

Introduction to Java

NYU Tandon CS9053

Section I2 (Wednesdays)

- Dr. Constantine (Dean) Christakos
 - Email: dean@christakos.com
- TAs:
 - Akshat Kaushik ak8853@nyu.edu
 - Mugdha Rane mhr8596@nyu.edu
 - Kaustubh Dilip Yeole ky2280@nyu.edu



Class Format

- ▣ Weekly Problem Sets
 - Covers the week's material
 - 40% of your grade
 - Will drop the lowest problem set
- ▣ Take Home Midterm
 - Cumulative evaluation of what you've learned so far
 - 30% of your grade
- ▣ Final Project
 - Project of your design, with proposal reviewed by us
 - 30% of your grade



Class Format

- ▮ Slides will generally be drawn from the Liang *Introduction to Java Programming and Data Structures* text, with some variation
- ▮ I will be going through code which will be available on Brightspace in each week's section.



What You're Going to Learn

- ▣ Java Syntax
- ▣ What Object Oriented Programming is all about
- ▣ Java's libraries and datatypes
- ▣ Learning Java's strengths – GUI development and Multithreaded Programming



What You Won't Learn

- ▮ The basics of computer programming (I assume you know this already)
- ▮ Software Engineering of Large Projects
- ▮ Computer Science Algorithms
 - E.g., the sort of thing you'd get quizzed on if you were interviewing at Google
 - Though some problem sets will be exercises in algorithmic implementation



Chapter 1 Introduction to Computers, Programs, and Java



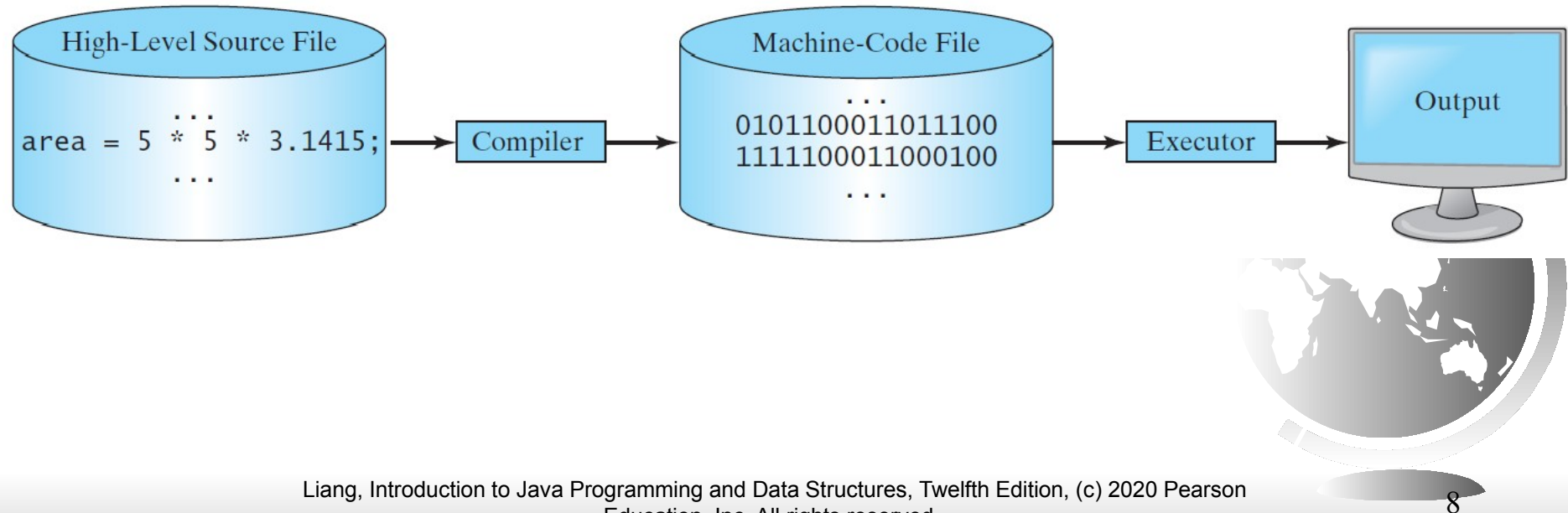
Objectives

- ▮ To understand computer basics, programs, and operating systems (§§1.2–1.4).
- ▮ To describe the relationship between Java and the World Wide Web (§1.5).
- ▮ To understand the meaning of Java language specification, API, JDK, and IDE (§1.6).
- ▮ To write a simple Java program (§1.7).
- ▮ To display output on the console (§1.7).
- ▮ To explain the basic syntax of a Java program (§1.7).
- ▮ To create, compile, and run Java programs (§1.8).
- ▮ To use sound Java programming style and document programs properly (§1.9).
- ▮ To explain the differences between syntax errors, runtime errors, and logic errors (§1.10).
- ▮ To develop Java programs using NetBeans (§1.11).
- ▮ To develop Java programs using Eclipse (§1.12).



