Chapter 1 Introduction to Computers, Programs, and Java

Objectives

- To review computer basics, programs, and operating systems
- To explore the relationship between Java and the World Wide Web
- To distinguish the terms API, IDE, and JDK
- To write a simple Java program
- To display output on the console
- To explain the basic syntax of a Java program
- To create, compile, and run Java programs
- (GUI) To display output using the JOptionPane output dialog boxes

Popular High-Level Languages

- COBOL (COmmon Business Oriented Language)
- FORTRAN (FORmula TRANslation)
- BASIC (Beginner All-purpose Symbolic Instructional Code)
- Pascal (named for Blaise Pascal)
- Ada (named for Ada Lovelace)
- C (whose developer designed B first)
- Visual Basic (Basic-like visual language developed by Microsoft)
- Delphi (Pascal-like visual language developed by Borland)
- C++ (an object-oriented language, based on C)
- C# (a Java-like language developed by Microsoft)
- Java (We use it in the book)

Why Java?

The answer is that Java enables users to develop and deploy applications on the Internet for servers, desktop computers, and small hand-held devices.

- The future of computing is being profoundly influenced by the Internet, and Java promises to remain a big part of that future. Java is the Internet programming language.

- Java is a general purpose programming language.
- Java is the Internet programming language.

Java, Web, and Beyond

Java can be used to develop

- ☞ Applications for hand-held devices such as mobile phones (next slide, Android …)
- ☞ Standalone applications across-platform on desktops and servers.
- ☞ Web applications.
  - – Java Applets (next slide, gradually out-of-date)
  - – Web Applications on the server side
    - ◆ to generate dynamic Web pages

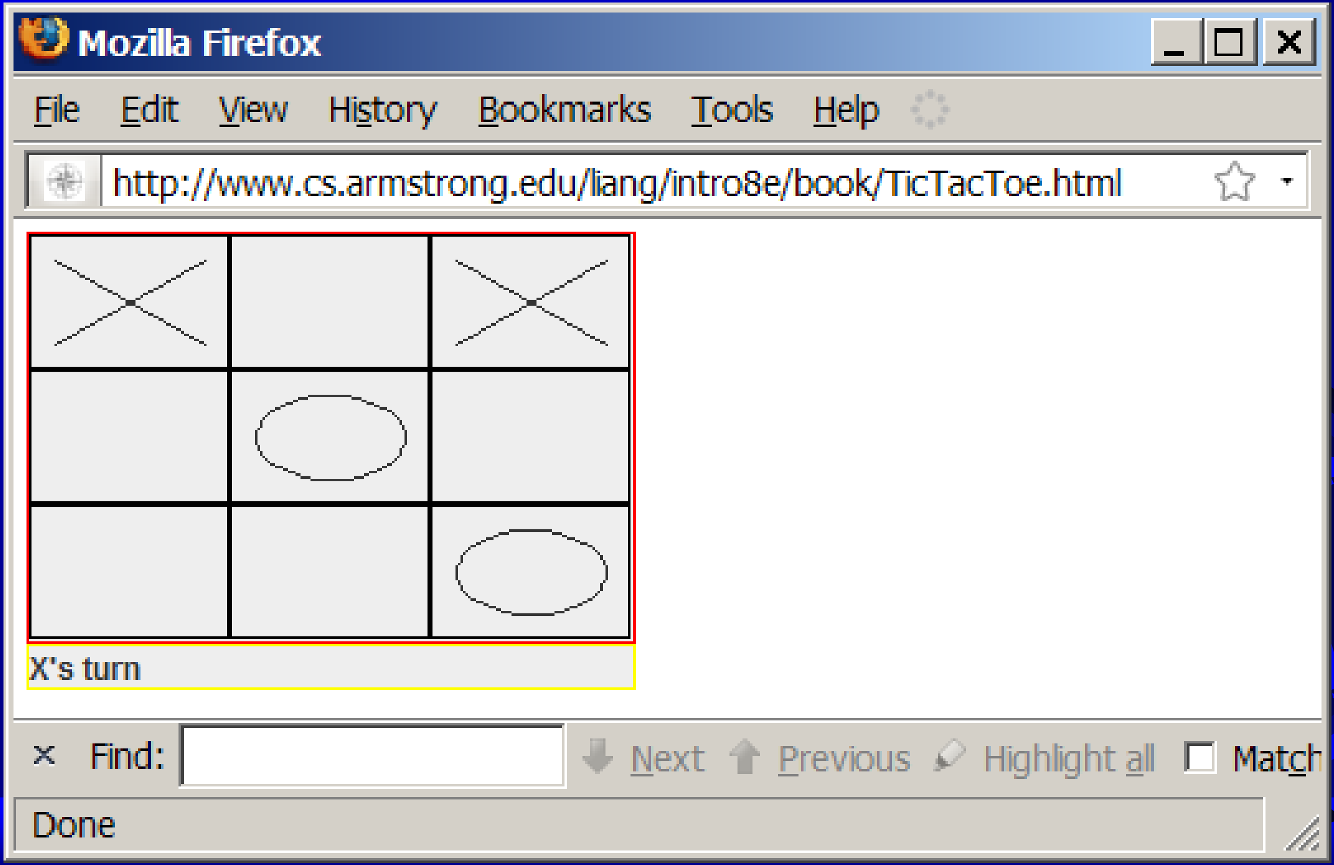
Mobile Apps

–Java-programmed calendar, games, …, on mobile phones and PDAs

Examples of Java's Versatility (Applets)

