

# Getting Started with Java Programming

- A Simple Java Application
- Compiling Programs
- Executing Applications



# A Simple Application

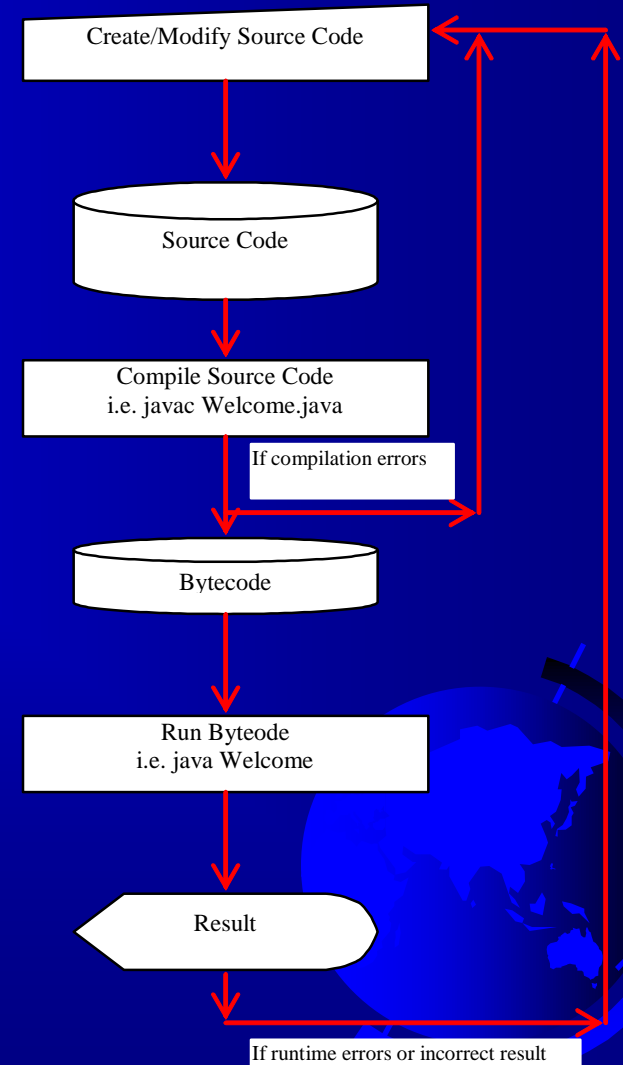
## Example 1.1

```
//This application program prints Welcome  
//to Java!  
package chapter1;  
  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```



# Creating and Compiling Programs

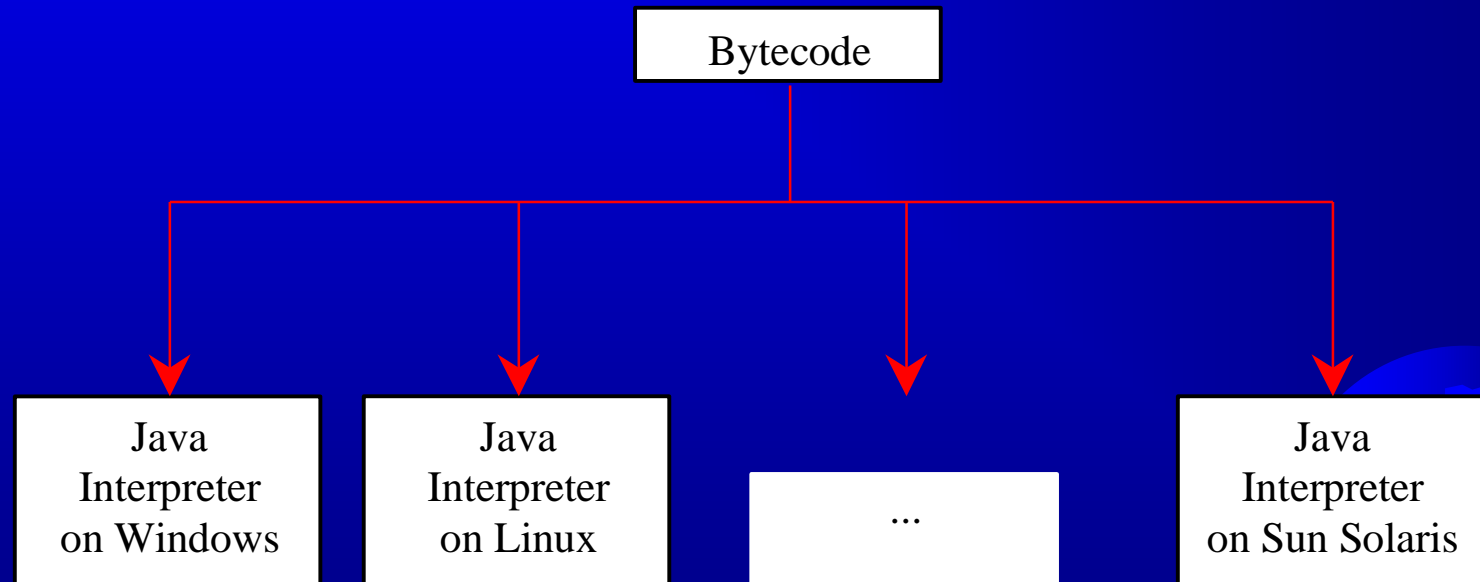
☞ On command line  
- `javac file.java`



# Executing Applications

☞ On command line

- `java classname`



# Example

