Object Oriented Programming

Introduction to Java

Ch. 1. Introduction to Programming, Computer Basics

School of AI. Software, Gachon University
Ahyoung Choi

Computer Basics: Outline

- Hardware and Memory
- Programs
- Programming Languages and Compilers
- Java Byte-Code

Hardware and Software

- Computer systems consist of *hardware* and *software.*
  - Familiarity with hardware basics helps us understand software.
- Hardware
  - *Tangible* parts of computer systems (physical machine)
  - e.g., CPU, Memory
- Software
  - *programs* - sets of instructions for the computer to follow.
  - Programs give instructions to the computer
  - e.g., Google Chrome, Eclipse, ..

Hardware and Software

- Hardware
  - Circuits that execute, store and interact with instructions
    - Execution: CPU
    - Storage: Memory
    - Interaction: Peripherals, like keyboards, monitors, networks
- Software
  - An organized collection of instructions

Hardware and Memory

- Most modern computers have similar components including
  - Input devices (keyboard, mouse, etc.)
  - Output devices (display screen, printer, etc.)
  - A processor
  - Two kinds of memory (main memory and auxiliary(secondary) memory).

Instructions

- An instruction is a sequences of 0’s and 1’s that represents a single operation on the computer
  - Example: 00000101 00000001 00000010
  - $\text{Means:}\quad \underbrace{\text{ADD}}_{\textit{Instruction}} \quad \underbrace{1 \qquad 2}_{\textit{Data}}$
  - The output will be 3
  - These 0’s and 1’s are called **bits** Why only 0 and 1?

Why just 0s and 1s?

- Because it is easy to make an electrical device that has only two stable states
- Machines with only 2 stable states are easy to make, but programming using only 0s and 1s is difficult.
- Fortunately, the conversion of numbers, letters, strings of characters, audio, video, and programs is done automatically.
  – You need not be concern about the conversion.

Processor (CPU)

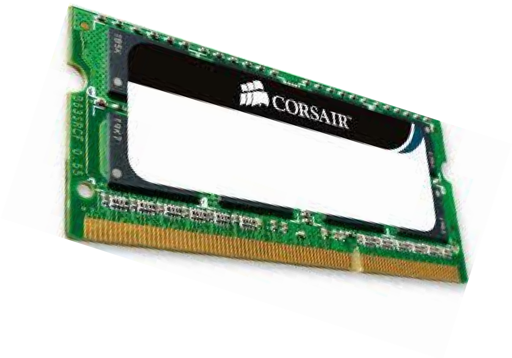
- Also called the *CPU* (Central Processing Unit) or the *chip* (e.g. Pentium processor)
- It is the "brain" of the computer CPU executes the instructions
  - CPU's working routine
    - read instructions and data from memory
    - do calculation
    - write calculation results back to memory

- Intel Core i7 **3.4 GHz**
  - Executes *at most* 3,400,000,000 instructions per second

Memory

- Memory holds
  - programs
  - data for the computer to process
  - the results of intermediate processing.
- Two kinds of memory
  - main memory
  - auxiliary/secondary memory

Main memory



- Working memory used to store
  - Current program
  - Data that program is using
  - Results of intermediate calculations
- Disappears when you shut down your computer
- Usually measured in megabytes, gigabytes (e.g. 256 Mbytes , 8Gbyes of RAM)
  - RAM is short for *random access memory*
  - A *byte* is a quantity of memory

Auxiliary Memory

- Also called *secondary memory*
- Disk drives, CDs, DVDs etc.
- More or less permanent (nonvolatile)
- Usually measured in gigabytes (e.g. 500 gigabyte hard drive)

Bits, Bytes, and Addresses

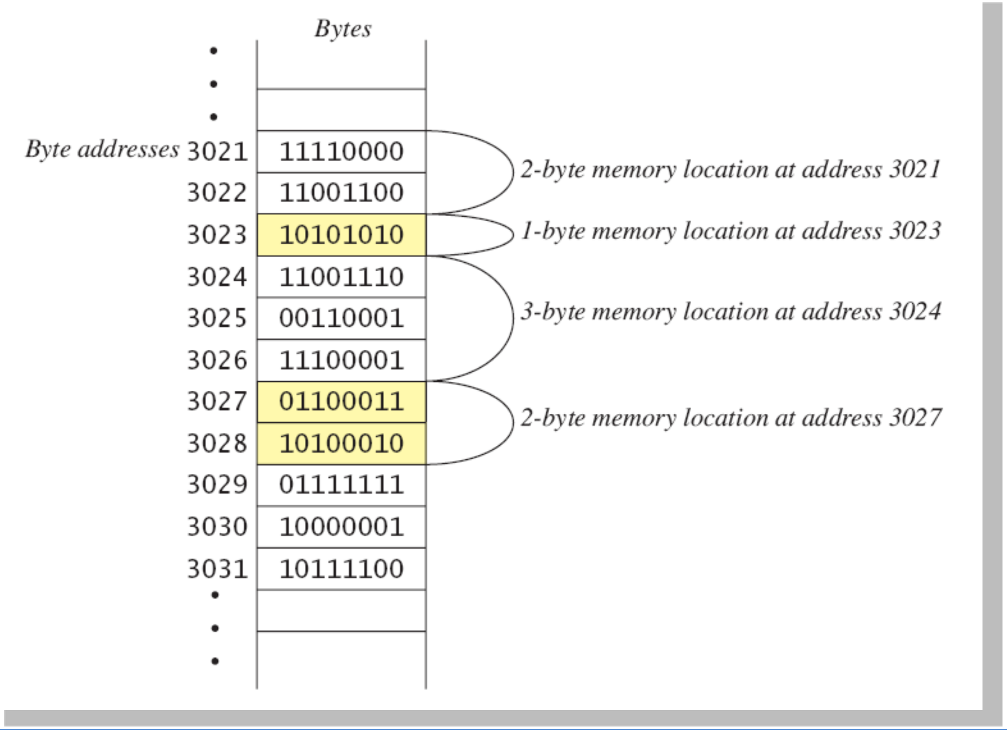
- A *bit* is a digit with a value of either 0 or 1.
- A *byte* consists of 8 bits.
- Each byte in main memory resides at a numbered location called its *address*.

GB? MB? KB?

- 1 bit = 0 or 1
- 1 byte = 8 bits
  - Smallest addressable unit of memory
- Kilo, Mega, Giga, Tera
  - 1 KB = 1,000 bytes (1 thousand bytes)
  - 1 MB = 1,000 KB = 1,000,000 bytes (1 million bytes)
  - 1 GB = 1,000 MB = 1,000,000,000 bytes (1 billion bytes)
  - 1 TB = 1,000 GB = 1,000,000,000,000 bytes!

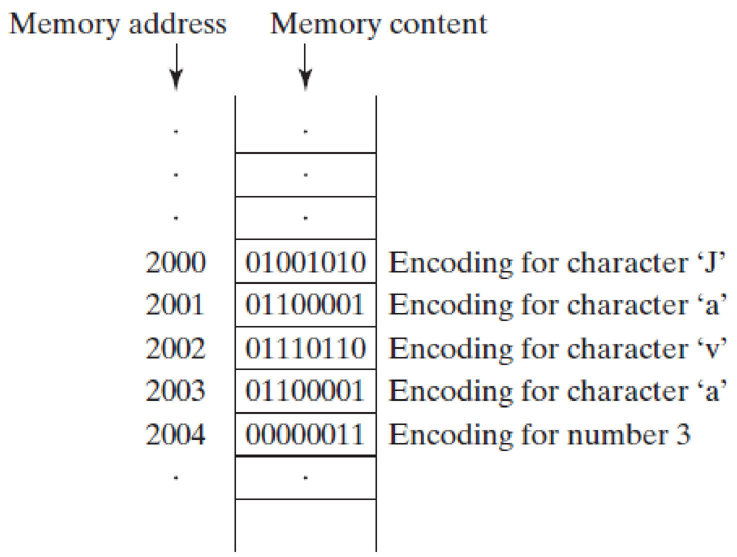
Main Memory

- Memory address
  - To locate certain memory positions
  - CPU fetches data according to memory address



Storing Data

- Data of all kinds (numbers, letters, strings of characters, audio, video, even programs) are encoded and stored using 1s and 0s.
- When more than a single byte is needed, several adjacent bytes are used.
  - The address of the first byte is the address of the unit of bytes.



