

CEN 419

Introduction to Java Programming



Dr. H. Esin ÜNAL
FALL
2021

Slides are modified from original slides of Y. Daniel Liang

Programs

- Computer *programs*, known as *software*, are instructions to the computer.

- ***You tell a computer what to do through programs.***

Without programs, a computer is an empty machine.

- Computers do not understand human languages, so you need to use computer languages to communicate with them.
- Programs are written using ***programming languages***.

Programming Languages

Machine Language

Assembly Language

High-Level Language

Machine language is a set of primitive instructions built into every computer. The instructions are in the form of binary code, so you have to enter binary codes for various instructions. Program with native machine language is a tedious process. Moreover the programs are highly difficult to read and modify.

For example, to add two numbers, you might write an instruction in binary like this:

1 1 0 1 1 0 1 0 1 0 0 1 1 0 1 0

Programming Languages

Machine Language

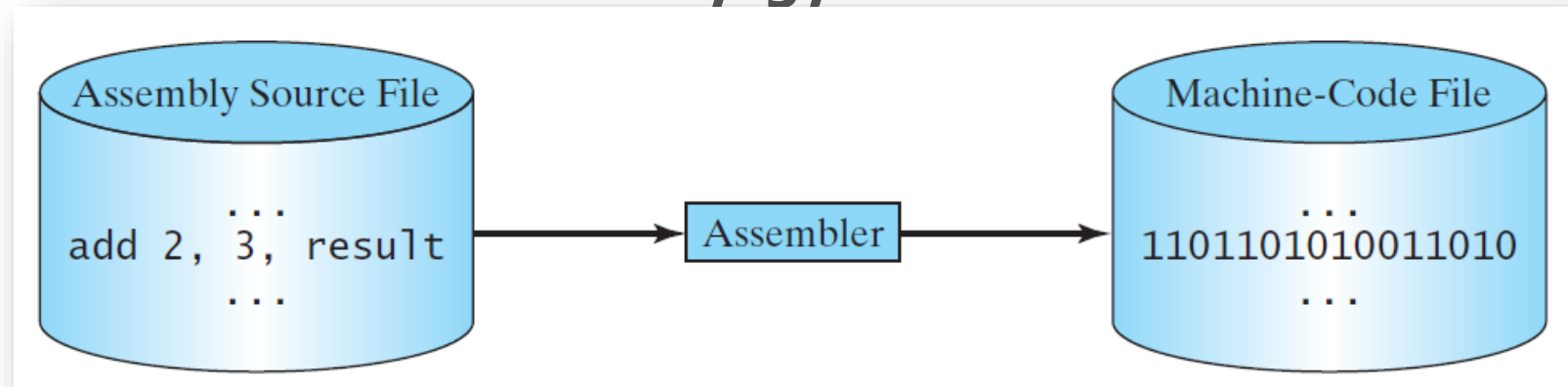
Assembly Language

High-Level Language

Assembly languages were developed to make programming easy. Since the computer cannot execute assembly language, a program called assembler is used to convert assembly language programs into machine code.

For example, to add 2 and 3 and get the result, you might write an instruction in assembly code like this:

add 2, 3, result



Programming Languages

Machine Language Assembly Language

High-Level Language
























The *high-level languages* are English-like and easy to learn and use. They are platform independent. The instructions in a high-level programming language are called statements.

For example, the following is a high-level language statement that computes the area of a circle with radius 5:

```
area = 5 * 5 * 3.1415;
```

Programming Languages

Language Ranking: IEEE Spectrum

Rank	Language	Type	Score
1	Python	  	100.0
2	Java	  	95.4
3	C	  	94.7
4	C++	  	92.4
5	JavaScript		88.1
6	C#	   	82.4
7	R		81.7
8	Go	 	77.7
9	HTML		75.4
10	Swift	 	70.4