–Java Applets:

- –Java programs, run from a Web browser
- –use GUI to interact with users and process requests



X's turn

Done

Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-Neutral
- Java Is Portable
- Java's Performance
- Java Is Multithreaded
- Java Is Dynamic

Java is partially modeled on C++, but greatly simplified and improved. Some people refer to Java as "C++--" because it is like C++ but with more functionality and fewer negative aspects.

Java’s History

- Java, 1995, Sun World
  - Now， Oracle

- Early History Website:

http://java.sun.com/features/1998/05/birthday.html

JDK Versions

JDK (Java Develop Kit):

JDK 1.02 (1995)

- …
- JDK 2 (1998) a. k. a. Java 2
- …
- JDK 6 (2006) or Java 6
- …
- JDK 8 (2013) or Java 8 (popular, stable)
- …
- JDK 21 (newest ?)
  - **https://www.oracle.com/technetwork/java/javase/downloads/index.html**

JDK Editions

- **Java Standard Edition (JavaSE)**
  - before 2005, called: J2SE
  - to develop **client-side** standalone applications or applets.
  - This book uses JavaSE to introduce Java programming.
  - Java SE 21, （newest？）
- Java Enterprise Edition (JavaEE)
  - J2EE
  - to develop server-side applications such as Java servlets and Java ServerPages.
- Java Micro Edition (JavaME).
  - J2ME
  - to develop applications for mobile devices such as cell phones.

Popular Java IDEs

JDK： a set of separate programs,

each invoked from a command line, for development and testing

IDE （Integrated Development Environment）

- NetBeans Open Source by Sun
- Eclipse Open Source by IBM
- IntelliJ IDEA Open Source by JetBrains
- ……