Files

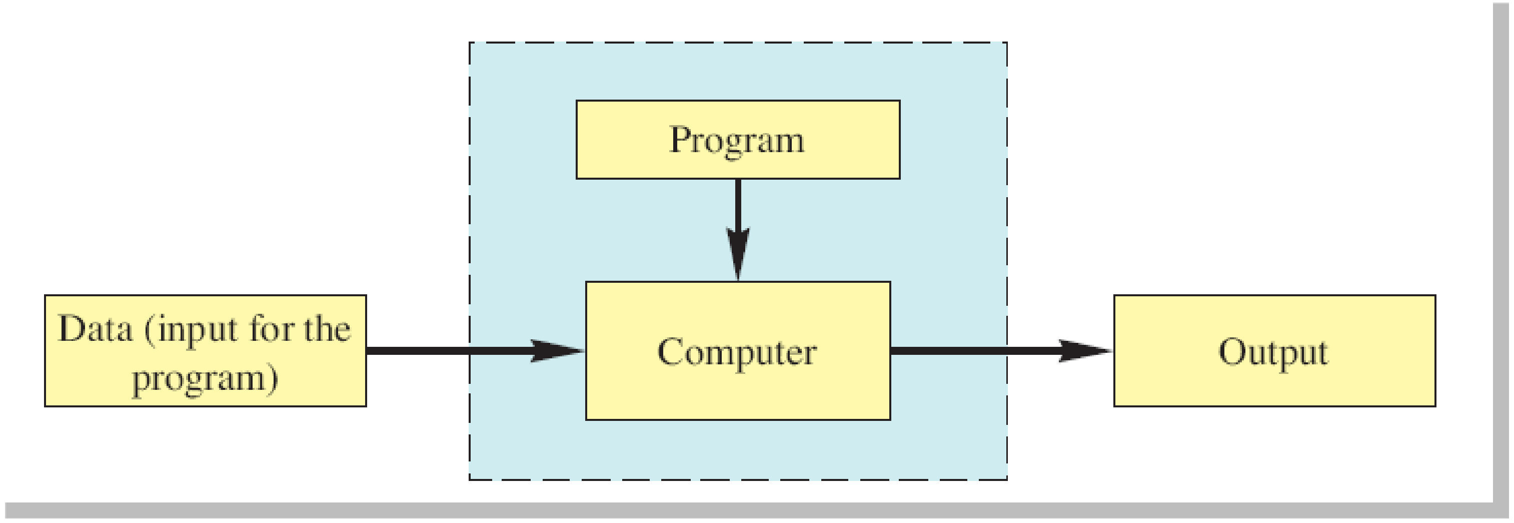
- Large groups of bytes in auxiliary memory are called *files*.
- Files have names.
- Files are organized into groups called *directories* or *folders*.
- Java programs are stored in files.
- Programs files are copied from auxiliary memory to main memory in order to be run.

Programs

- A set of instructions for a computer to follow
  - Also known as software
  - We use programs almost daily (email, word processors, video games, bank ATMs, etc.).
- “Following the instructions” is called *running* or *executing* the program.

- You will be writing programs (in Java)

Running a Program



- Sometimes the computer and the program are considered to be one unit.
  - Programmers typically find this view to be more convenient.

Input and Output

- Normally, a computer receives two kinds of input:
  - The program
  - The *data* needed by the program.
- The output is the result(s) produced by following the instructions in the program.

Programming Languages

- Why do we need languages when we have instructions?

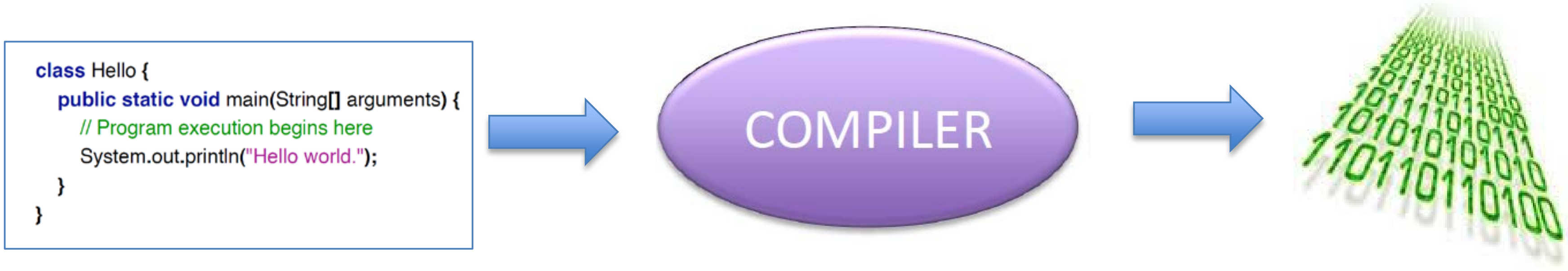
- A: Too hard for humans to write bits directly

Programming Languages

- *High-level languages* are relatively easy to use
  - Java, C#, C++, Visual Basic, Python, Ruby.
- Unfortunately, computer hardware does not understand high-level languages.
  - Therefore, a high-level language program must be translated into a *low-level language.*

From Languages to Instructions

- The translator is called **compiler**
  - It is also a program
  - From human-readable to machine-readable



```
class Hello {
   public static void main(String[] arguments) {
      // Program execution begins here
      System.out.println("Hello world.");
   }
}
```

Compiler vs. Interpreter

Compiler vs. Interpreter

Compilers

- A *compiler* translates a program from a high-level language to a low-level language.
- Compilers produce *machine-* or *assembly-language* programs called *object programs.*
- *e.g.* C, C++, JAVA, C# etc.

```
void invert (unsigned char *image, int width, int height) {
  for (int i=0; i<width*height; i++)
  image[i]=255-image[i];
}
```

```
addcc %r1,-4,%r1      10000010 10000000 01111111 11111100
addcc %r1,%r2,%r4     10001000 10000000 01000000 00000010
ld %r4,%r5            11001010 00000001 00000000 00000000
ba loop               00010000 10111111 11111111 11111011
addcc %r3,%r5,%r3     10000110 10000000 11000000 00000101
jmpl %r15+4,%r0       10000001 11000011 11100000 00000100
```

Interpreters

- A *interpreter* alternates the translation and execution of statements in a program
- Recall that compilation is done once, and the resulting objet program can be run over and over again (.exe)
- A compiled program generally runs faster than an interpreted one
- *e.g.* Javascript, HTML, SQL, Python, Ruby etc.

NOTE: It is very important to understand the difference between compilers and interpreters

Disadvantage of translation

- Most high-level languages need a different compiler or interpreter for each type of computer and for each operating system.
- However, compilers and interpreters are large programs that are expensive and time-consuming to write
- Furthermore, each system requires unique libraries or syntax of source programs.
  - It means that programmers need to modify the source programs for each system.
  - E.g., A source code may need to be modified to run in a different system (OS)

Question..

- One of the significant advantages of Java is
  - Java is **platform-independent**.
  - Java programs can be moved easily from one computer system to another.
    - The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels.
- How is this possible ?

Java Byte-Code

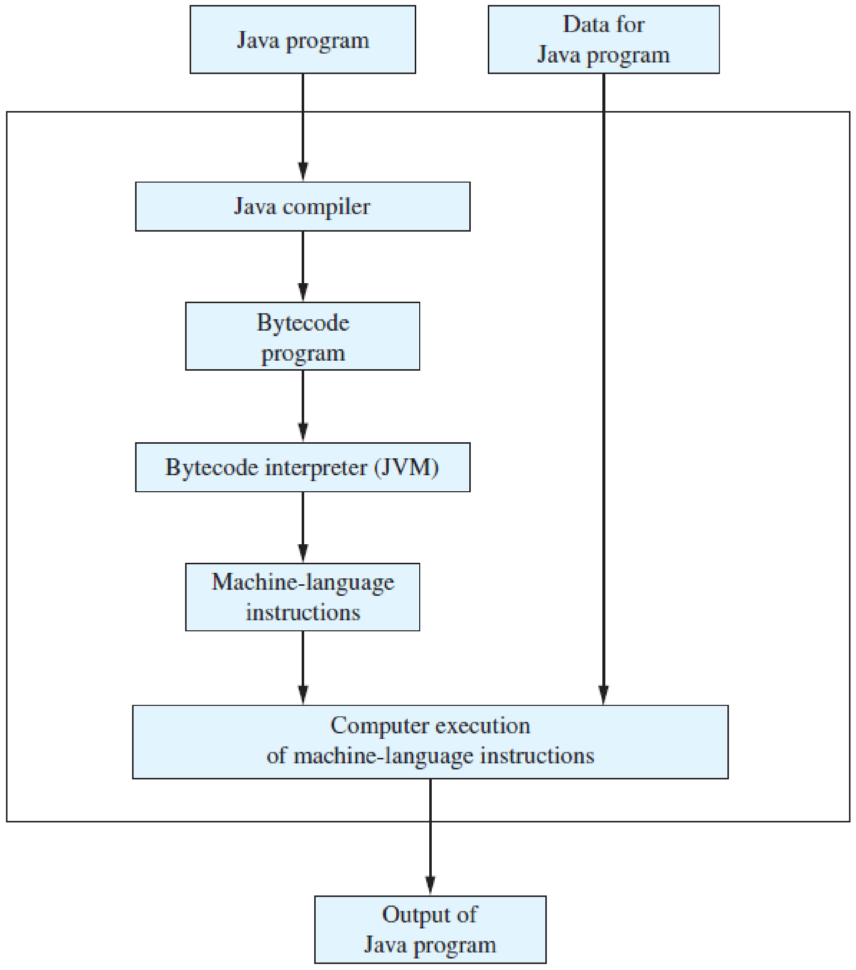
- The Java *compiler* does not translate a Java program into *assembly language* or *machine language* for a particular computer.
- Instead, it translates a Java program into *byte-code*.
  - Byte-code is the machine language for a hypothetical computer (or *interpreter*) called the **Java Virtual Machine**.