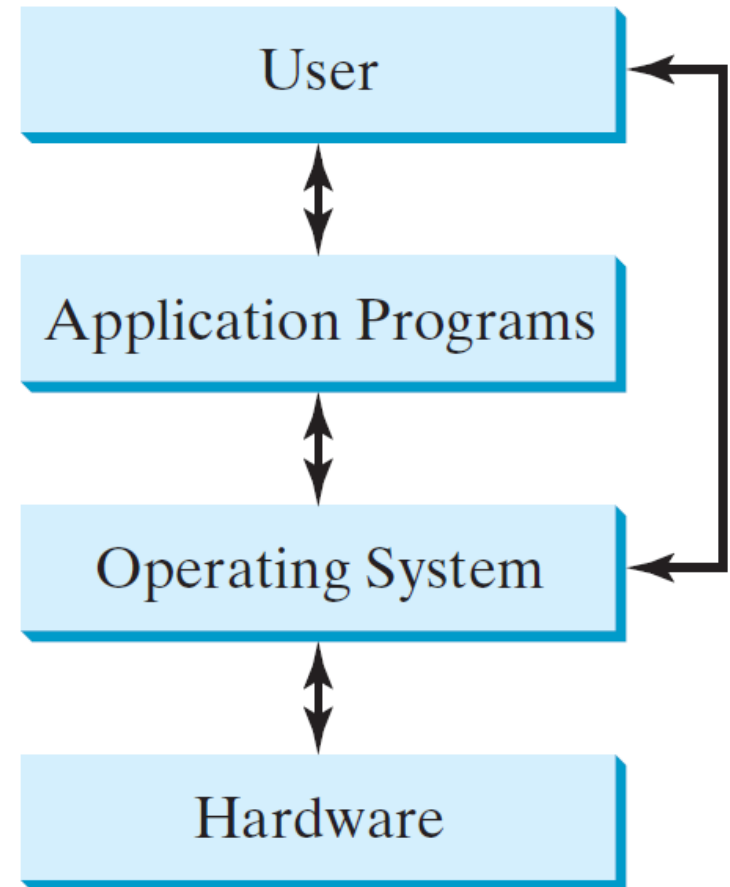
Compiling Source Code

A compiler translates the entire source code into a machine-code file, and the machine-code file is then executed, as shown in the following figure.



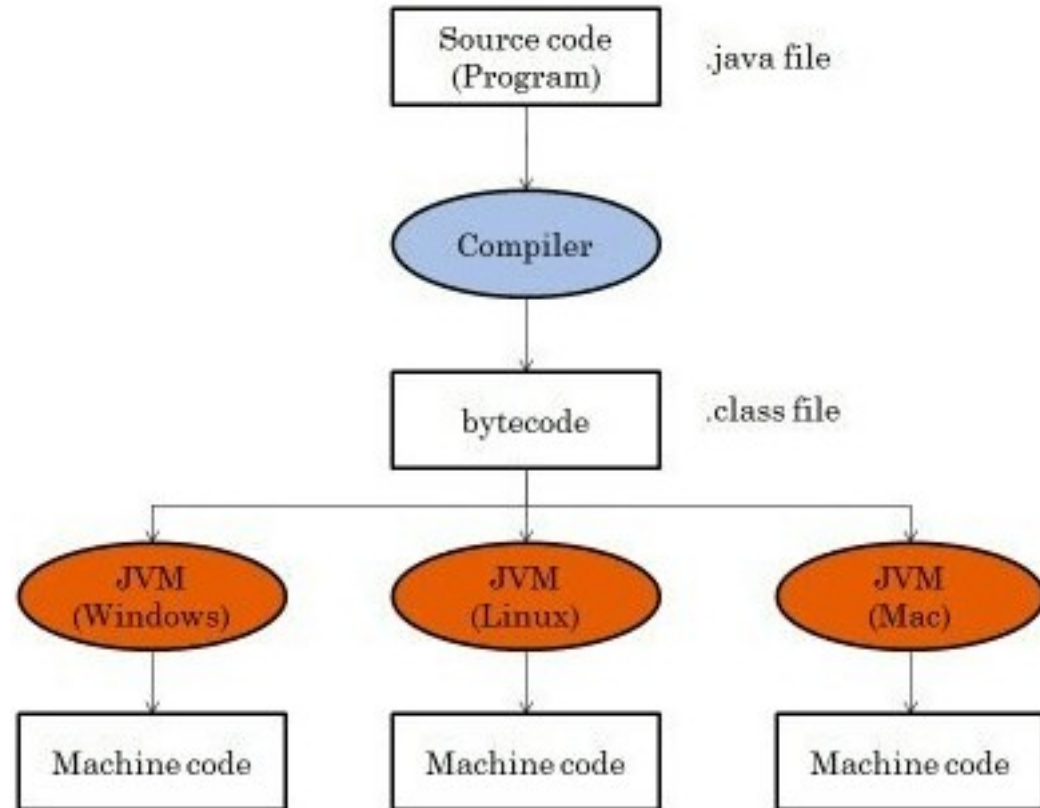
Operating Systems

The *operating system* (OS) is a program that manages and controls a computer's activities. The popular operating systems for general-purpose computers are Microsoft Windows, Mac OS, and Linux. Application programs, such as a Web browser or a word processor, cannot run unless an operating system is installed and running on the computer.



Where Java Fits In

- Java programs aren't compiled into Machine Code, they're compiled into Bytecode, and the Java Virtual Machine for each platform translates Bytecode into Machine Code when it loads a class
- That means (ideally) you can write Java code once, compile the bytecode and run the bytecode anywhere



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 standalone applications.
- ▣ Java can be used to develop applications running from a browser.
- ▣ Java can also be used to develop applications for hand-held devices.
- ▣ Java can be used to develop applications for Web servers.