A Simple Java Program

Listing

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

Program file: Welcome.java

animation

Trace a Program Execution

Comments

class name: by convention, start with uppercase letter

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

main method

Reserved words: keywords

Program file: Welcome.java

animation

Trace a Program Execution

Enter main method

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

animation

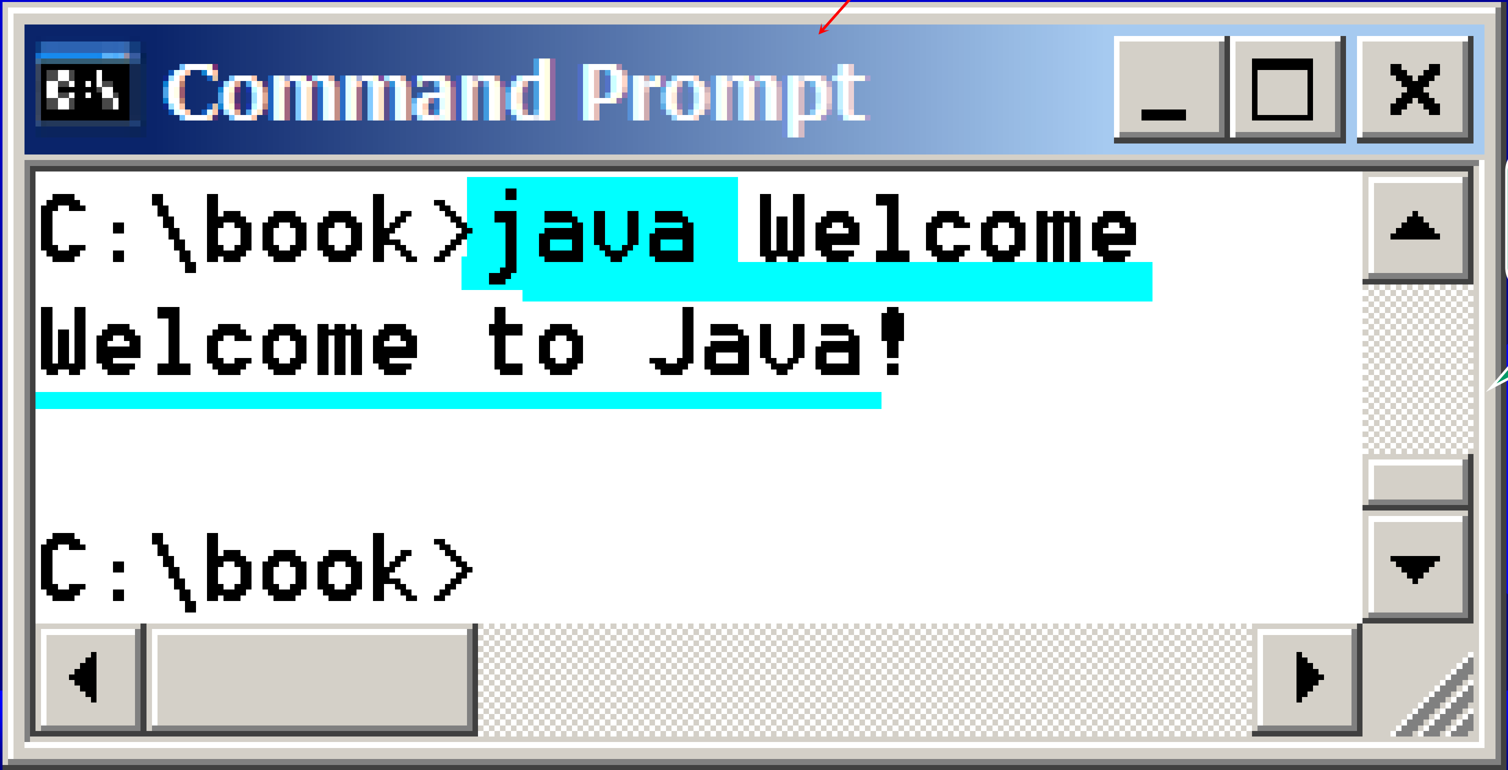
Trace a Program Execution

Execute statement

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

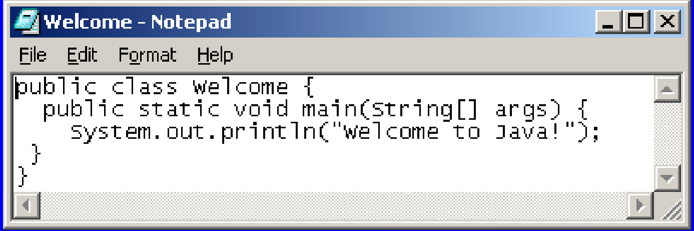
animation

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



```
C:\book>java Welcome
Welcome to Java!

C:\book>
```

print a message to the console



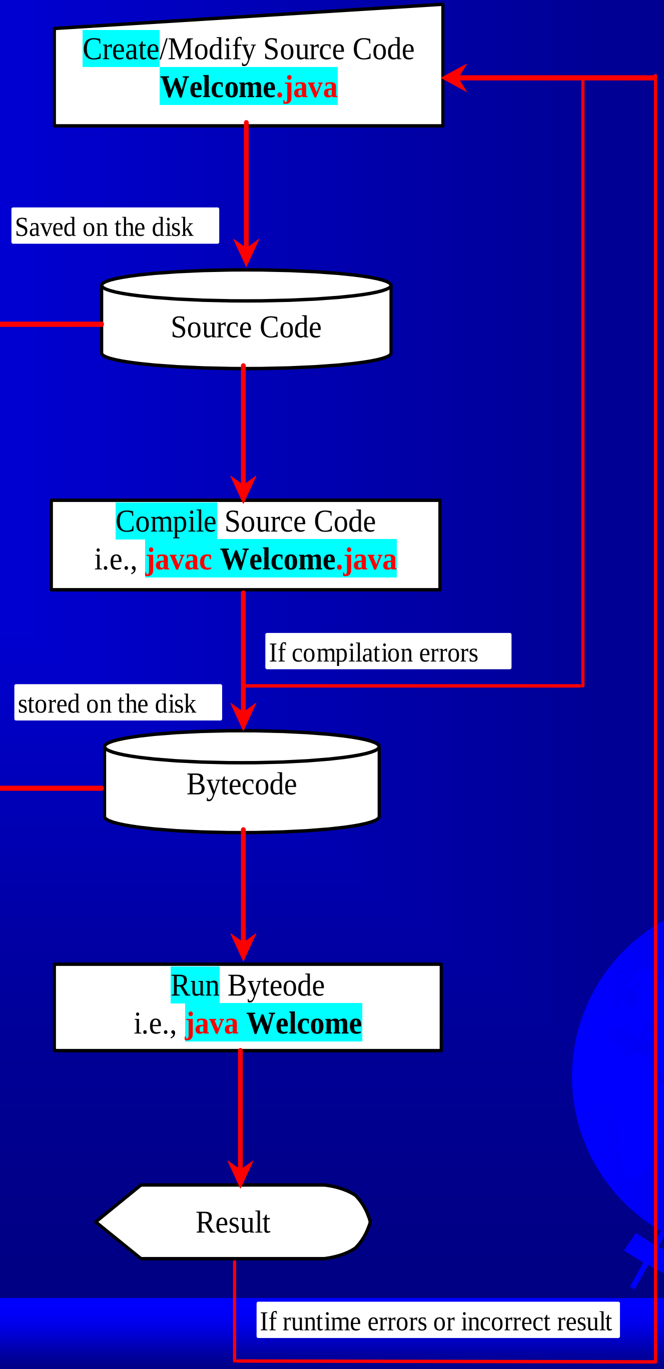
```
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Source code (developed by the programmer)

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

Byte code (generated by the compiler for JVM to read and interpret, not for you to understand)

