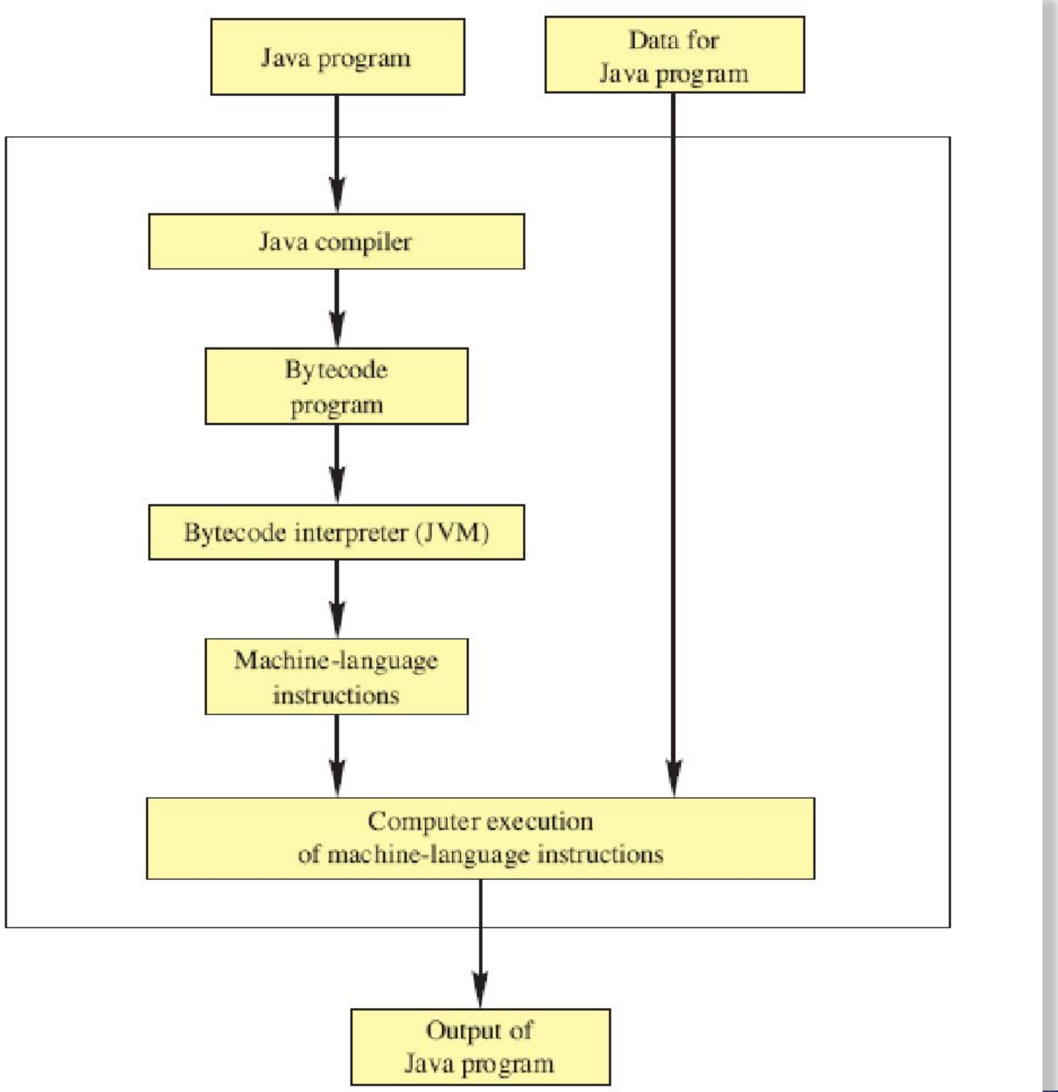


Java Virtual Machine (JVM)

- 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.