Java **Byte-Code**

- A byte-code program is easy to translate into machine language for any particular computer.
- A program called an *interpreter* translates each byte-code instruction, executing the resulting machine-language instructions on the particular computer before translating the next byte-code instruction.

Running a Java Program

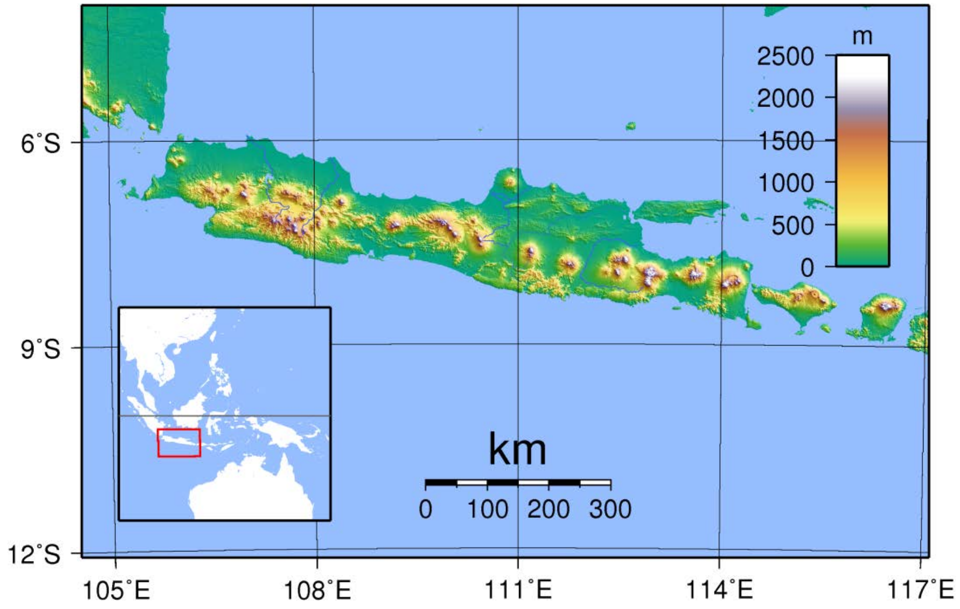
FIGURE 1.3 **Compiling and Running a Java Program**


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

Portability

- After compiling a Java program into byte-code, that byte-code can be used on any computer with a byte-code interpreter and without a need to recompile.
  - Of course, different systems may have different interpreters (JVMs) for their systems
- Byte-code can be sent over the Internet and used anywhere in the world.
- This makes Java suitable for Internet applications.

A SIP of JAVA



Java n. An island of Indonesia, 48,842 square miles in area, lying between the Indian Ocean and the Java Sea.
java n. Informal. Brewed coffee. [From Java.]

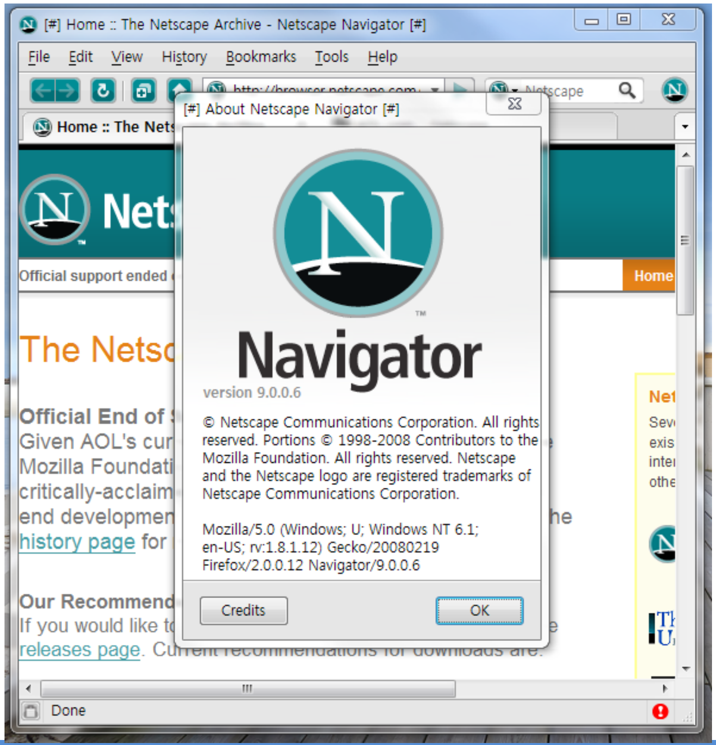
History of Java

- In 1991, James Gosling and Sun Microsystems began designing a language for home appliances (toasters, TVs, etc.).

✓ Challenging, because home appliances are controlled by many different chips (processors)
✓ Programs were translated first into an intermediate language common to all appliance processors.
✓ Then the intermediate language was translated into the machine language for a particular appliance's processor.

History of Java

- In 1994, Gosling realized that the language would be ideal for Internet applications
  - Especially for Web browsers that could run programs over the Internet
  - Sun produced the browser known today as HotJava
  - In 1995, Netscape capable of running Java was developed



[#] Home :: The Netscape Archive - Netscape Navigator [#]
File Edit View History Bookmarks Tools Help
[#] About Netscape Navigator [#]
Navigator
Credits
OK
Done

version 9.0.0.6

© Netscape Communications Corporation. All rights reserved. Portions © 1998-2008 Contributors to the Mozilla Foundation. All rights reserved. Netscape and the Netscape logo are registered trademarks of Netscape Communications Corporation.

Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.8.1.12) Gecko/20080219 Firefox/2.0.0.12 Navigator/9.0.0.6

Why is the language named Java?

- The original name of the Java language was Oak
  - Later the creators realized that there already was a computer language named Oak
- One traditional story is that the name was thought of during a long and tedious meeting while the participants drank coffee, and the rest, as they say, is history

Applications and Applets

- Two kinds of java programs: *applications* and *applets*
- Applications
  - Regular programs
  - Meant to be run on your computer
- Applets
  - Little applications
  - An Applet is meant to be sent to another location on the internet and run there

A First Java Application

We are ready for our First Application

- Question: Assume that your first program is in the memory. Now you compile it, then run it. It reads two numbers from the keyboard and output the result to the screen. **What happens inside your computer during the procedure?**

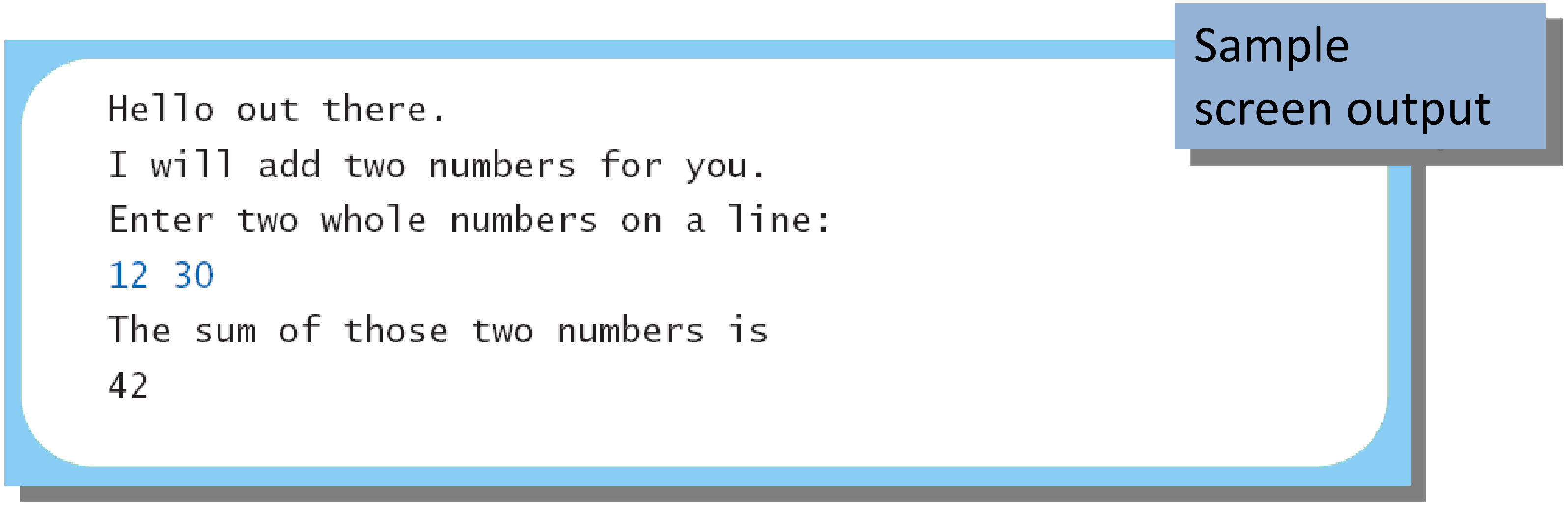
- Write down the answer.