```
…
Method Welcome()
  0 aload_0
  …

Method void main(java.lang.String[])
  0 getstatic #2 …
  3 ldc #3 <String "Welcome to
Java!">
  5 invokevirtual #4 …
```



Create

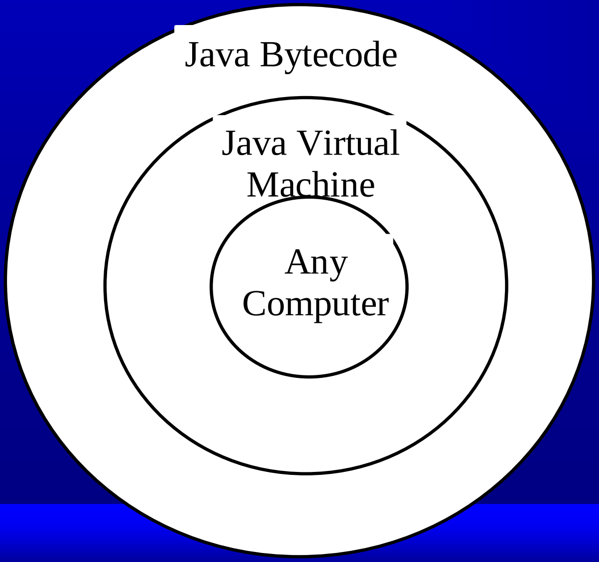
Compile

Run

Compiling Java Source Code

- You can port a **C** source program to any machine with appropriate compilers. The source program must be recompiled, however, because the object program can only run on a specific machine.
- Nowadays computers are networked to work together. Java was designed to run object programs on any platform.

With Java, you write the program once, and compile the source program into a special type of object code, known as ***bytecode***. The bytecode can then run on any computer with a **Java Virtual Machine** : a software that interprets Java bytecode to object code.



Anatomy of a Java Program

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

Comments

Three types of comments in Java.

Line comment: A line comment is preceded by two slashes (//) in a line.

Paragraph comment: A paragraph comment is enclosed between /* and */ in one or multiple lines.

javadoc comment: javadoc comments begin with /** and end with */. They are used for documenting classes, data, and methods. They can be extracted into an HTML file using JDK's javadoc command.

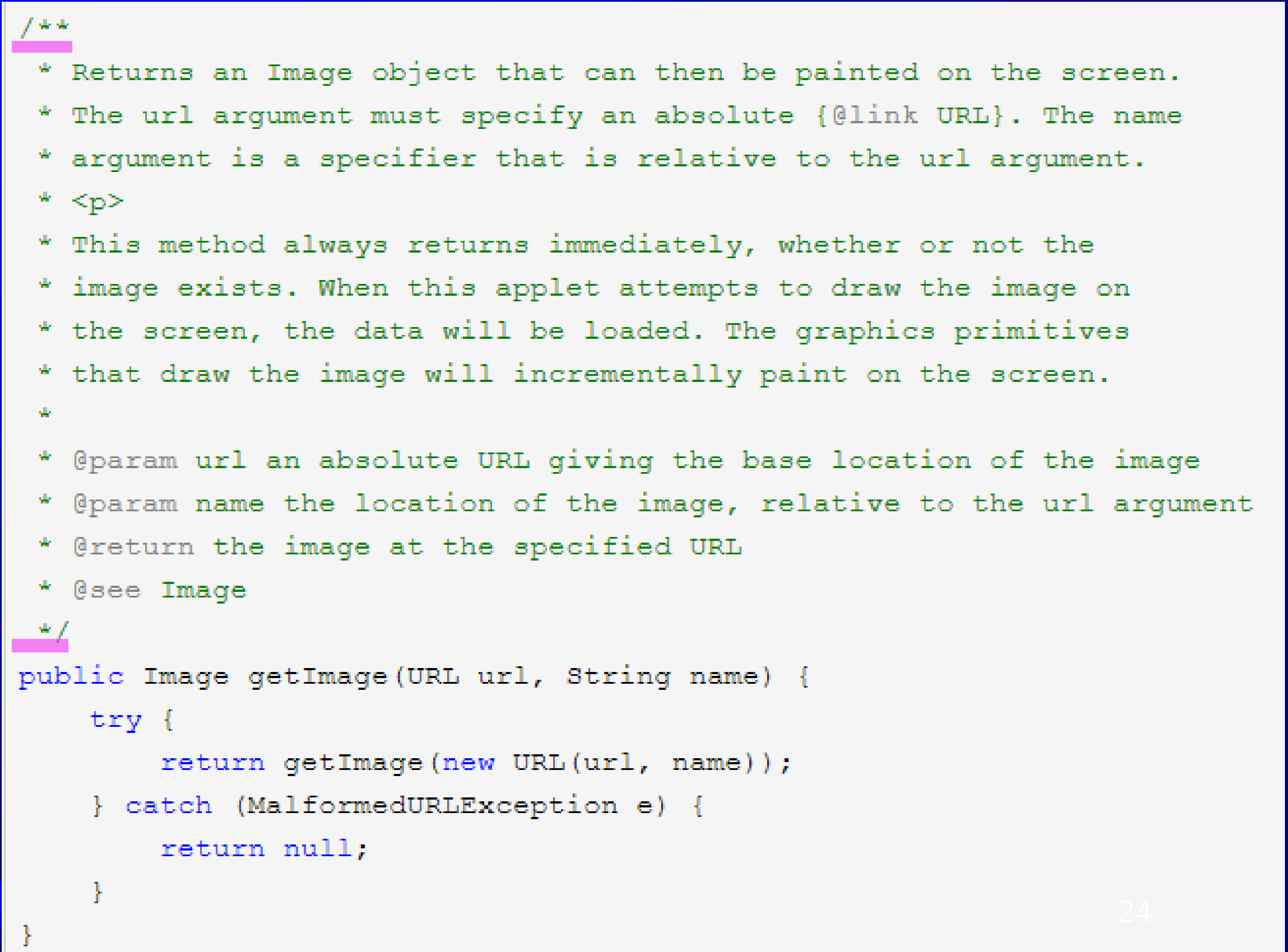
```
//  Comment in one line.