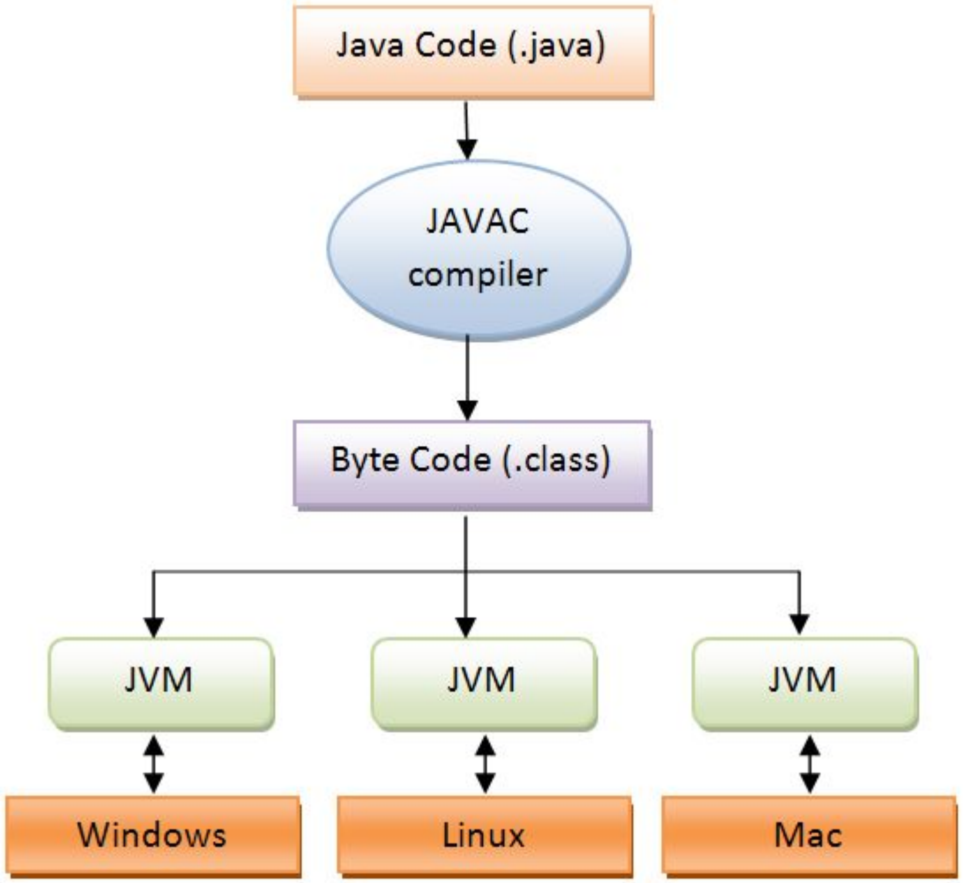
Compiling and Running Java


Java program
Data for Java program
Java compiler
Bytecode program
Bytecode interpreter (JVM)
Machine-language instructions
Computer execution of machine-language instructions
Output of Java program

Java Virtual Machine (JVM)

- JVM provides great portability.

  "Compile once, run everywhere!"



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.
    - H/W + OS + VM + Web browser

Agenda

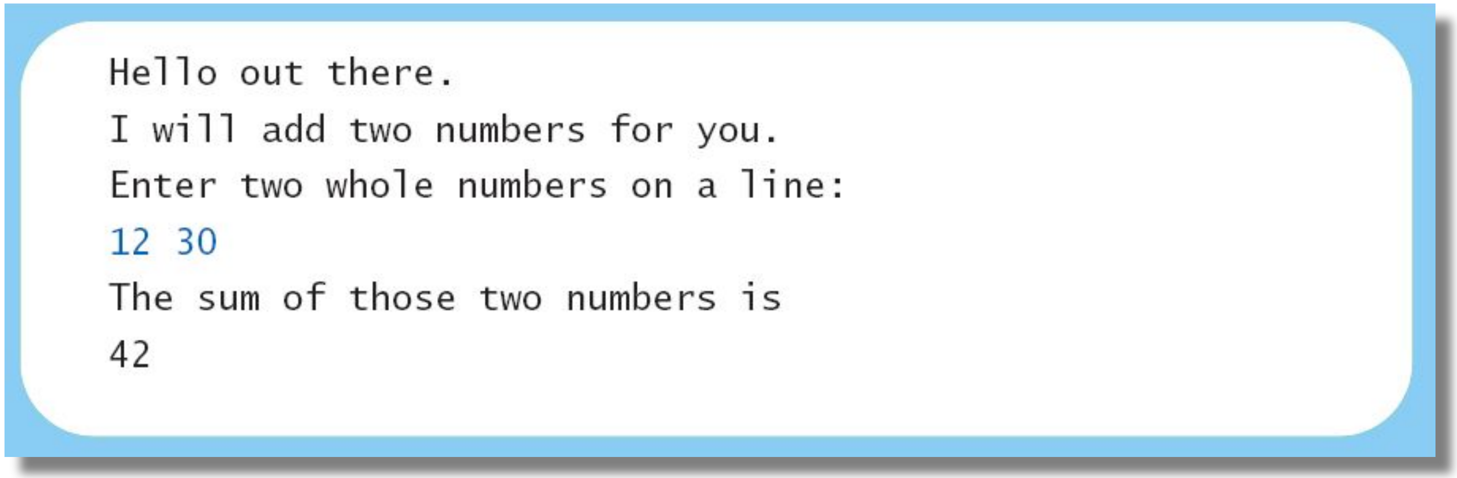
- Computer Basics
- **The First Java Application**
- Programming Basics
- Graphics Supplement