Answer

- The compiler translates your program to instructions
  - CPU reads your Java program from memory
  - CPU does some calculation (or transformation)
  - CPU writes the results (instructions) back to memory
- Your program is now in memory as instructions
  - CPU reads these instructions from memory
  - The keyboard input is stored in a specific memory address
  - CPU reads the input and computes the result
  - CPU writes the result to another memory address (screen)

Our First Java Application

- Write a following Java program
  - Read two integers and display the number of integers between them, including themselves
  - E.g., print 4 for given two integers 3 and 6

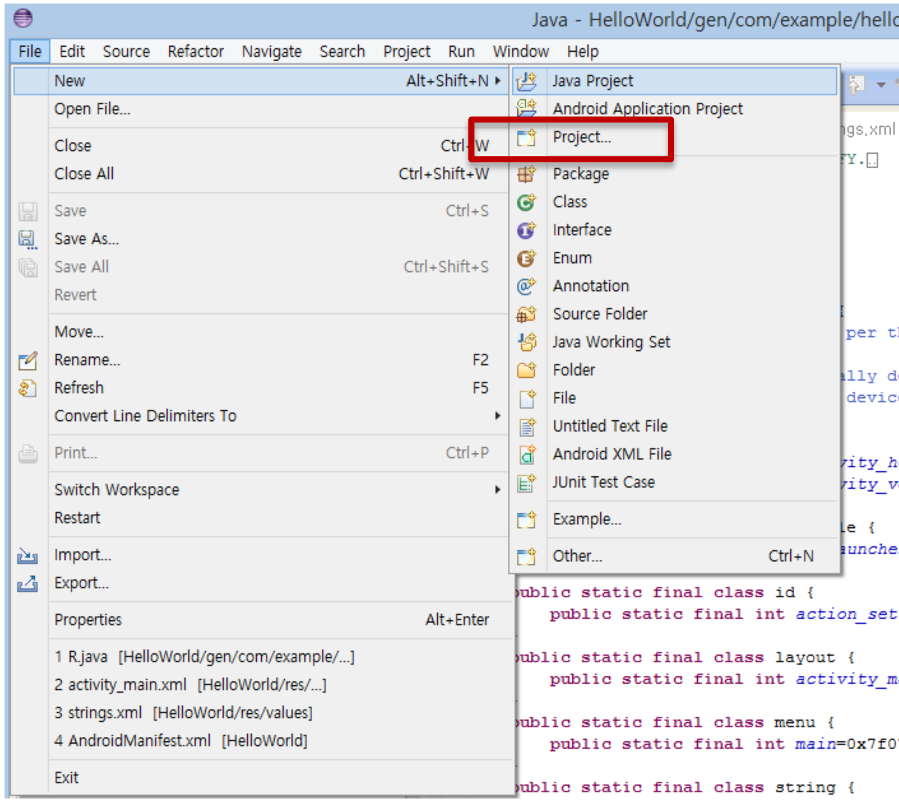


Sample screen output

```
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
```

Lab: New Java Project

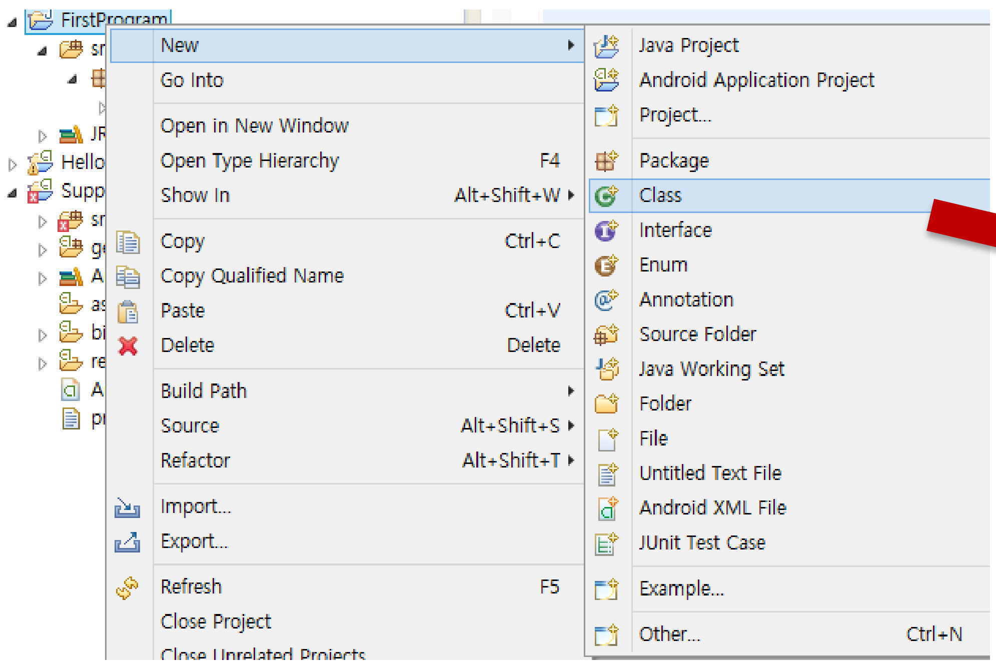
- New project
  - [New] – [Java Project]
  - Project Name : **FirstProgram**

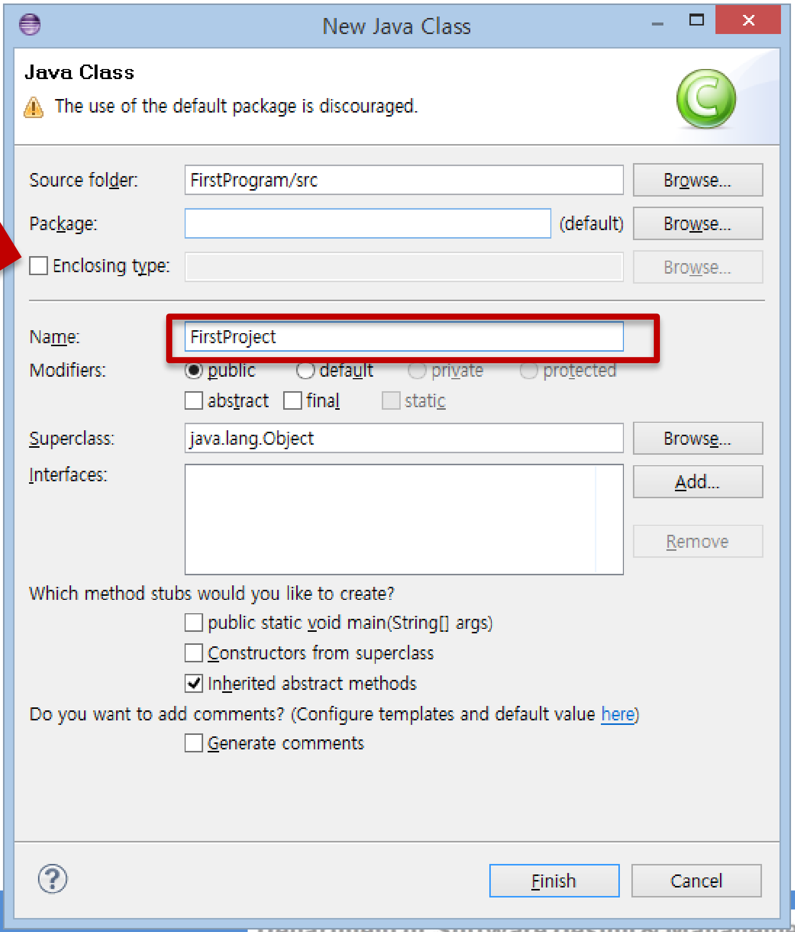


```
ublic static final class id {
    public static final int action_set
ublic static final class layout {
    public static final int activity_m
ublic static final class menu {
    public static final int main=0x7f0
ublic static final class string {
```

Lab: New Java Project

- Right Click on your project (in Project Explorer)





New Java Class

Java Class

The use of the default package is discouraged.