Language Types

Web



Enterprise



Mobile



Embedded



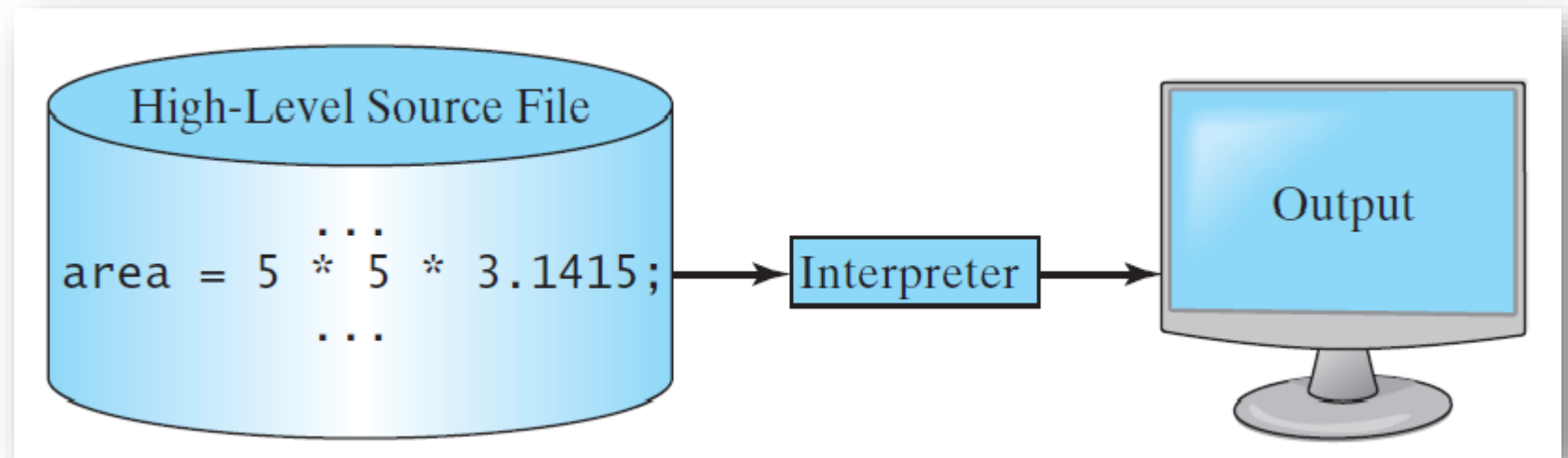
Source: <https://spectrum.ieee.org/top-programming-languages/>

Interpreting or Compiling Source Code

- A program written in a high-level language is called a *source program* or *source code*.
- Because a computer cannot understand a source program, a source program must be translated into machine code for execution.
- The translation can be done using another programming tool called an *interpreter* or a *compiler*.

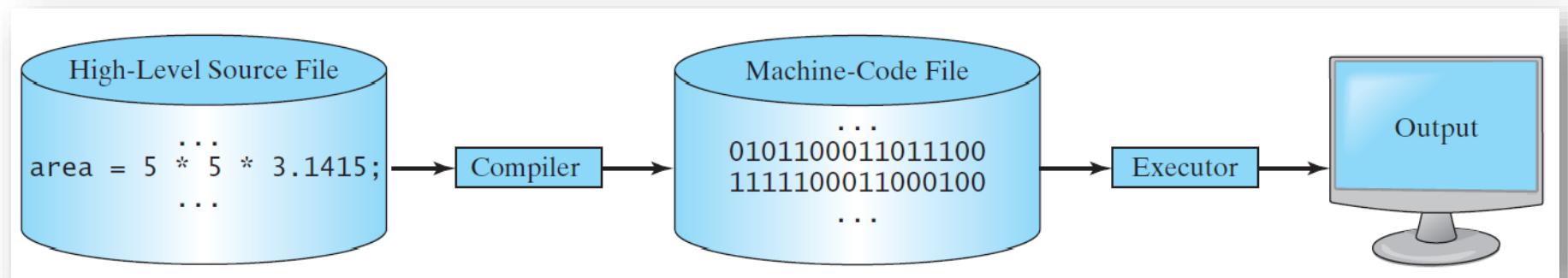
Interpreting Source Code

- An *interpreter* reads one statement from the source code, translates it to the machine code or virtual machine code, and then executes it right away, as shown in the figure.
- Note that a statement from the source code may be translated into several machine instructions.



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.



Why Java?

- The answer is that Java enables users to **develop and deploy applications on the Internet for servers, desktop computers, and mobile 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 high-level programming language.

Characteristics of Java

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

Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust (Reliable)
- 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.**

Java uses automatic memory allocation and garbage collection, whereas C++ requires the programmer to allocate memory and collect garbage.

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 (Reliable)
- 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.

A Java program is object-oriented because programming in Java is centered on creating objects, manipulating objects, and making objects work together.

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 (Reliable)
- 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 (Reliable)
- 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).

With Java, you compile the source code once, and the bytecode generated by a Java compiler can run on any platform with a Java interpreter. The Java interpreter translates the bytecode into the machine language of the target machine.

Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- **Java Is Robust (Reliable)**
- 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 (ie. does not support pointers, thereby eliminating the possibility of overwriting memory and corrupting data.)

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 (Reliable)
- **Java Is Secure**
- Java Is Architecture-Neutral
- Java Is Portable
- Java's Performance
- Java Is Multithreaded
- Java Is Dynamic

As an Internet programming language, Java is used in a networked and distributed environment.

If you download a Java applet and run it on your computer, it will not damage your system because Java implements several security mechanisms to protect your system against harm caused by stray programs.

The security is based on the premise that **nothing should be trusted.**

Characteristics of Java

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

Java is **interpreted**. This feature enables Java to be architecture-neutral, or to use an alternative term, **platform-independent**.

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 (Reliable)
- 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.**

The Java environment is portable to new hardware and operating systems. In fact, the Java compiler itself is written in Java.

Characteristics of Java

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

Java's **performance is sometimes criticized**. The execution of the bytecode is never as fast as it would be with a compiled language, such as C++.

Because Java is interpreted, the bytecode is not directly executed by the system, but is run through the interpreter.

Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust (Reliable)
- 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.

Multithreading is particularly useful in graphical user interface (GUI) and network programming.

Multithreading is a necessity in multimedia and network programming.

Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust (Reliable)
- 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.

History of Java

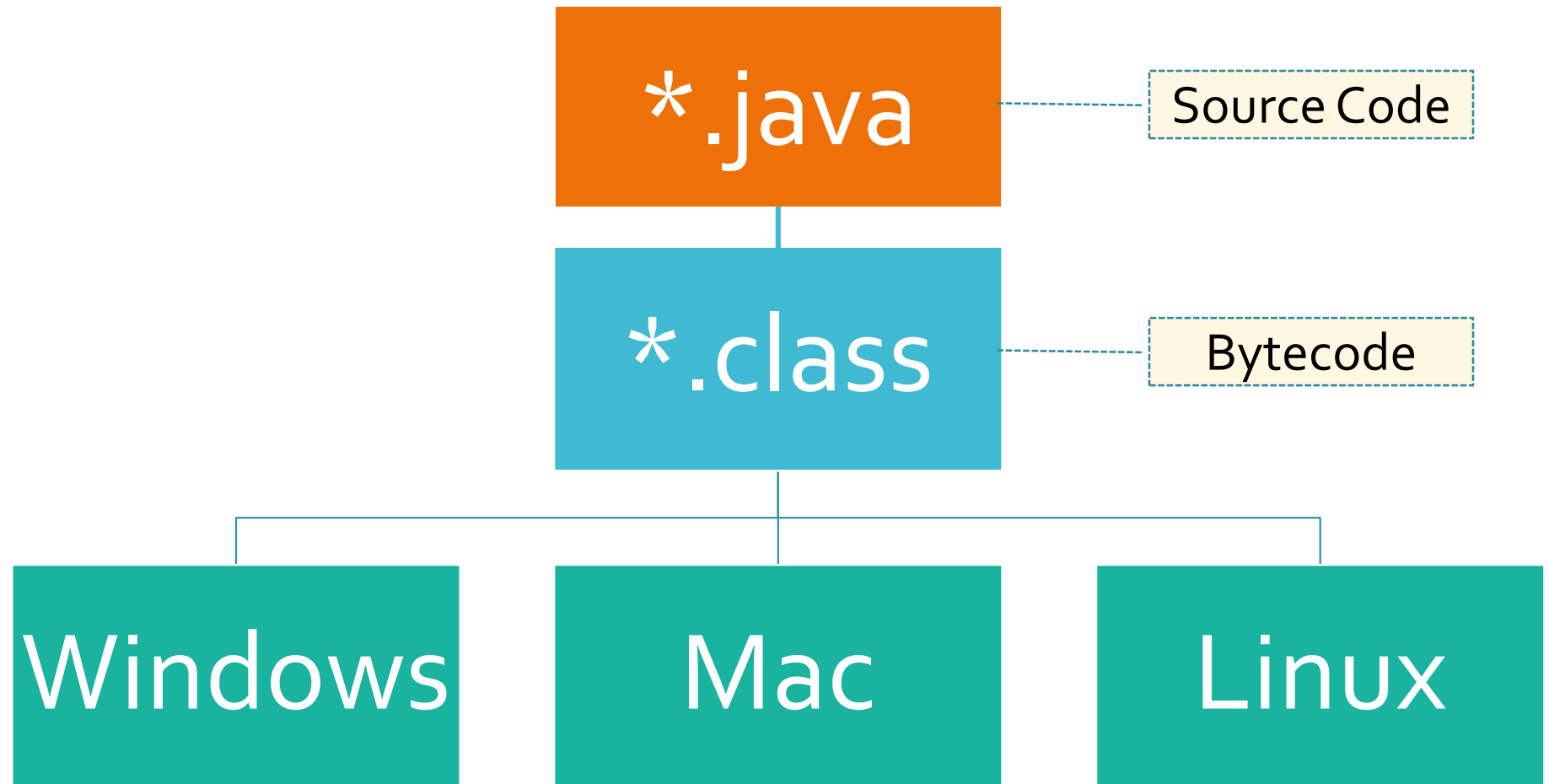
- Java was developed by James Gosling and his team at Sun Microsystems in California.
- The language was first called Oak, named after the oak tree outside of Gosling's office, but the name was already taken, so the team renamed it to Java (May 20, 1995).
- The language was based on C and C++. Java inherits its syntax from C and its object model is adapted from C++.
- C# is influenced from Java. Many of C# features are directly parallel to Java.
- Early History Website:

<http://www.java.com/en/javahistory/index.jsp>

Java, Web, and Beyond

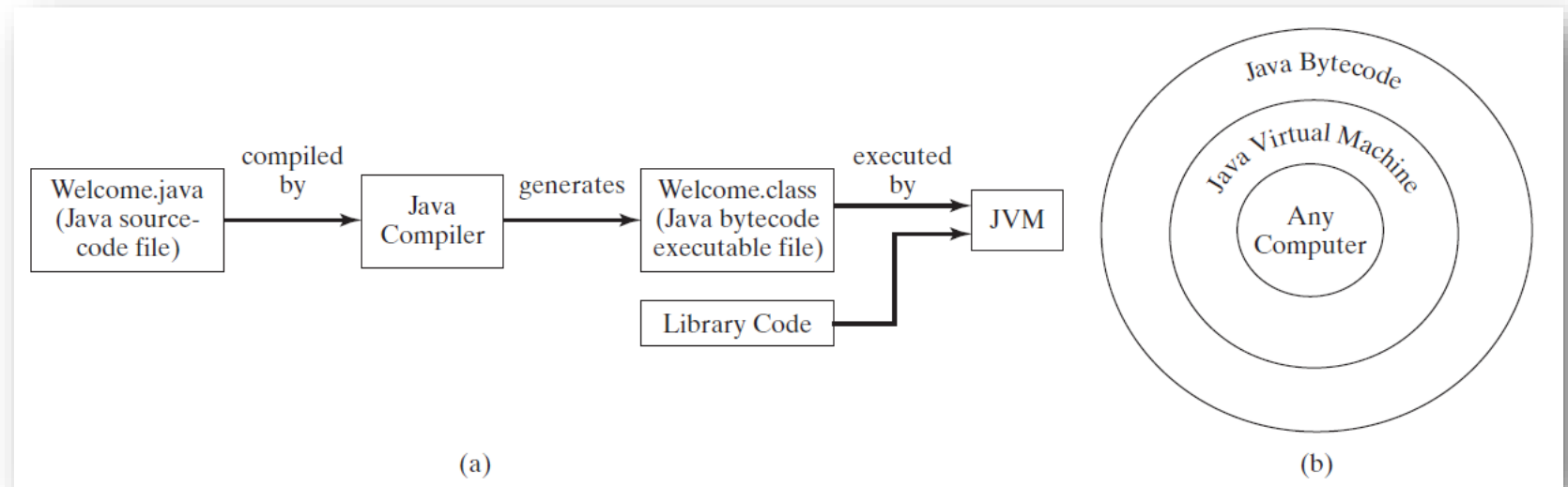
- Java can be used to develop *standalone applications* (Robotic rover on Mars).
- Java can be used to develop applications *running from a browser* (Applets).
- Java can also be used to develop applications for *hand-held devices* (Android cell phones).
- Java can be used to develop applications for *Web servers* (Many commercial Websites).

Architecture of Java



Compiling Java Source Code

- 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.



JDK

- *Java Development Kit – JDK* provides the environment to *develop and execute (run)* the Java program.
- JDK is only used by **Java Developers**.
- It contains **JRE + development tools**.