 /*
comments in multiple lines.
comments in multiple lines.
*/
```

```
//  Comment in one line.

 /*
  * comments in multiple lines.
  * comments in multiple lines.
  */
```

// Comment in one line.

/*
 * comments in multiple lines.
 * comments in multiple lines.
 */

/**
 * (description)
 *
 * (block tags)
 */
(Method code)



```
/**
 * Returns an Image object that can then be painted on the screen.
 * The url argument must specify an absolute {@link URL}. The name
 * argument is a specifier that is relative to the url argument.
 * <p>
 * This method always returns immediately, whether or not the
 * image exists. When this applet attempts to draw the image on
 * the screen, the data will be loaded. The graphics primitives
 * that draw the image will incrementally paint on the screen.
 *
 * @param url an absolute URL giving the base location of the image
 * @param name the location of the image, relative to the url argument
 * @return the image at the specified URL
 * @see Image
 */
public Image getImage(URL url, String name) {
    try {
        return getImage(new URL(url, name));
    } catch (MalformedURLException e) {
        return null;
    }
}
```

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 Listing 1.1 are public, static, and void. Their use will be introduced later in the book.

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 in Chapter 6, "Objects and Classes."

Statements

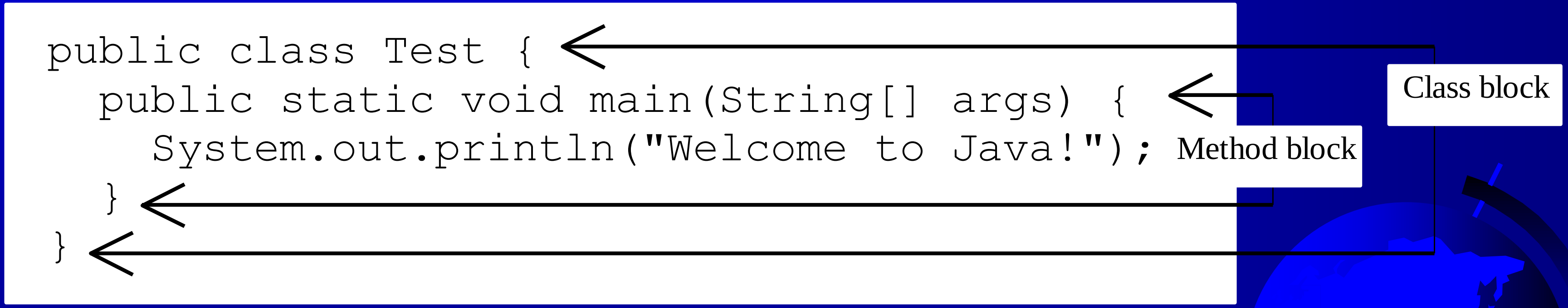
A statement represents an action or a sequence of actions.

System.out.println(“Welcome to Java!”);

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!");
  }
}
```

Classes

The class is the essential Java construct. A class is a template for objects.