Source folder: FirstProgram/src

Package: (default)

☐ Enclosing type:

Name: FirstProject

Modifiers: [✓ public] default private protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

Lab: Our First 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 number 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);
    }
 }
```

Run !

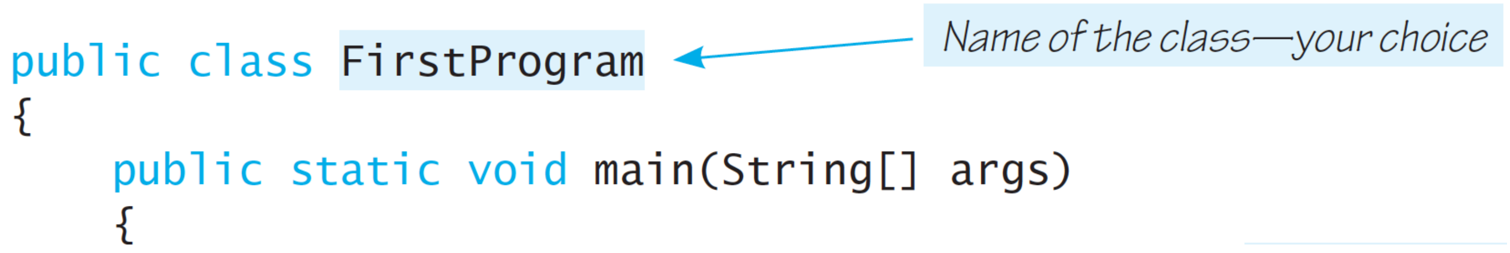
Sample Screen Output

```
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
```

Some Terminology: Import Package

- Import := borrow something from somewhere else
- *package* := a library of classes that have been defined already.
  - `import java.util.Scanner;`
- ***java.util*** is a package that contains useful standard tools
  - ***java.math*** contains mathematical tools
  - ***java.net*** contains network connection tools
  - Scanner is one standard tool about keyboard inputting in this package

Begin the Program



```
public class FirstProgram
{
    public static void main(String[] args)
    {
```

Name of the class—your choice

- Begin a program named ***FirstProgram***
  - Program names should make sense
    - Example: Another name for this program could be ***AddTwoNumbers***
- You should always capitalize the first letter or each word in your program name

Printing to the Screen

- This three lines write what in the quote to screen

```
System.out.println("Hello out there.");
System.out.println("I will add two number for you.");
System.out.println("Enter two whole numbers on a line:");
```

- Remember **`System.out.println(“”)`** can print whatever is in parentheses to the screen.

Read from Keyboard

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

- With the help of ***Scanner***, we read two numbers from the keyboard
  - These two numbers are saved in certain memory locations, represented by ***n1*** and ***n2***

Again

```
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

Calculate and output the result

There is no quotation mark, because you want the computer to output the result of *n1+n2*, not the text *"n1+n2"*

Concept: Class and Object

- **Class**: a piece of code we can use in a program
  - It is an abstract specification of a category
- **Object**: a piece of data/instructions in memory
  - It is a member of a class
  - It is something that actually exists (but still an abstraction)

- Example
  - ***Car***: it can be a class
  - ***Alice's Car, Bob's Car***: they are objects, in the class of *Car*

Some Terminology

- The item(s) inside parentheses are called *argument(s)* and provide the information needed by methods.
- A *variable* is something that can store data.
- An instruction to the computer is called a *statement*; it ends with a semicolon.

```
System.out.println("Hello out there.");
```

- The grammar rules for a programming language are called the *syntax* of the language.

Some Terminology : Method

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

- A class contains methods
  - An action which an object is able to perform
  - Every Java application has a method called main
- The object performs an action when you *invoke* or *call* one of its methods

```
objectName.methodName(argumentsTheMethodNeeds);
```

EXAMPLES:

```
System.out.println("Hello out there.");
n1 = keyboard.nextInt();
```

Compiling a Java Program or Class

- A Java program consists of one or more classes, which must be compiled before running the program.
- You need not compile classes that accompany Java (e.g. `System` and `Scanner`).
- Each class should be in a separate file.
- The name of that file must be the name of the class, with **.java** added to the end.
  - E.g., The class *FirstProgram* must be in a file named *FirstProgram.java*

Compiling and Running

- Use an *IDE* (integrated development environment) which combines a text editor with commands for compiling and running Java programs.
  - Eclipse is the most popular IDE for JAVA
  - www.eclipse.org
- When a Java program is compiled, the byte-code version of the program has the same name, but the ending is changed from **`.java`** to **`.class`**.

Compiling and Running

- A Java program can involve any number of classes.

- The class to run will contain the words

```
public static void main(String[] args)
```

Compile & Run Without Eclipse (1)

(1) Create the source file:

- open a text editor, type in the code which defines a class (*HelloWorldApp*) and then save it in a file (*HelloWorldApp.java*)
- file and class name are case sensitive and must be matched exactly (except the `.java` part)

Example Code: HelloWorldApp.java

```
/**
 * The HelloWorldApp class implements an application
 * that displays "Hello World!" to the standard output
*/
public class HelloWorldApp {
  public static void main(String[] args) {
    // Display "Hello World!"
    System.out.println("Hello World!");
  }
}
```

★ `Java is CASE SENSITIVE!`

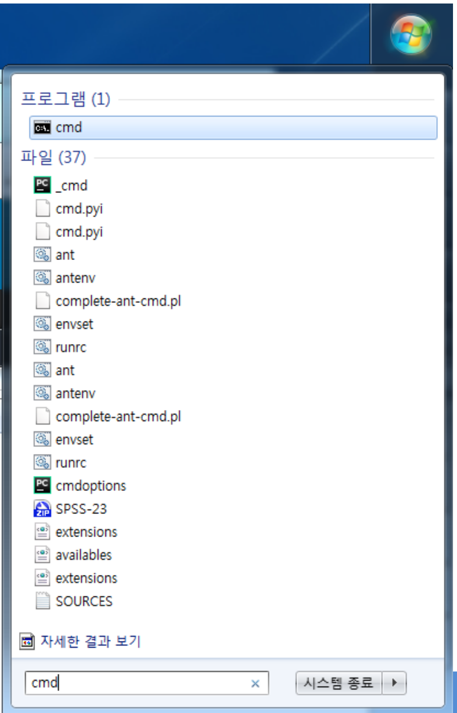
Compile & Run Without Eclipse: (2)

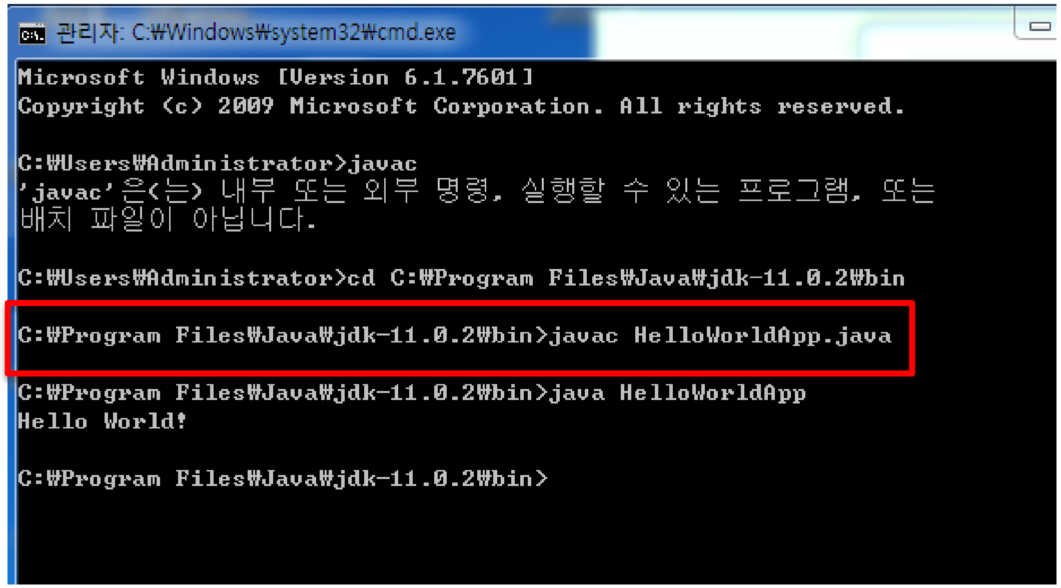
(2) Compile the program:

– compile HelloWorldApp.java by using the following command:

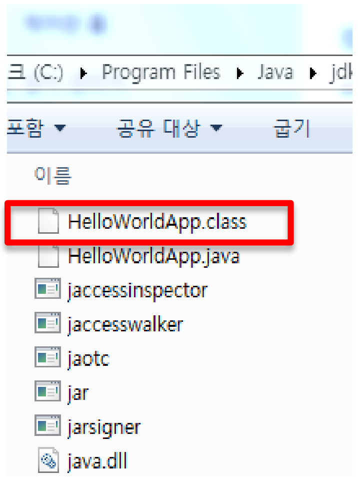
```
javac HelloWorldApp.java
```

it generates a file named HelloWorldApp.class




관리자: C:₩Windows₩system32₩cmd.exe

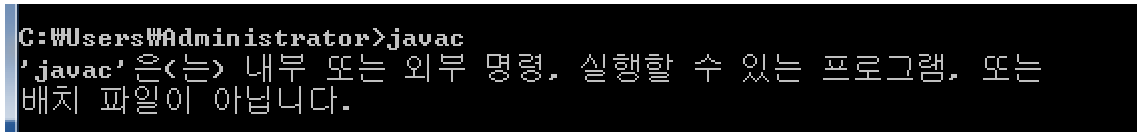
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:₩Users₩Administrator>javac
'javac'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.