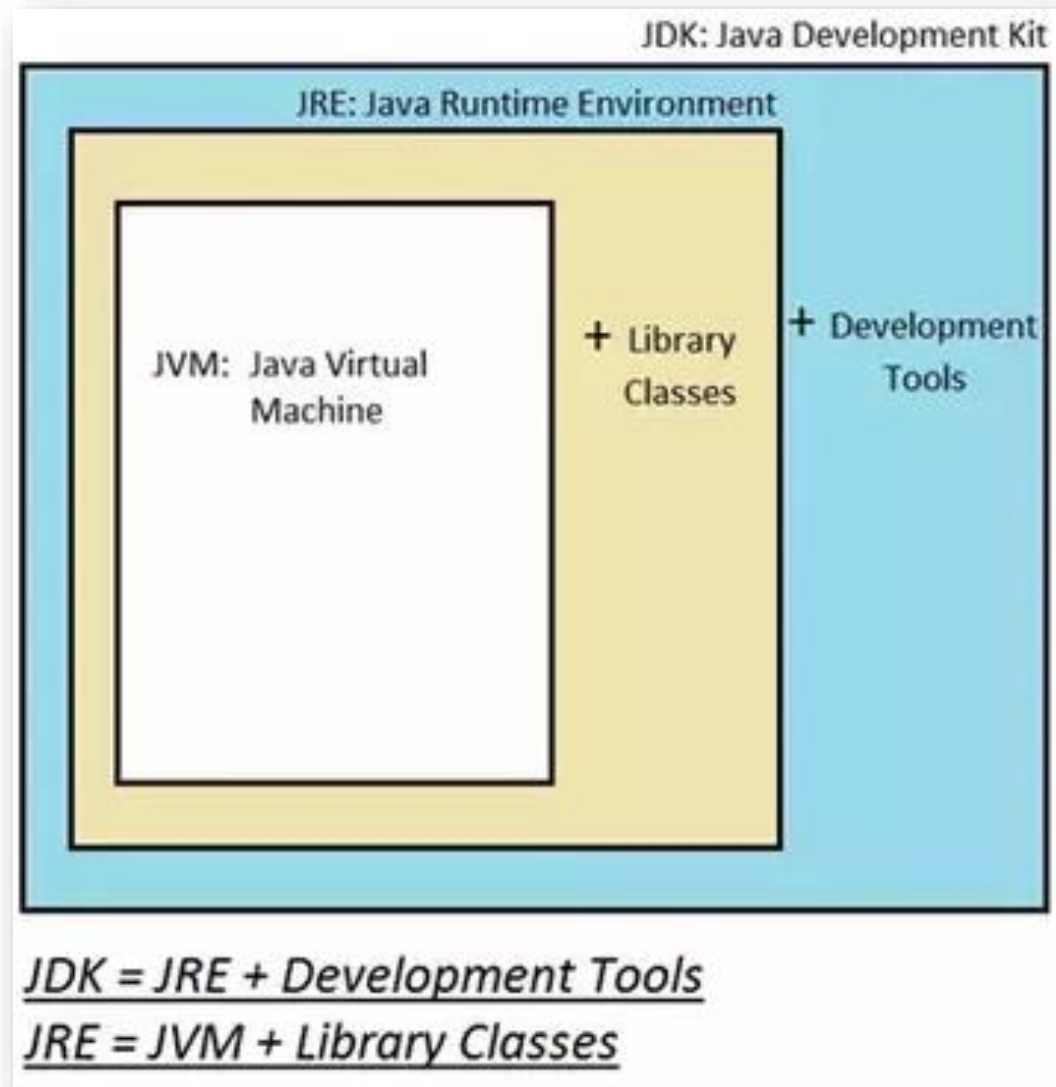
JRE

- **Java Runtime Environment – JRE** is an installation package which provides environment to **only run (not develop)** the java program (or application) onto your machine.
- JRE is only used by **end users** of your system.
- It contains **set of libraries + other files** that JVM uses at runtime.

JVM

- **Java Virtual Machine - JVM** is used to *run the compiled source codes* platform independently.
- Whatever java program you run using JRE or JDK goes into JVM and JVM is responsible to *execute the java program line by line* hence it is also known as **interpreter**.
- You don't need to install JVM separately into your machine because it is inbuilt into your JDK or JRE installation package

JDK – JRE – JVM Concepts



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.

We are going to use J2SE to introduce Java programming.