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);
    }
}
```

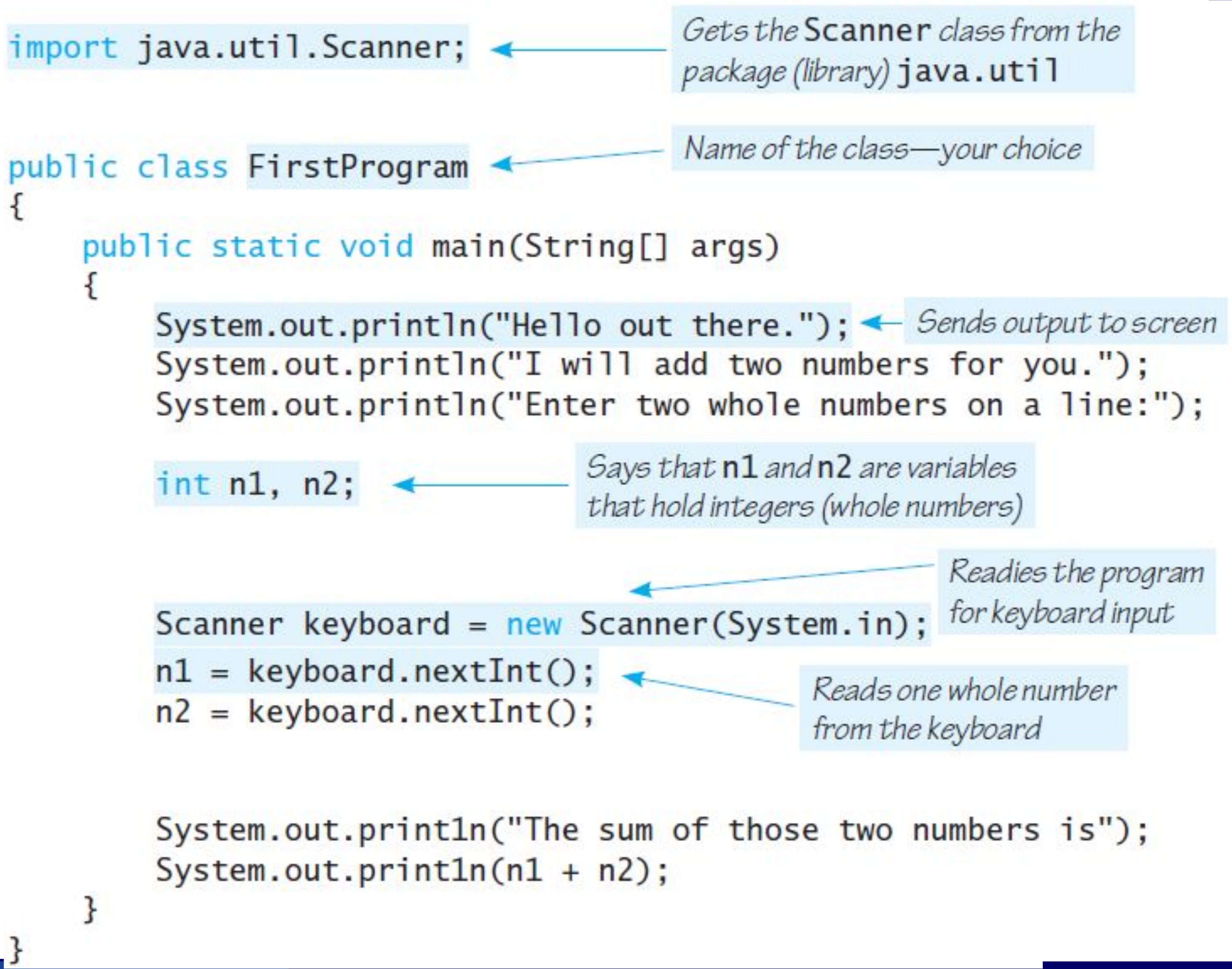
The First Java Application Program

- 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
```