```
javac Welcome.java
```

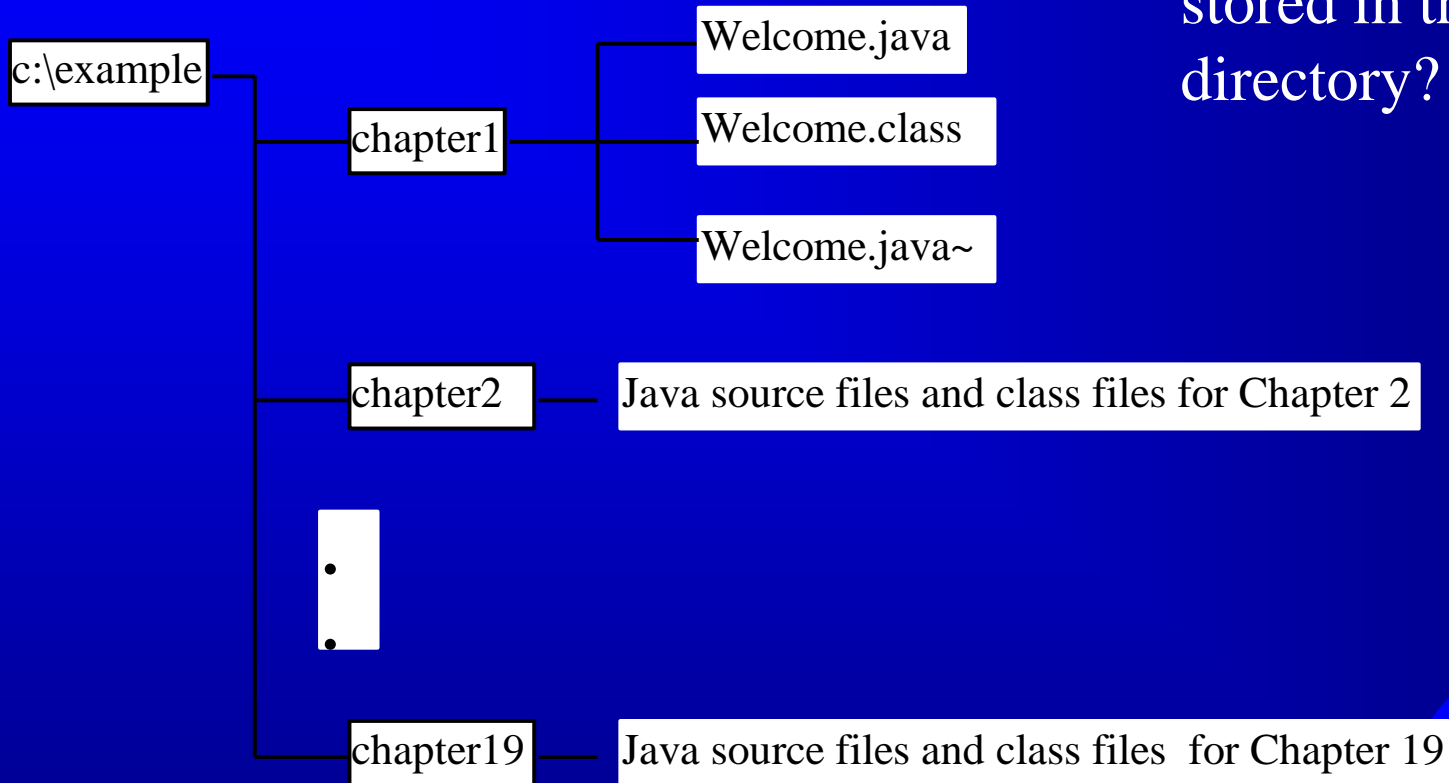
```
java Welcome
```