Java's History

- James Gosling and Sun Microsystems
- Oak
- Java, May 20, 1995, Sun World
- HotJava
 - The first Java-enabled Web browser
- Early History Summary:

<https://codegym.cc/groups/posts/594-history-of-java-a-full-story-of-java-development-from-1991-to-2021>



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



<https://liveexample.pearsoncmg.com/etc/JavaCharacteristics.pdf>

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.



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 inherently object-oriented.

Although many object-oriented languages began strictly as procedural languages, Java was designed from the start to be object-oriented. Object-oriented programming (OOP) is a popular programming approach that is replacing traditional procedural programming techniques.

One of the central issues in software development is how to reuse code. Object-oriented programming provides great flexibility, modularity, clarity, and reusability through encapsulation, inheritance, and polymorphism.



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

Distributed computing involves several computers working together on a network. Java is designed to make distributed computing easy. Since networking capability is inherently integrated into Java, writing network programs is like sending and receiving data to and from a file.



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

You need an interpreter to run Java programs. The programs are compiled into the Java Virtual Machine code called bytecode. The bytecode is machine-independent and can run on any machine that has a Java interpreter, which is part of the Java Virtual Machine (JVM).



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 compilers can detect many problems that would first show up at execution time in other languages.

Java has eliminated certain types of error-prone programming constructs found in other languages.

Java has a runtime exception-handling feature to provide programming support for robustness.



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 implements several security mechanisms to protect your system against harm caused by stray programs.

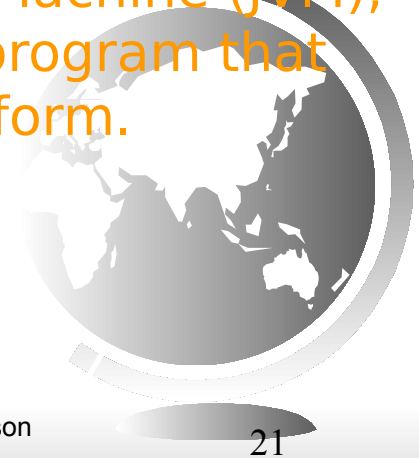


Characteristics of Java

- 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

Write once, run anywhere

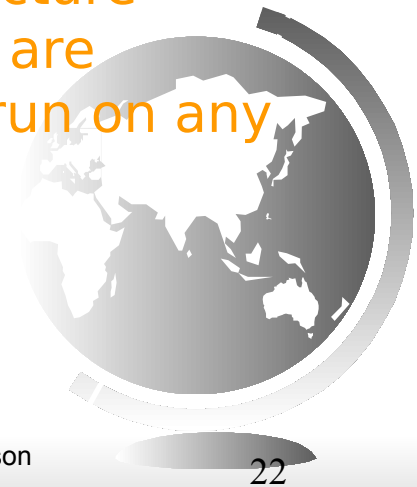
With a Java Virtual Machine (JVM),
you can write one program that
will run on any platform.



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

Because Java is architecture neutral, Java programs are portable. They can be run on any platform without being recompiled.



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

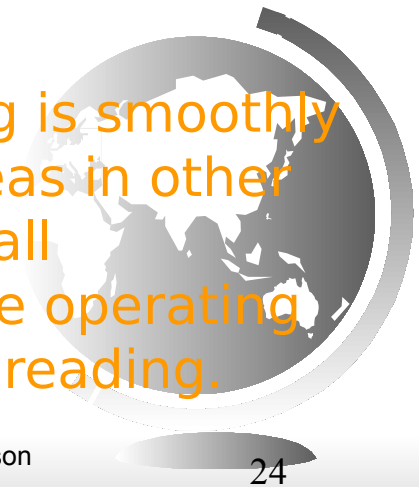
Java's performance has been improved impressively over the years with every new version.



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

Multithread programming is smoothly integrated in Java, whereas in other languages you have to call procedures specific to the operating system to enable multithreading.



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 was designed to adapt to an evolving environment. New code can be loaded on the fly without recompilation. There is no need for developers to create, and for users to install, major new software versions. New features can be incorporated transparently as needed.



JDK Versions

- JDK 1.02 (1995)
- JDK 1.1 (1996)
- JDK 1.2 (1998)
- JDK 1.3 (2000)
- JDK 1.4 (2002)
- JDK 1.5 (2004) a. k. a. JDK 5 or Java 5
- JDK 1.6 (2006) a. k. a. JDK 6 or Java 6
- JDK 1.7 (2011) a. k. a. JDK 7 or Java 7
- JDK 1.8 (2014) a. k. a. JDK 8 or Java 8
- Java 9, 10, 11, 12, 13, 14, 15, 16, 17.



Why Java 8?

- The latest version of Java is Java 17 as of Sept. 2021
- Java 8 was a *significant* release with many important syntactical features that were not available in Java 1.6 and 1.7.
 - forEach loops
 - Lambda expressions
 - The Stream API
- Java 8 is a “LTS” (Long Term Support) version, and commercial use of Java starting with Java 11 requires that organizations pay for Java, so organizations stick with Java 8 rather than paying for subsequent LTS versions like Java 11 and Java 17



JDK Editions

- ▣ Java Standard Edition (J2SE)
 - J2SE can be used to develop client-side standalone applications or applets.
- ▣ Java Enterprise Edition (J2EE)
 - J2EE can be used to develop server-side applications such as Java servlets, Java ServerPages, and Java ServerFaces.
- ▣ Java Micro Edition (J2ME).
 - J2ME can be used to develop applications for mobile devices such as cell phones.

This book uses J2SE to introduce Java programming.