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

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

```
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
  - In the code, we use 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
- **<u>Programming Basics</u>**
- 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 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, …

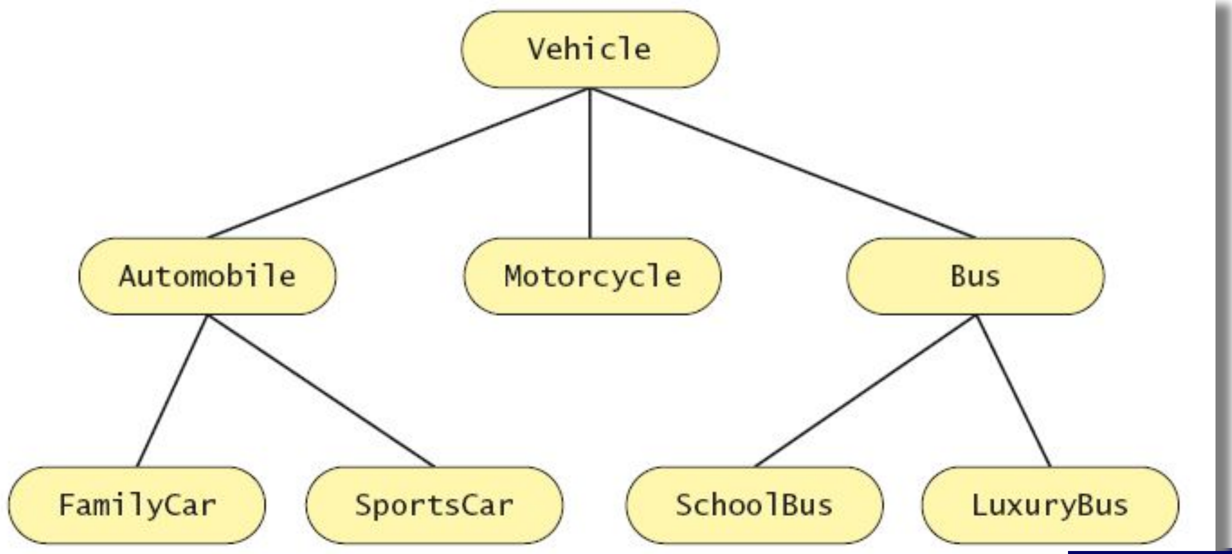
Polymorphism

- 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

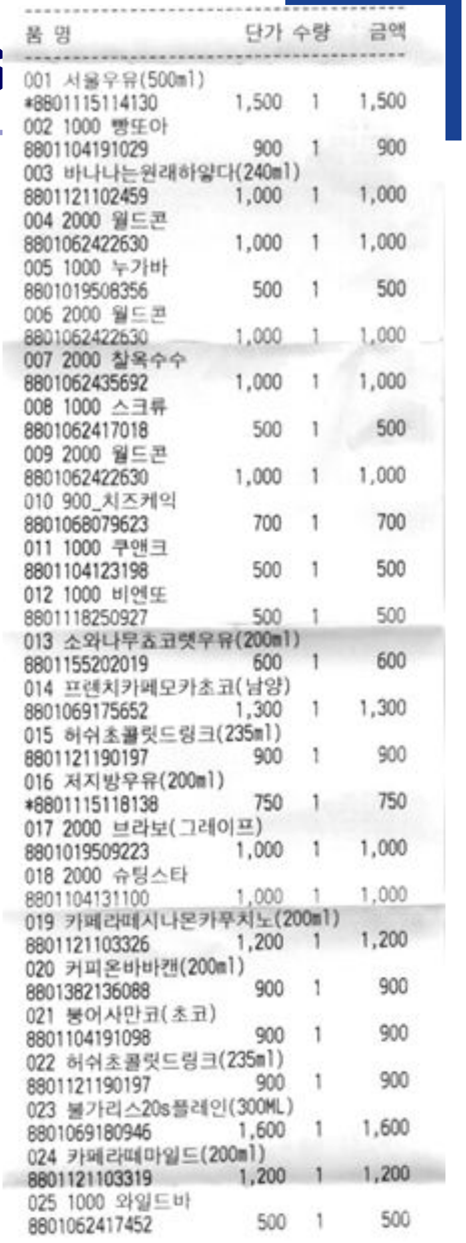
- Once an algorithm is defined, expressing it in Java (or in another programming language) usually is easy.

- 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.