```
output:...
```



# Compiling and Running a Program

Where are the files stored in the directory?




# Anatomy of a Java Program

- ➡ Comments
- ➡ Package
- ➡ Reserved words
- ➡ Modifiers
- ➡ Statements
- ➡ Blocks
- ➡ Classes
- ➡ Methods
- ➡ The main method



# Comments

In Java, comments are preceded by two slashes (//) in a line, or enclosed between /\* and \*/ in one or multiple lines. When the compiler sees //, it ignores all text after // in the same line. When it sees /\*, it scans for the next \*/ and ignores any text between /\* and \*/.





# Package

The second line in the program (package chapter1;) specifies a package name, chapter1, for the class Welcome. Compiler compiles the source code in Welcome.java, generates Welcome.class, and stores Welcome.class in the chapter1 folder.



# Reserved Words


*Reserved words or keywords* are words that have a specific meaning to the compiler and cannot be used for other purposes in the program. For example, when the compiler sees the word class, it understands that the word after class is the name for the class. Other reserved words in Example 1.1 are public, static, and void. Look for Java keywords and their use.

# Modifiers

Java uses certain reserved words called *modifiers* that specify the properties of the data, methods, and classes and how they can be used. Examples of modifiers are public and static. Other modifiers are private, final, abstract, and protected. A public datum, method, or class can be accessed by other programs. A private datum or method cannot be accessed by other programs. Modifiers are discussed later.

# Statements

*A statement* represents an action or a sequence of actions. The statement `System.out.println("Welcome to Java!")` in the program in Example 1.1 is a statement to display the greeting "Welcome to Java!" Every statement in Java ends with a semicolon (;).



# Blocks

A pair of braces in a program forms a block that groups components of a program.

```
public class Test {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

Class block

Method block



# Classes

The *class* is the essential Java construct. A class is a template or blueprint for objects. To program in Java, you must understand classes and be able to write and use them. The mystery of the class will continue to be unveiled throughout this book. For now, though, understand that a program is defined by using one or more classes.



# Methods

What is System.out.println? It is a *method*: a collection of statements that performs a sequence of operations to display a message on the console. It can be used even without fully understanding the details of how it works. It is used by invoking a statement with a string argument. The string argument is enclosed within parentheses. In this case, the argument is "Welcome to Java!" You can call the same println method with a different argument to print a different message.



# main Method

The main method provides the control of program flow. The Java interpreter executes the application by invoking the main method.

The main method looks like this:

```
public static void main(String[] args) {  
    // Statements;  
}
```

It's the entry point for all java programs





# Displaying Text in a Message Dialog Box

you can use the showMessageDialog method in the JOptionPane class. JOptionPane is one of the many predefined classes in the Java system, which can be reused rather than “reinventing the wheel.”

Source

Run



# The showMessageDialog Method


```
JOptionPane.showMessageDialog(null, "Welcome to  
Java!",  
    "Example 1.2",  
    JOptionPane.INFORMATION_MESSAGE));
```



# The exit Method

Use Exit to terminate the program and stop all threads.

NOTE: When your program starts, a thread is spawned to run the program. When the showMessageDialog is invoked, a separate thread is spawned to run this method. The thread is not terminated even you close the dialog box. To terminate the thread, you have to invoke the exit method.



# END