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

System.out.println(…);

- Defined in Pakage: Java.lang （System Class）

  https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/lang/System.html

☞println() method: a collection of statements that performs a sequence of operations to display a message on the console.

☞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.

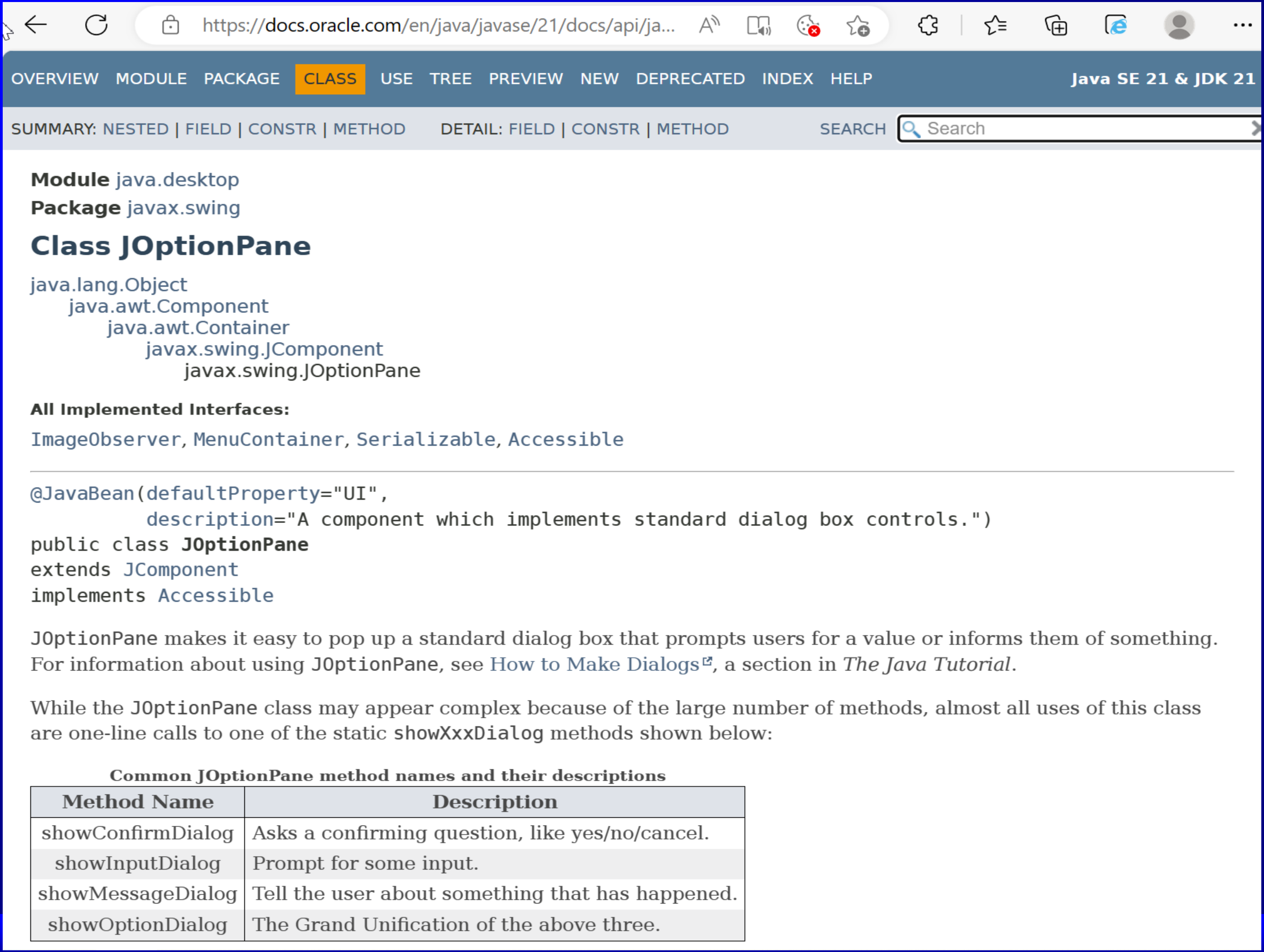
The main method looks like this:

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

Displaying Text in a Message Dialog Box

you can use the showMessageDialog method in the JOptionPane class. JOptionPane is one of the many predefined

– https://docs.oracle.com/en/java/javase/


https://docs.oracle.com/en/java/javase/21/docs/api/ja...
OVERVIEW MODULE PACKAGE CLASS USE TREE PREVIEW NEW DEPRECATED INDEX HELP
Java SE 21 & JDK 21
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
SEARCH
Search

Module java.desktop
Package javax.swing

Class JOptionPane

java.lang.Object
    java.awt.Component
        java.awt.Container
            javax.swing.JComponent
                javax.swing.JOptionPane

All Implemented Interfaces:

`ImageObserver`, `MenuContainer`, `Serializable`, `Accessible`

```
@JavaBean(defaultProperty="UI",
          description="A component which implements standard dialog box controls.")
public class JOptionPane
extends JComponent
implements Accessible
```

`JOptionPane` makes it easy to pop up a standard dialog box that prompts users for a value or informs them of something. For information about using `JOptionPane`, see How to Make Dialogs, a section in *The Java Tutorial*.