C:₩Users₩Administrator>cd C:₩Program Files₩Java₩jdk-11.0.2₩bin

C:₩Program Files₩Java₩jdk-11.0.2₩bin>javac HelloWorldApp.java

C:₩Program Files₩Java₩jdk-11.0.2₩bin>java HelloWorldApp
Hello World!

C:₩Program Files₩Java₩jdk-11.0.2₩bin>
```



성공!

Compile & Run Without Eclipse: (2)

(2) Compile the program:



```
C:₩Users₩Administrator>javac
'javac'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.
```

`'javac' is not recognized as an internal or  external command, operable program or hatch file.`

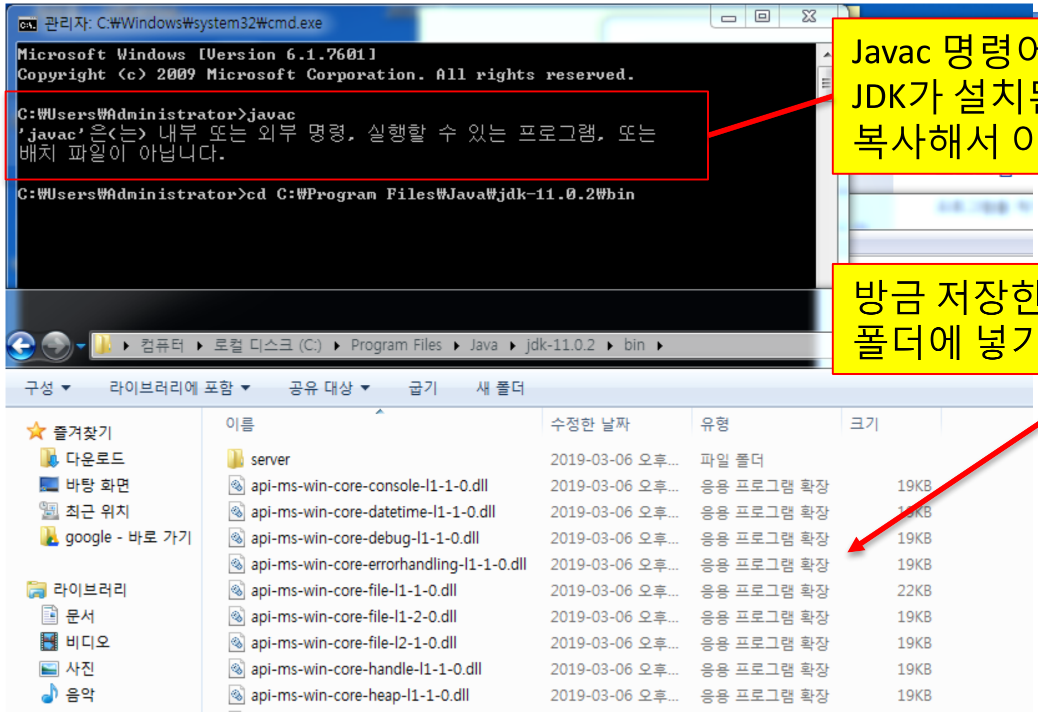
if you see one of these errors, you have two choices:

1) specify the full path in which the `javac` program locates every time. For example:

```
C:\j2sdk1.4.2_09\bin\javac HelloWorldApp.java
```

2) set the PATH environment variable

Compile & Run Without Eclipse: (2)



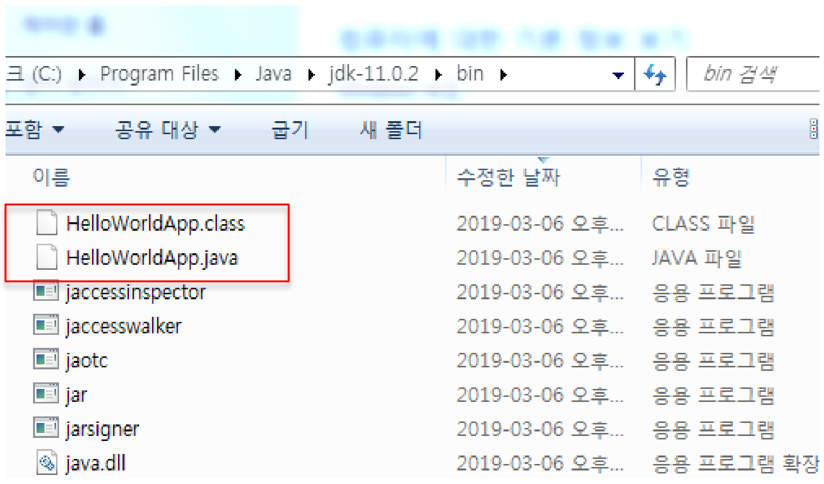
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>javac
'javac'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.

C:\Users\Administrator>cd C:\Program Files\Java\jdk-11.0.2\bin
```

컴퓨터 ▸ 로컬 디스크 (C:) ▸ Program Files ▸ Java ▸ jdk-11.0.2 ▸ bin ▸

이름	수정한 날짜	유형	크기
server	2019-03-06 오후...	파일 폴더	
api-ms-win-core-console-l1-1-0.dll	2019-03-06 오후...	응용 프로그램 확장	19KB
api-ms-win-core-datetime-l1-1-0.dll	2019-03-06 오후...	응용 프로그램 확장	19KB
api-ms-win-core-debug-l1-1-0.dll	2019-03-06 오후...	응용 프로그램 확장	19KB
api-ms-win-core-errorhandling-l1-1-0.dll	2019-03-06 오후...	응용 프로그램 확장	19KB
api-ms-win-core-file-l1-1-0.dll	2019-03-06 오후...	응용 프로그램 확장	22KB
api-ms-win-core-file-l1-2-0.dll	2019-03-06 오후...	응용 프로그램 확장	19KB
api-ms-win-core-file-l2-1-0.dll	2019-03-06 오후...	응용 프로그램 확장	19KB
api-ms-win-core-handle-l1-1-0.dll	2019-03-06 오후...	응용 프로그램 확장	19KB
api-ms-win-core-heap-l1-1-0.dll	2019-03-06 오후...	응용 프로그램 확장	19KB

Javac 명령어를 인식하지 못하면
JDK가 설치된 폴더 주소를
복사해서 이동 (cd 명령어)

방금 저장한 java 파일을 jdk bin
폴더에 넣기



크 (C:) ▸ Program Files ▸ Java ▸ jdk-11.0.2 ▸ bin ▸

이름	수정한 날짜	유형
HelloWorldApp.class	2019-03-06 오후...	CLASS 파일
HelloWorldApp.java	2019-03-06 오후...	JAVA 파일
jaccessinspector	2019-03-06 오후...	응용 프로그램
jaccesswalker	2019-03-06 오후...	응용 프로그램
jaotc	2019-03-06 오후...	응용 프로그램
jar	2019-03-06 오후...	응용 프로그램
jarsigner	2019-03-06 오후...	응용 프로그램
java.dll	2019-03-06 오후...	응용 프로그램 확장

Compile & Run Without Eclipse: (3)