Popular Java IDEs

- ▣ NetBeans
- ▣ Eclipse
- ▣ I will use Eclipse for most in-class demonstrations, and Eclipse makes submission easiest for the TAs. However, there is no required IDE for this class, and you are free to use the IDE of your choice.



A Simple Java Program

Listing 1.1

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

Welcome

Note: Clicking the green button displays the source code with interactive animation. You can also run the code in a browser. Internet connection is needed for this button.

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!");  
    }  
}
```



Trace a Program Execution

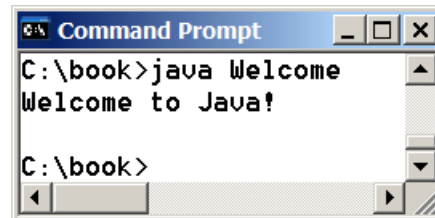
Execute statement

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



Trace a Program Execution

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



print a message to the console

Two More Simple Examples

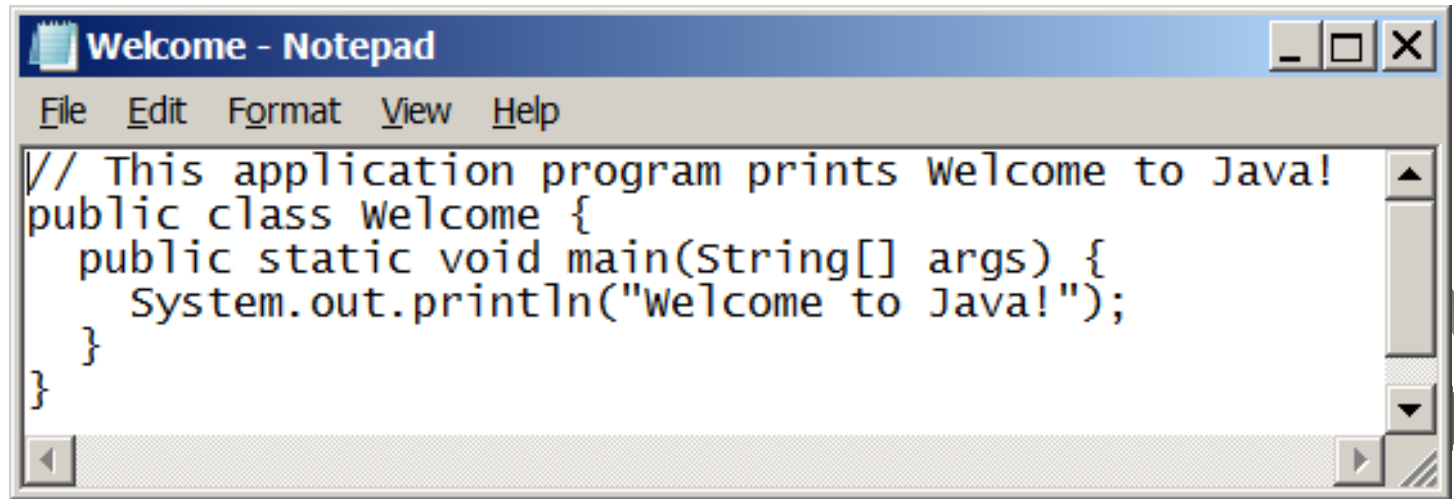
WelcomeWithThreeMessages

ComputeExpression



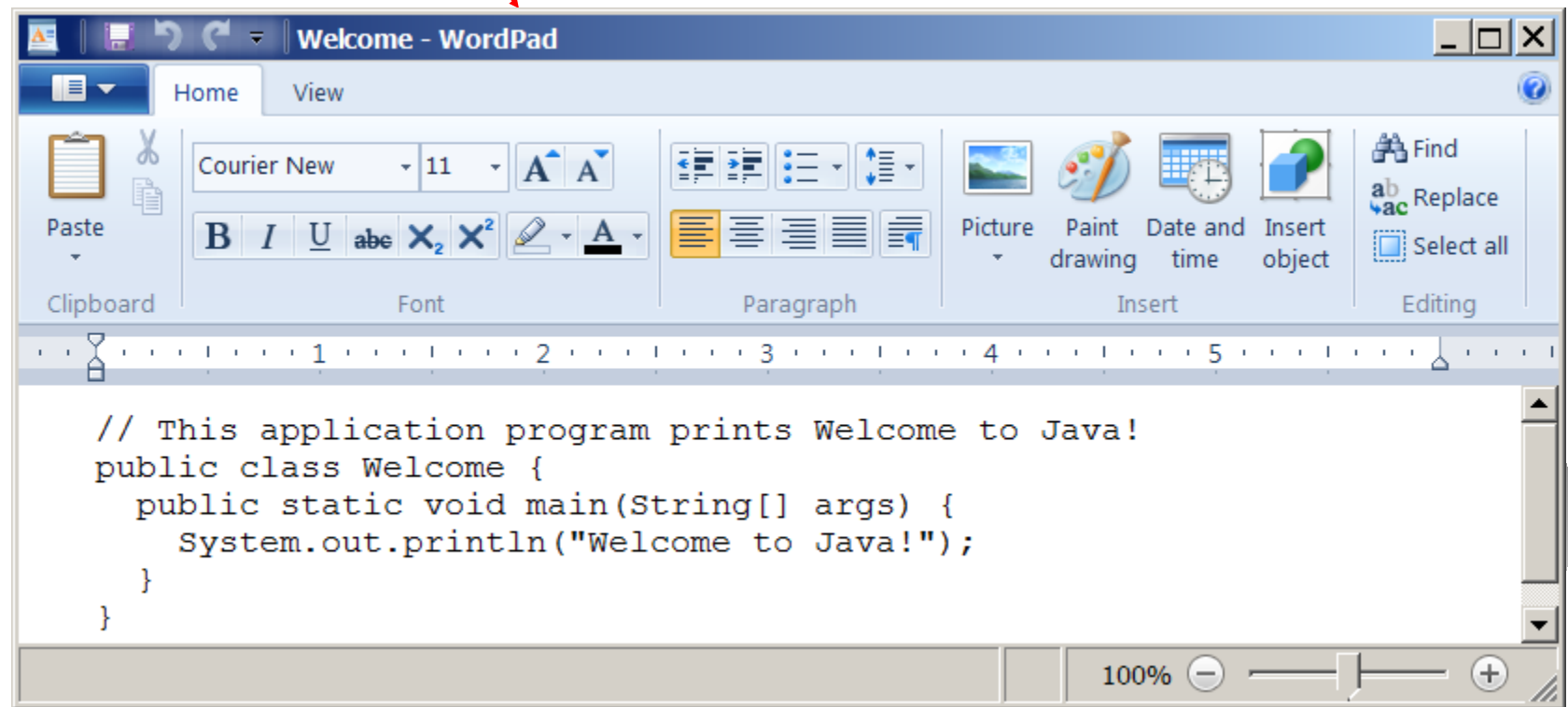
Creating and Editing Using NotePad

To use NotePad, type
notepad
Welcome.java
from the DOS prompt.



Creating and Editing Using WordPad

To use WordPad, type
write Welcome.java
from the DOS prompt.