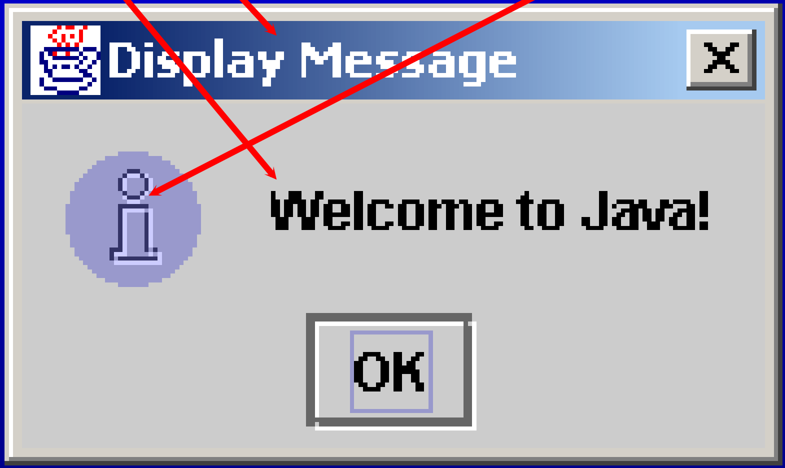
While the `JOptionPane` class may appear complex because of the large number of methods, almost all uses of this class are one-line calls to one of the static `showXxxDialog` methods shown below:

Common JOptionPane method names and their descriptions

Method Name	Description
showConfirmDialog	Asks a confirming question, like yes/no/cancel.
showInputDialog	Prompt for some input.
showMessageDialog	Tell the user about something that has happened.
showOptionDialog	The Grand Unification of the above three.

The showMessageDialog Method

```
JOptionPane.showMessageDialog(
  null,
  "Welcome to Java!",
  "Display Message",
  JOptionPane.INFORMATION_MESSAGE);
```



Two Ways to Invoke the Method

One is to use a statement as shown in the example:

```
JOptionPane.showMessageDialog(null, x,
  y, JOptionPane.INFORMATION_MESSAGE);
```

4 parameters

where x is a string for the **text** to be displayed, and y is a string for the **title** of the message dialog box.

The other is to use a statement like this:

```
JOptionPane.showMessageDialog(null, x);
```

2 parameters

where x is a string for the text to be displayed.