```
---------------------------------------------
품 명                      단가 수량     금액
---------------------------------------------
001 서울우유(500ml)
*8801115114130            1,500    1    1,500
002 1000 빵또아
8801104191029               900    1      900
003 바나나는원래하얗다(240ml)
8801121102459             1,000    1    1,000
004 2000 월드콘
8801062422630             1,000    1    1,000
005 1000 누가바
8801019508356               500    1      500
006 2000 월드콘
8801062422630             1,000    1    1,000
007 2000 찰옥수수
8801062435692             1,000    1    1,000
008 1000 스크류
8801062417018               500    1      500
009 2000 월드콘
8801062422630             1,000    1    1,000
010 900_치즈케익
8801068079623               700    1      700
011 1000 쿠앤크
8801104123198               500    1      500
012 1000 비엔또
8801118250927               500    1      500
013 소와나무쵸코렛우유(200ml)
8801155202019               600    1      600
014 프렌치카페모카쵸코(남양)
8801069175652             1,300    1    1,300
015 허쉬초콜릿드링크(235ml)
8801121190197               900    1      900
016 저지방우유(200ml)
*8801115118138              750    1      750
017 2000 브라보(그레이프)
8801019509223             1,000    1    1,000
018 2000 슈팅스타
8801104131100             1,000    1    1,000
019 카페라떼시나몬카푸치노(200ml)
8801121103326             1,200    1    1,200
020 커피온바바캔(200ml)
8801382136088               900    1      900
021 붕어사만코(초코)
8801104191098               900    1      900
022 허쉬초콜릿드링크(235ml)
8801121190197               900    1      900
023 불가리스20s플레인(300ML)
8801069180946             1,600    1    1,600
024 카페라떼마일드(200ml)
8801121103319             1,200    1    1,200
025 1000 와일드바
8801062417452               500    1      500