```
File Edit Format View Help
// This application program prints welcome to Java!
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!");
    }
}
```

Bytecode (generated by the compiler for JVM to read and interpret)

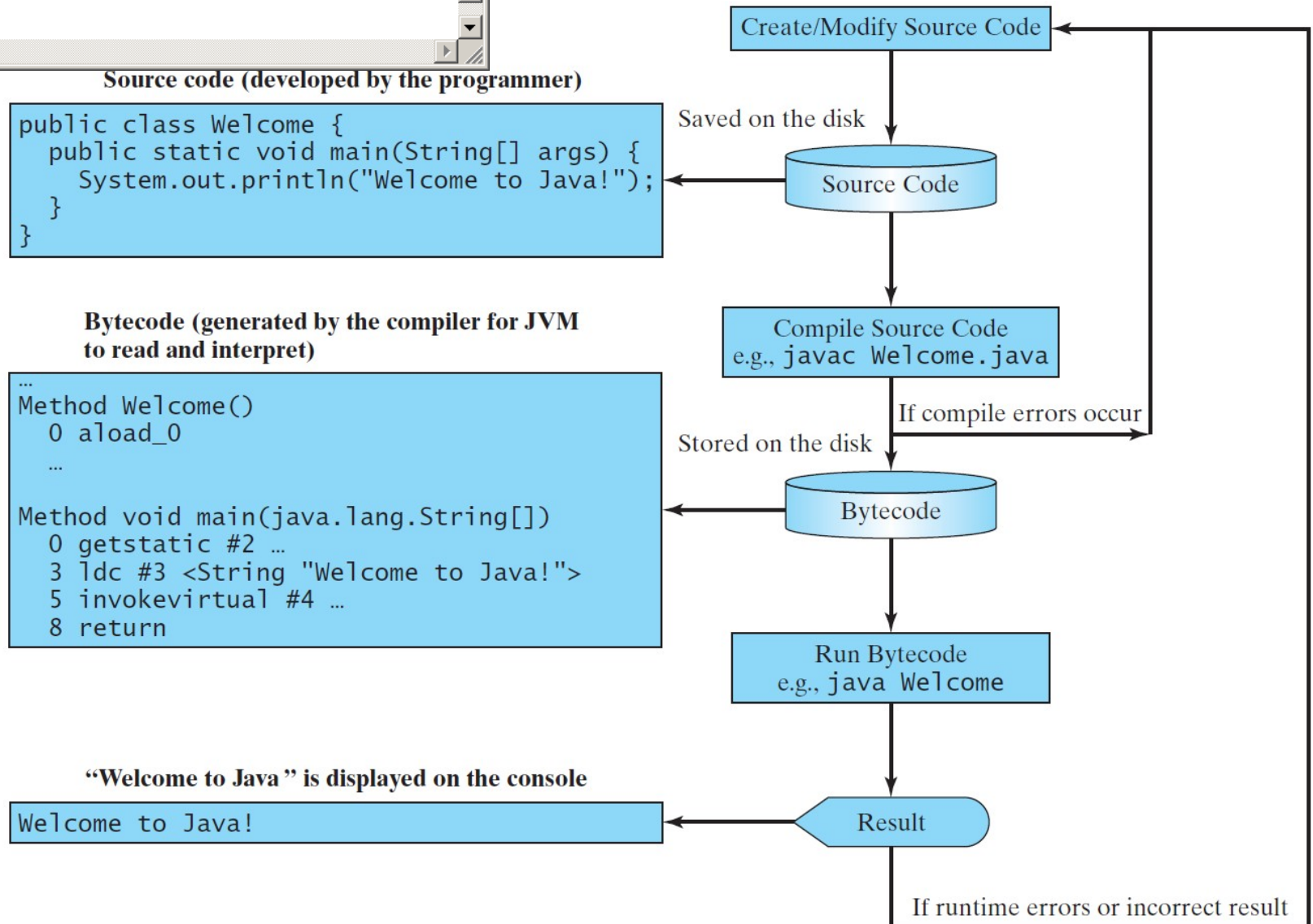
```
...
Method Welcome()
  0 aload_0
  ...