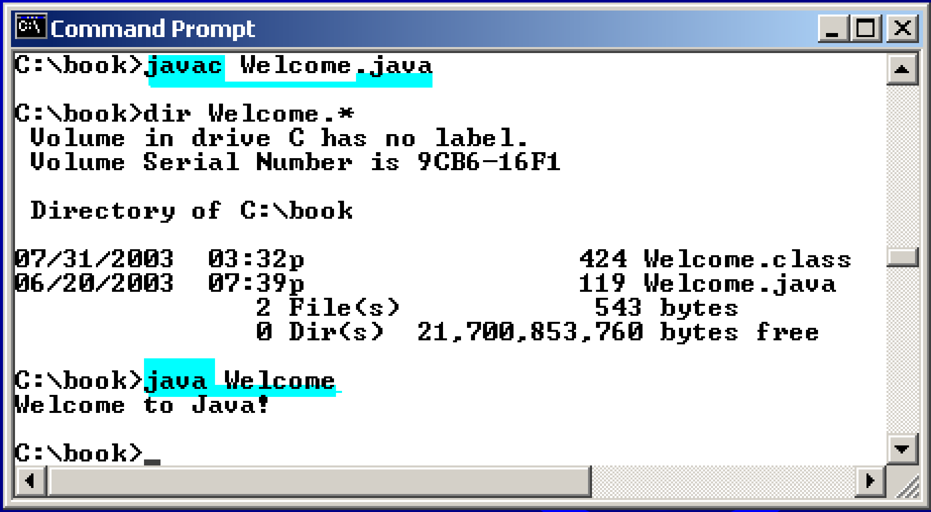
Tutorials

Movies & Readings

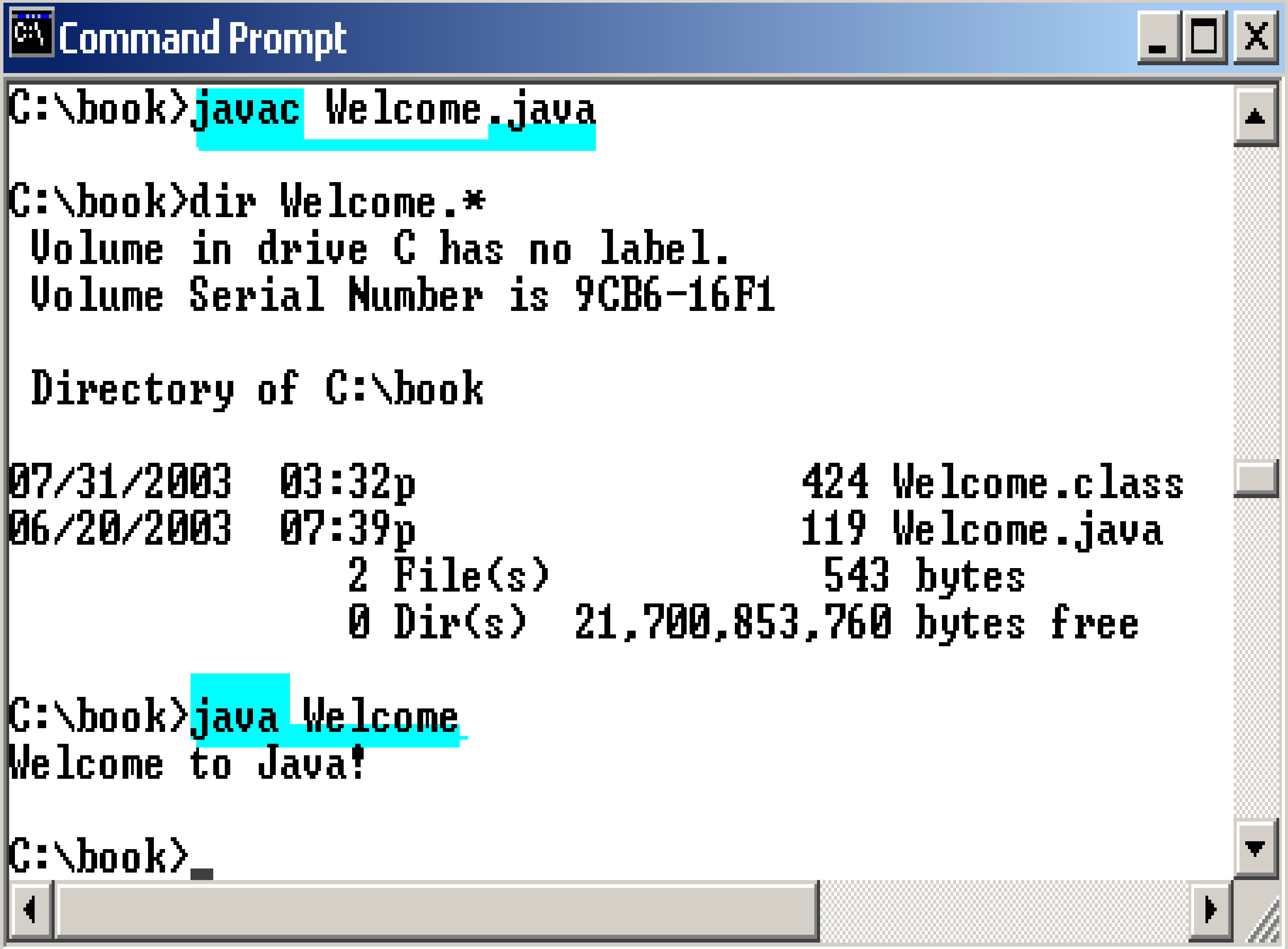
Install JDK & Eclipse

Compiling and Running Java from the Command Window

- Set path to JDK bin directory
  - set path= c:\Program Files\java\jdk1.6.0\bin
- Set classpath to include the current directory
  - set classpath=.
- Compile
  - javac Welcome.java
- Run
  - java Welcome



```
C:\book>javac Welcome.java

C:\book>dir Welcome.*
 Volume in drive C has no label.
 Volume Serial Number is 9CB6-16F1

 Directory of C:\book

07/31/2003  03:32p                 424 Welcome.class
06/20/2003  07:39p                 119 Welcome.java
               2 File(s)            543 bytes
               0 Dir(s)  21,700,853,760 bytes free

C:\book>java Welcome
Welcome to Java!

C:\book>_
```



```
C:\book>javac Welcome.java

C:\book>dir Welcome.*
 Volume in drive C has no label.
 Volume Serial Number is 9CB6-16F1

 Directory of C:\book

07/31/2003  03:32p                 424 Welcome.class
06/20/2003  07:39p                 119 Welcome.java
               2 File(s)            543 bytes
               0 Dir(s)  21,700,853,760 bytes free

C:\book>java Welcome
Welcome to Java!

C:\book>_
```

Course Report for “Java programming”

☞ <u>Chapter1 Self-Test Questions :</u>

https://media.pearsoncmg.com/ph/esm/ecs_liang_ijp_11/cw/#selftest