```

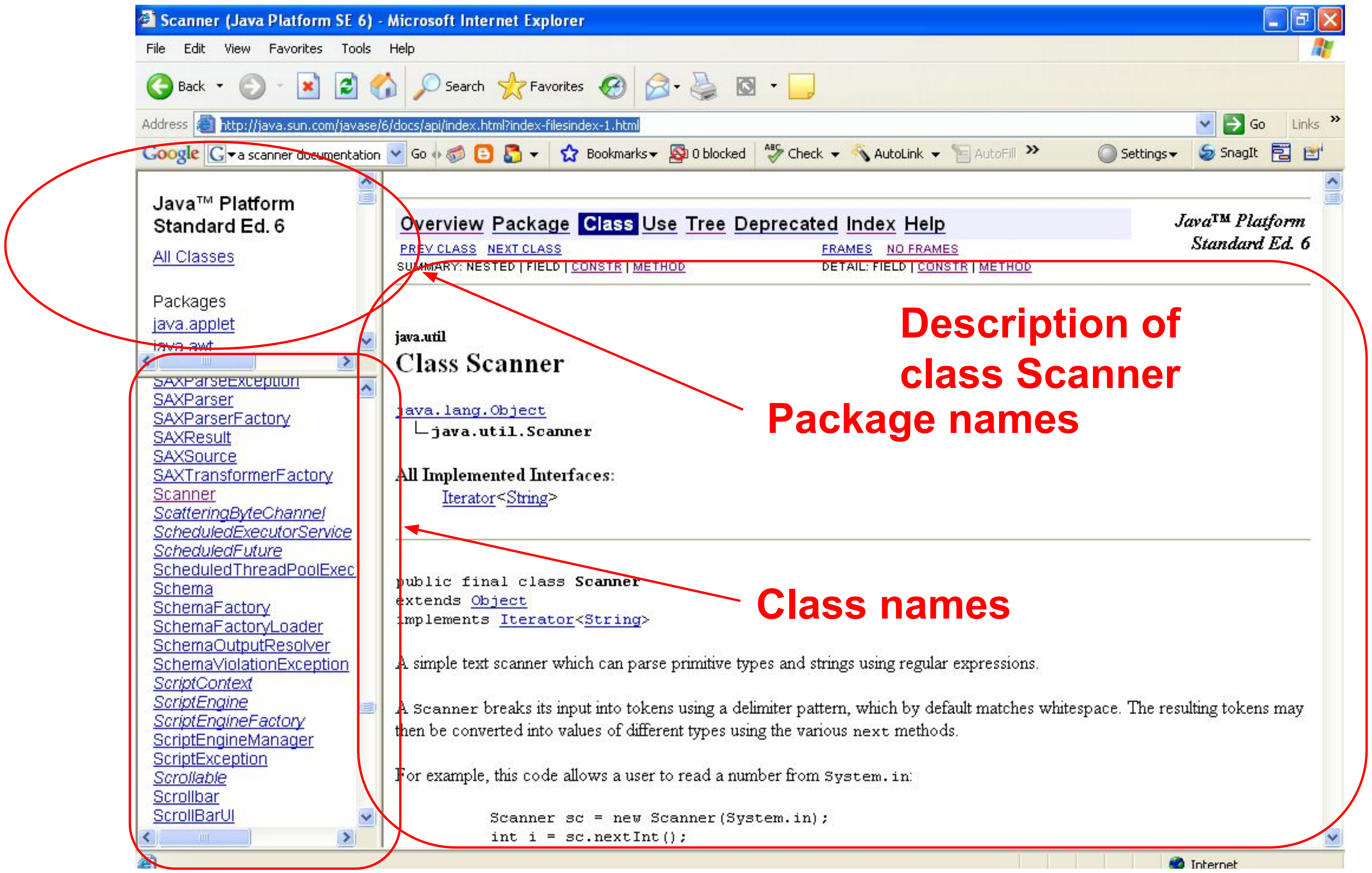
Reusable Components

- Most programs are created by combining existing components.
  - Programs 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

  http://docs.oracle.com/javase/7/docs/api/

Java Platform API

It moved to http://docs.oracle.com/javase/7/docs/api/


Scanner (Java Platform SE 6) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://java.sun.com/javase/6/docs/api/index.html?index-filesindex-1.html
Go Links
Description of class Scanner
Package names
Class names

Java™ Platform Standard Ed. 6

All Classes

Packages
java.applet
java.awt

SAXParseException
SAXParser
SAXParserFactory
SAXResult
SAXSource
SAXTransformerFactory
Scanner
ScatteringByteChannel
ScheduledExecutorService
ScheduledFuture
ScheduledThreadPoolExec
Schema
SchemaFactory
SchemaFactoryLoader
SchemaOutputResolver
SchemaViolationException
ScriptContext
ScriptEngine
ScriptEngineFactory
ScriptEngineManager
ScriptException
Scrollable
Scrollbar
ScrollBarUI

Overview Package **Class** Use Tree Deprecated Index Help

Java™ Platform Standard Ed. 6

PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

java.util
Class Scanner

```
java.lang.Object
  └java.util.Scanner
```

All Implemented Interfaces:
Iterator<String>

```
public final class Scanner
extends Object
implements Iterator<String>
```

A simple text scanner which can parse primitive types and strings using regular expressions.

A `Scanner` breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various `next` methods.

For example, this code allows a user to read a number from `System.in`:

```
        Scanner sc = new Scanner(System.in);
        int i = sc.nextInt();
```