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

“Welcome to Java” is displayed on the console

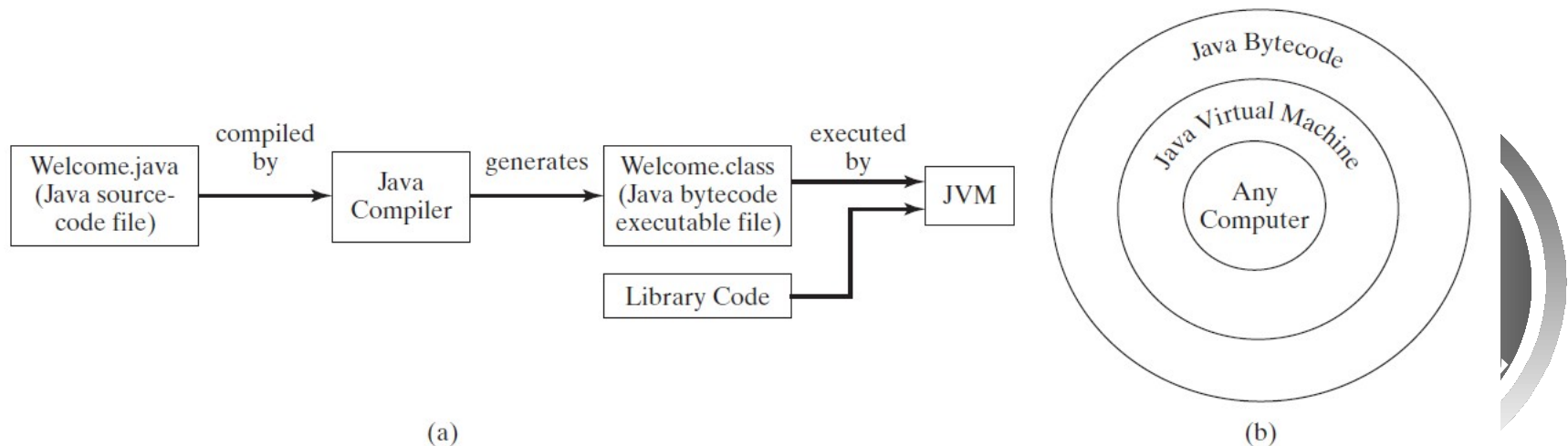
Welcome to Java!

Creating, Compiling, and Running Programs



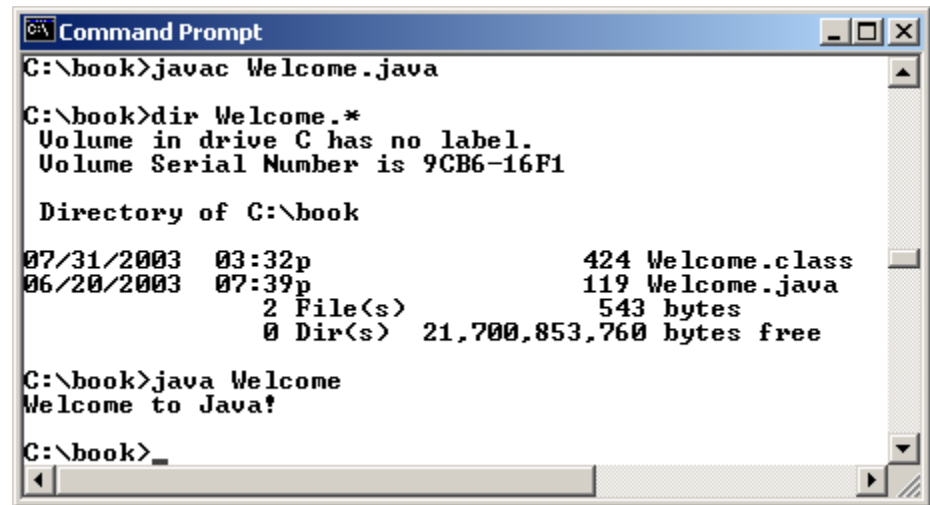
Compiling Java Source Code

You can port a 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, as shown below. Java Virtual Machine is a software that interprets Java bytecode.