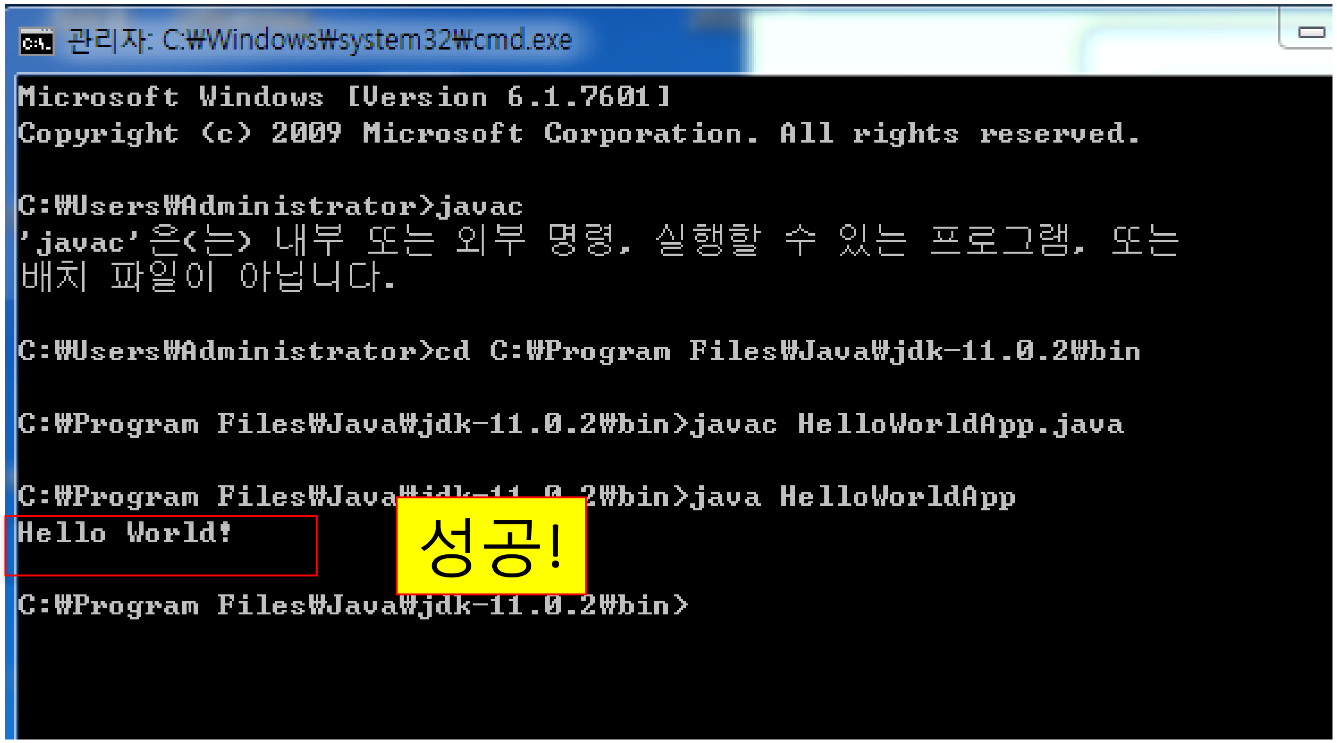
(3) Run the program:

- run the code through:

```
java HelloWorldApp
```

- Note that the command is `java`, not `javac`, and you refer to `HelloWorldApp`, not `HelloWorldApp.java` or `HelloWorldApp.class`

Compile & Run Without Eclipse: (3)



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:₩Users₩Administrator>javac
'javac'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.

C:₩Users₩Administrator>cd C:₩Program Files₩Java₩jdk-11.0.2₩bin

C:₩Program Files₩Java₩jdk-11.0.2₩bin>javac HelloWorldApp.java

C:₩Program Files₩Java₩jdk-11.0.2₩bin>java HelloWorldApp
Hello World!

C:₩Program Files₩Java₩jdk-11.0.2₩bin>
```

성공!

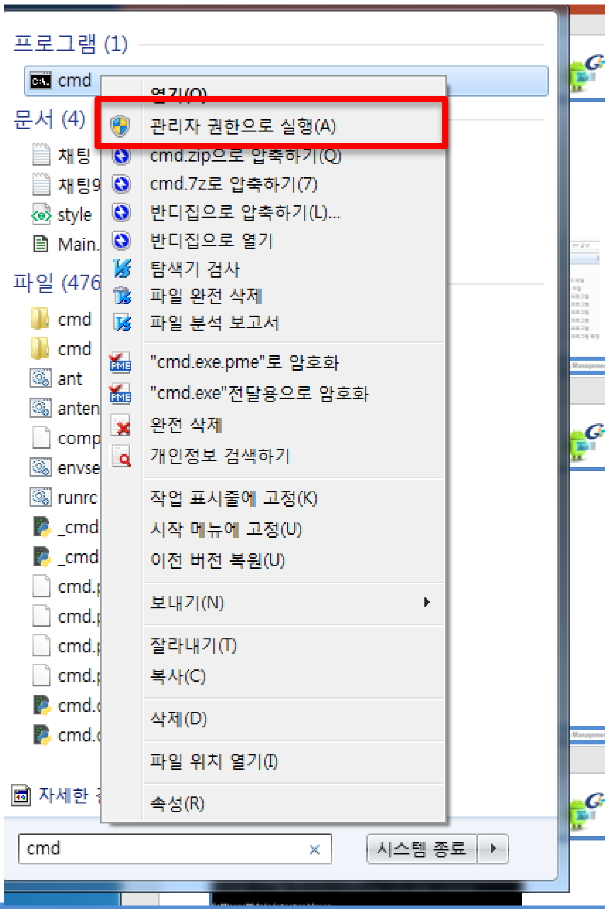
Errors

- if you see this error, you may need to set the environment variable CLASSPATH.

```
Exception in thread "main"
java.lang.NoClassDefFoundError:
    HelloWorldApp
```

- If you see this error, run CMD as
  – 관리자 권한으로 실행

```
\Program Files\Java\jdk-15.0.2\bin>javac Main.java
Main.java:1: error: error while writing Main:
C:\Program Files\Java\jdk-15.0.2\bin\Main.class
public class Main {
       ^
1 error
```



Practice 1.2

- Ex1_2a. Write a following Java program
  - Read your birth year and print your age
  - E.g., print 20 for the given birth year 1997
- Ex1_2b. Write a following Java program
  - Read two integers and display the number of integers between them, including themselves
  - E.g., print 4 for given two integers 3 and 6

PROGRAMMING BASICS

Object-Oriented Programming (OOP)

- Our world consists of *objects* (people, trees, cars, cities, airline reservations, etc.).
- Objects have details (attributes).
- Objects can perform *actions* which affect themselves and other objects in the world.
- Object-oriented programming (*OOP*) treats a program as a collection of objects that interact by means of actions.

vs. Procedural Programming

- Compared to **Procedural Programming**
  - Different ways to abstract the world
  - Procedural programming is **temporal** abstraction;
  - Object-oriented programming is **spatial** abstraction.

Example: Car Information System

- OOP thinks in objects:
  - The world has cars
    - Cars have *Make, Color, PlateNumber, Owner*, etc.
    - Cars can *Total(), Repaint(), ChangePlate()*, etc.
  - The world has drivers
    - Drivers have *Name, Age, Sex, LicenseNumber*, etc.
    - Drivers can *ChangeName(), GetOld(), RevokeLicense()*, etc.
  - Drivers and cars interact
    - A driver can have a car (or many cars)
    - A driver can buy/sell a car

- Procedural programming thinks in procedures:
  - **First**, we input all the car and driver records into system
    - We need to save them as variables
  - **Second**, we update the records in case
    - If a car is repainted, we change the color variable of the car
    - If a car is totaled, we delete the record of the car
    - If a driver changes his name, we change the name variable of this driver
    - ……
  - **Finally**, in the end of every day/week/month, we backup/print/report all information

Object-Oriented Programming

- OOP is easy to understand
  - We will learn some principles in OOP that are more complicated

- But we can not abandon procedural programming
  - Because we need procedural programming to write all the methods
  - Such procedures are called **algorithms**

OOP Terminology

- 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.

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.
  - Encapsulation often is called *information hiding.*

Encapsulation

Accessibility Example

- An automobile consists of several parts and pieces and is capable of doing many useful things.
  - Awareness of the accelerator pedal, the brake pedal, and the steering wheel is important to the driver.
  - Awareness of the fuel injectors, the automatic braking control system, and the power steering pump is not important to the driver.

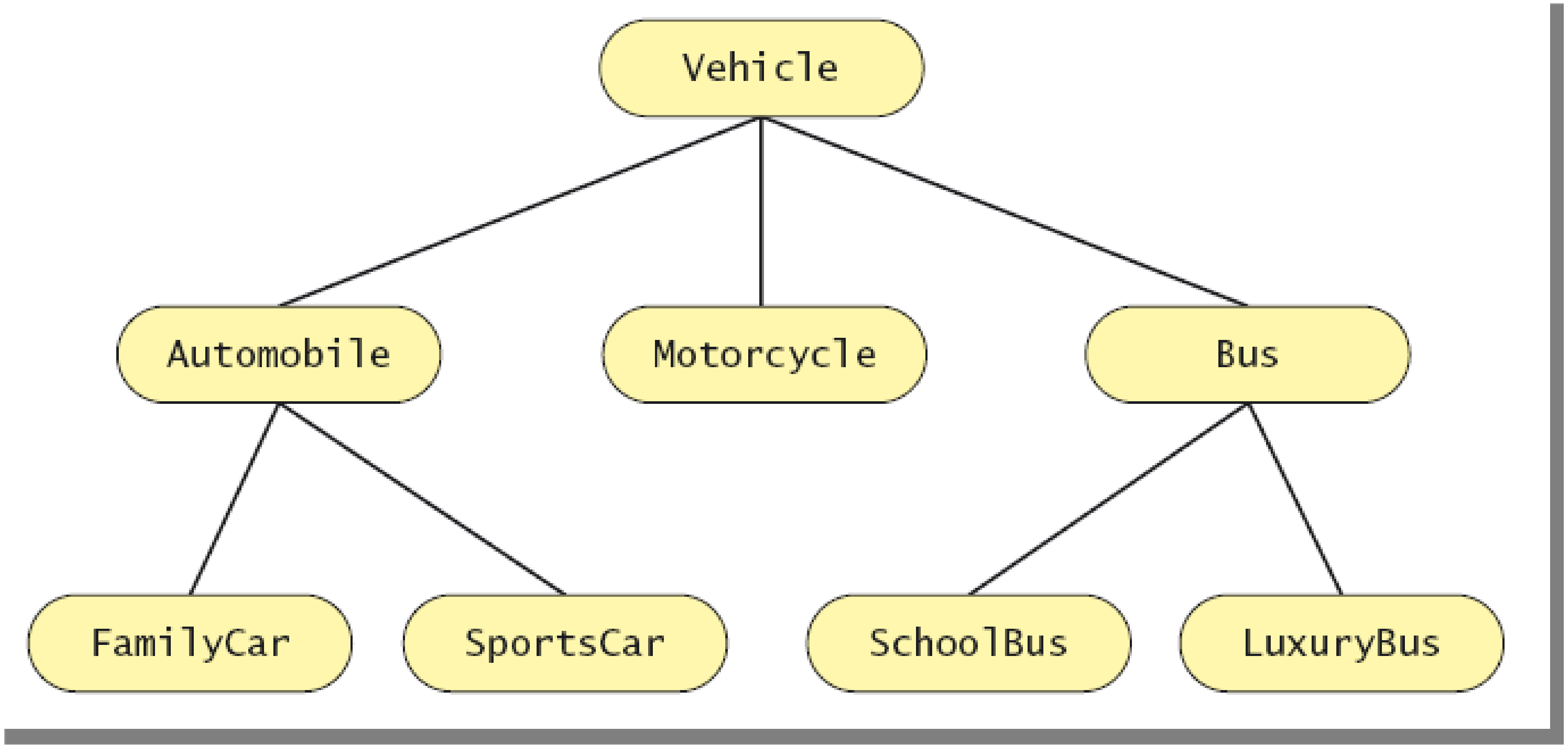
Polymorphism

- From the Greek meaning “many forms”
- The same program instruction adapts to mean different things in different contexts.
  - A method name, used as an instruction, produces results that depend on the class of the object that used the method.
  - Analogy: “Go play your favorite sport” causes different people to do different activities

Polymorphism

- Leg!

Inheritance



Inheritance

- Classes can be organized using *inheritance.*
- A class at lower levels inherits all the characteristics of classes above it in the hierarchy.
- At each level, classifications become more specialized by adding other characteristics.
- Higher classes are more inclusive; lower classes are less inclusive.

Inheritance in Java

- Used to organize classes
- “Inherited” characteristics do not need to be repeated.
- New characteristics are added.

Algorithms

- An algorithm is a set of instructions for solving a problem.
- An algorithm must be expressed completely and precisely.
- Algorithms usually are expressed in English or in *pseudocode.*