Compiling and Running Java from the Command Window

- Set path to JDK bin directory
 - set path=c:\Program Files\java\jdk1.8.0\bin
- Set classpath to include the current directory
 - set classpath=.
- Compile
 - javac Welcome.java
- Run
 - java Welcome



```
Command Prompt
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

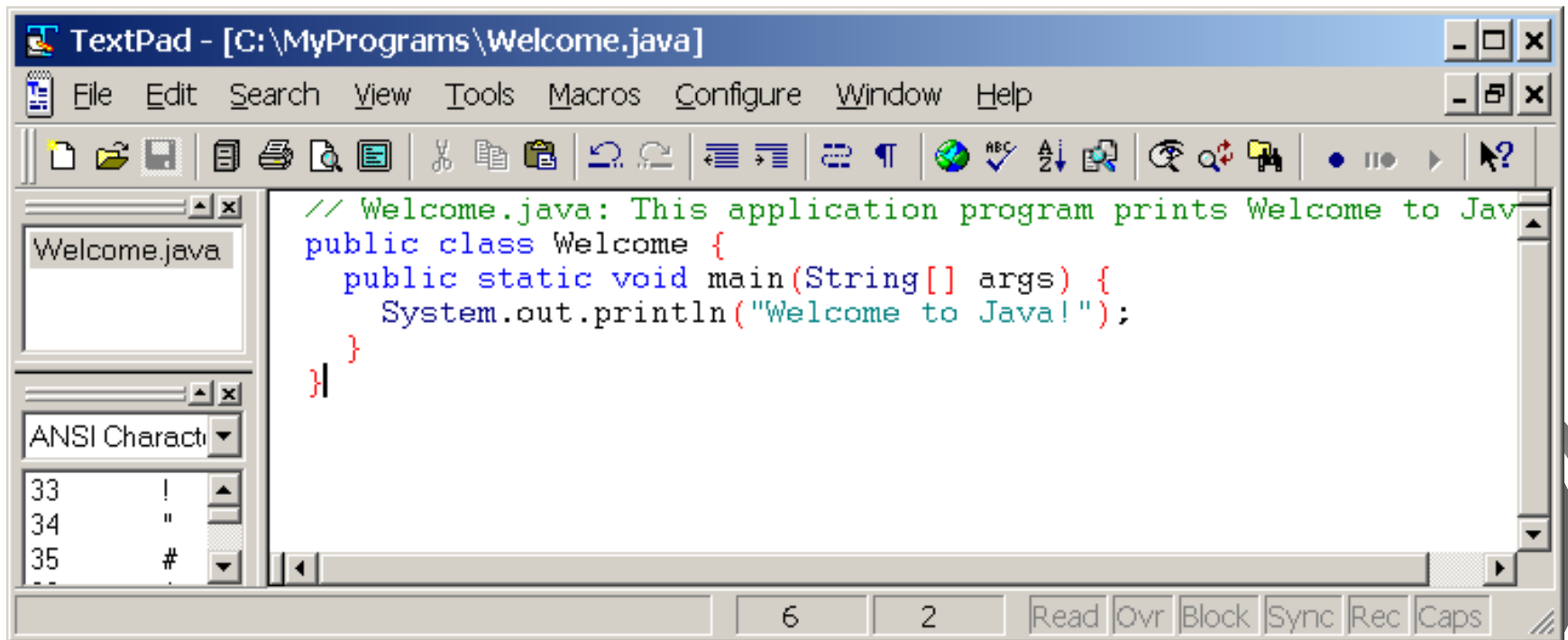
07/31/2003  03:32p                424 Welcome.class
06/20/2003  07:39p                119 Welcome.java
                2 File(s)              543 bytes
                0 Dir(s)  21,700,853,760 bytes free

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

Compiling and Running Java from TextPad

Companion
Website

- See Supplement II.A on the Website for details



Anatomy of a Java Program


- Class name
- Main method
- Statements
- Statement terminator
- Reserved words
- Comments
- Blocks



Class Name

Every Java program must have at least one class. Each class has a name. By convention, class names start with an uppercase letter. In this example, the class name is Welcome.

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



Main Method

Line 2 defines the main method. In order to run a class, the class must contain a method named main. The program is executed from the main method.

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

Statement


A statement represents an action or a sequence of actions. The statement `System.out.println("Welcome to Java!")` in the program in Listing 1.1 is a statement to display the greeting "Welcome to Java!".

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

Statement Terminator

Every statement in Java ends with a semicolon (;).


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



Keywords

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.

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



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



Special Symbols

| Character Name | Description | |
|----------------|-------------------------------------|--|
| { } | Opening and closing braces | Denotes a block to enclose statements. |
| () | Opening and closing parentheses | Used with methods. |
| [] | Opening and closing brackets | Denotes an array. |
| // | Double slashes | Precedes a comment line. |
| " " | Opening and closing quotation marks | Enclosing a string (i.e., sequence of characters). |
| ; | Semicolon | Marks the end of a statement. |




{ ... }

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

(...)

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



•
;

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

// ...

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

..

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

Programming Style and Documentation

- Appropriate Comments
- Naming Conventions
- Proper Indentation and Spacing Lines
- Block Styles



Appropriate Comments

Include a summary at the beginning of the program to explain what the program does, its key features, its supporting data structures, and any unique techniques it uses.

Include your name, class section, instructor, date, and a brief description at the beginning of the program.



Naming Conventions

- Choose meaningful and descriptive names.
- Class names:
 - Capitalize the first letter of each word in the name. For example, the class name `ComputeExpression`.



Proper Indentation and Spacing

□ Indentation

- Indent two spaces.

□ Spacing

- Use blank line to separate segments of the code.



Block Styles

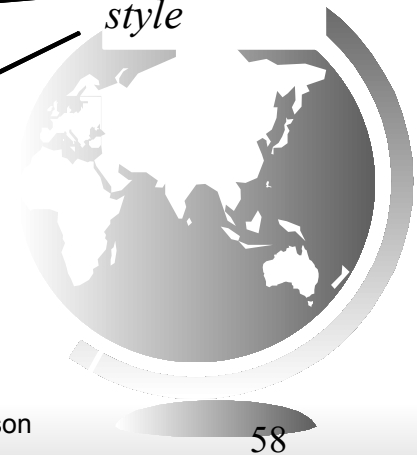
Use end-of-line style for braces.

*Next-line
style*

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("Block Styles");
    }
}
```

*End-of-line
style*

```
public class Test {
    public static void main(String[] args) {
        System.out.println("Block Styles");
    }
}
```



Programming Errors

- ▣ Syntax Errors
 - Detected by the compiler
- ▣ Runtime Errors
 - Causes the program to abort
- ▣ Logic Errors
 - Produces incorrect result



Syntax Errors

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



ShowSyntaxErrors

Runtime Errors

```
public class ShowRuntimeErrors {  
    public static void main(String[] args) {  
        System.out.println(1 / 0);  
    }  
}
```

ShowRuntimeErrors



Logic Errors

```
public class ShowLogicErrors {  
    public static void main(String[] args) {  
        System.out.println("Celsius 35 is Fahrenheit degree ");  
        System.out.println((9 / 5) * 35 + 32);  
    }  
}
```



ShowLogicErrors



Trace a Program Execution

```
public class ComputeArea {  
    /** Main method */  
    public static void main(String[] args) {  
        double radius;  
        double area;  
  
        // Assign a radius  
        radius = 20;  
  
        // Compute area  
        area = radius * radius * 3.14159;  
  
        // Display results  
        System.out.println("The area for the circle of radius " +  
            radius + " is " + area);  
    }  
}
```

radius

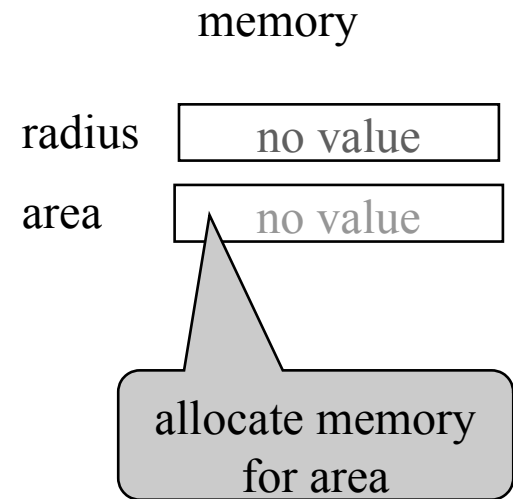
allocate memory
for radius

no value



Trace a Program Execution

```
public class ComputeArea {  
    /** Main method */  
    public static void main(String[] args) {  
        double radius;  
        double area;  
  
        // Assign a radius  
        radius = 20;  
  
        // Compute area  
        area = radius * radius * 3.14159;  
  
        // Display results  
        System.out.println("The area for the circle of radius " +  
            radius + " is " + area);  
    }  
}
```



Trace a Program Execution

```
public class ComputeArea {  
    /** Main method */  
    public static void main(String[] args) {  
        double radius;  
        double area;
```

```
        // Assign a radius
```

```
        radius = 20;
```

```
        // Compute area
```

```
        area = radius * radius * 3.14159;
```

```
        // Display results
```

```
        System.out.println("The area for the circle of radius " +  
            radius + " is " + area);
```

```
    }  
}
```

assign 20 to radius

20

radius

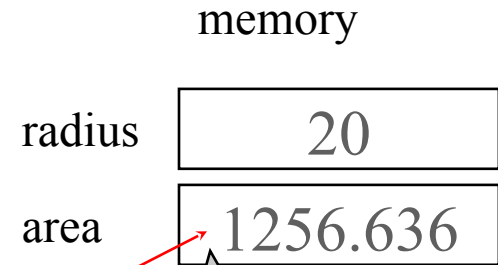
area

no value



Trace a Program Execution

```
public class ComputeArea {  
    /** Main method */  
    public static void main(String[] args) {  
        double radius;  
        double area;  
  
        // Assign a radius  
        radius = 20;  
  
        // Compute area  
        area = radius * radius * 3.14159;  
  
        // Display results  
        System.out.println("The area for the circle of radius " +  
            radius + " is " + area);  
    }  
}
```



compute area and assign it to variable area



Trace a Program Execution

```
public class ComputeArea {  
    /** Main method */  
    public static void main(String[] args) {  
        double radius;  
        double area;  
  
        // Assign a radius  
        radius = 20;  
  
        // Compute area  
        area = radius * radius * 3.14159;  
  
        // Display results  
        System.out.println("The area for the circle of radius "  
            + radius + " is " + area);  
    }  
}
```

memory

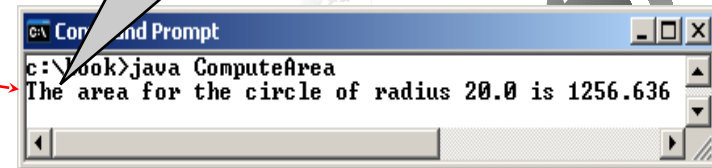
radius

20

area

1256.636

print a message to the
console



Reading Input from the Console

1. Create a Scanner object

```
Scanner input = new Scanner(System.in);
```

2. Use the method `nextDouble()` to obtain to a double value. For example,

```
System.out.print("Enter a double value: ");  
Scanner input = new Scanner(System.in);  
double d = input.nextDouble();
```

ComputeAreaWithConsoleInput

ComputeAverage



Implicit Import and Explicit Import

```
java.util.* ; // Implicit import
```

```
java.util.Scanner; // Explicit Import
```

No performance difference



For next week, Assignment 1

□ Intro assignment

- Most of your time will probably be spent getting Eclipse working correctly and familiarizing yourself with it
 - ♦ But that part is not graded!
- I promise most assignments won't be several pages long, as this one is
- Due 11:59pm, Feb. 2nd