```
For i = 1 to 100
  set print_number to false
  if i mod 11 = 0
     set print_number to true
  if print_number
     print i and print a newline
```

Reusable Components

- Most programs are created by combining components that exist already.
- Reusing components saves time and money.
- Reused components are likely to be better developed, and more reliable.
- New components should designed to be reusable by other applications.

Errors

- An error in a program is called a *bug.*
- Eliminating errors is called *debugging.*
- Three kinds or errors
  - Syntax errors
  - Runtime errors
  - Logic errors

Syntax Errors

- Grammatical mistakes in a program
  - The grammatical rules for writing a program are very strict
- The compiler catches syntax errors and prints an error message.
- Example: using a period where a program expects a comma

Runtime Errors

- Errors that are detected when your program is running, but not during compilation
- When the computer detects an error, it terminates the program and prints an error message.
- Example: attempting to divide by 0

Logic Errors

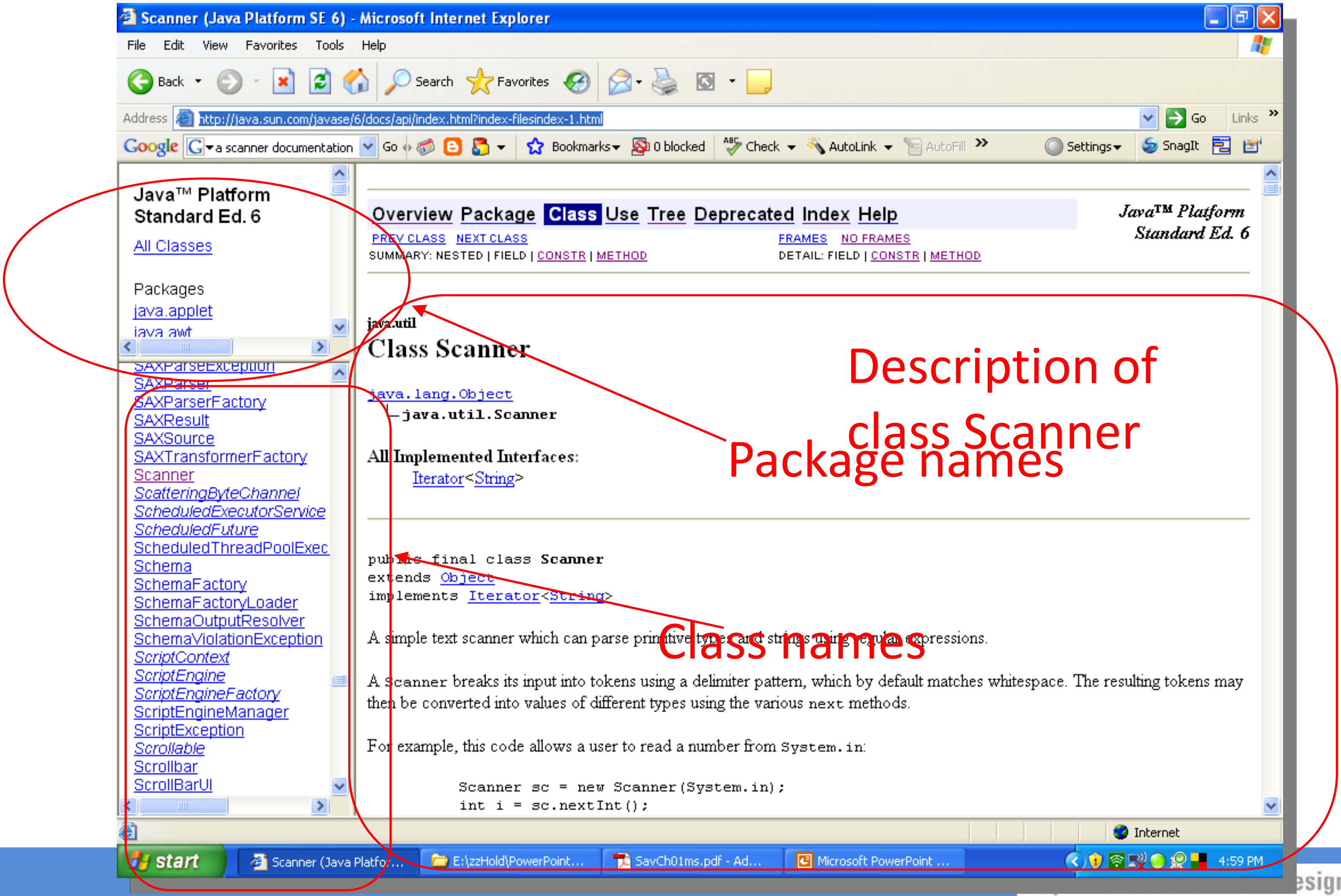
- Errors that are not detected during compilation or while running, but which cause the program to produce incorrect results
- Example: an attempt to calculate a Fahrenheit temperature from a Celsius temperature by multiplying by 9/5 and adding 23 instead of 32

Software Reuse

- Programs not usually created entirely from scratch
- Most contain components which already exist
- Reusable classes are used
  - Design class objects which are general
  - Java provides many classes
  - Note documentation on following slide

Documentation

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


Scanner (Java Platform SE 6) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://java.sun.com/javase/6/docs/api/index.html?index-filesindex-1.html
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();
```

Assignment

- Read Chapter 2.1-2.3