Popular Java IDEs

- Instead of using the JDK for developing and testing java programs, you can use a Java development tool software that provides an integrated development environment (IDE) for developing Java programs quickly.
- Editing, compiling, building, debugging, and online help are integrated in one graphical user interface.
 - NetBeans
 - Eclipse

Before Developing a Java Code

1. From the address given below download the installer of the JDK 8 (Java SE Development Kit 8) for your operating system. Since JDK includes JRE and JVM, it is sufficient only to install JDK.

<https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>

2. Install an IDE from the given addresses for free:

➤ NetBeans: <https://netbeans.org/downloads/>

➤ Eclipse: <https://eclipse.org/downloads/>

A Simple Java Program

```
// This program prints Welcome to Java!

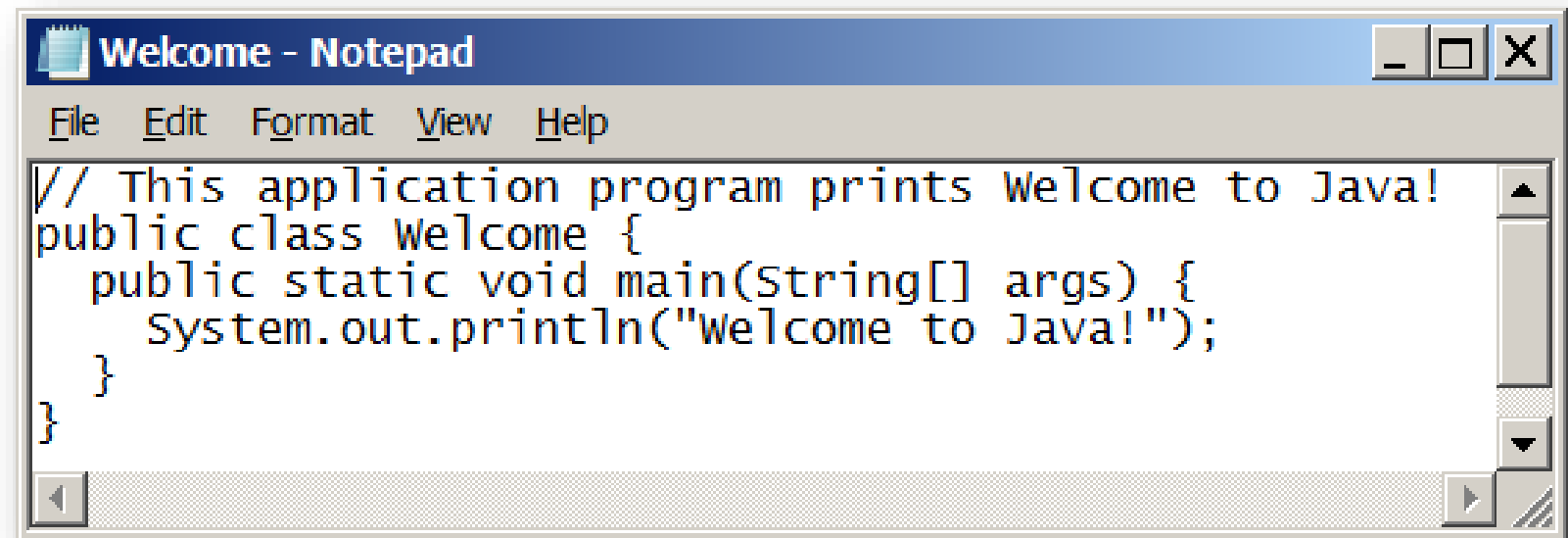
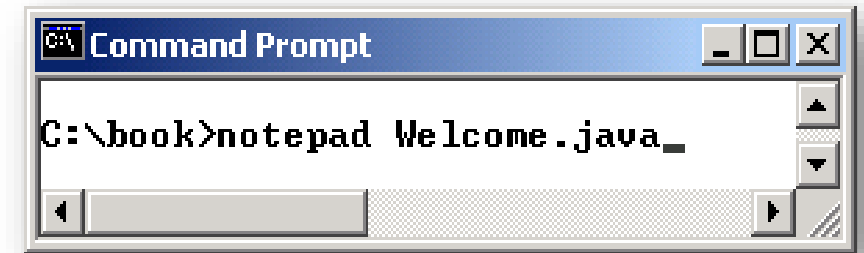
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Animation and Live Example

[Intro to Java Programming, Y. Daniel Liang - Welcome.java \(pearsoncmg.com\)](https://www.pearsoncmg.com)

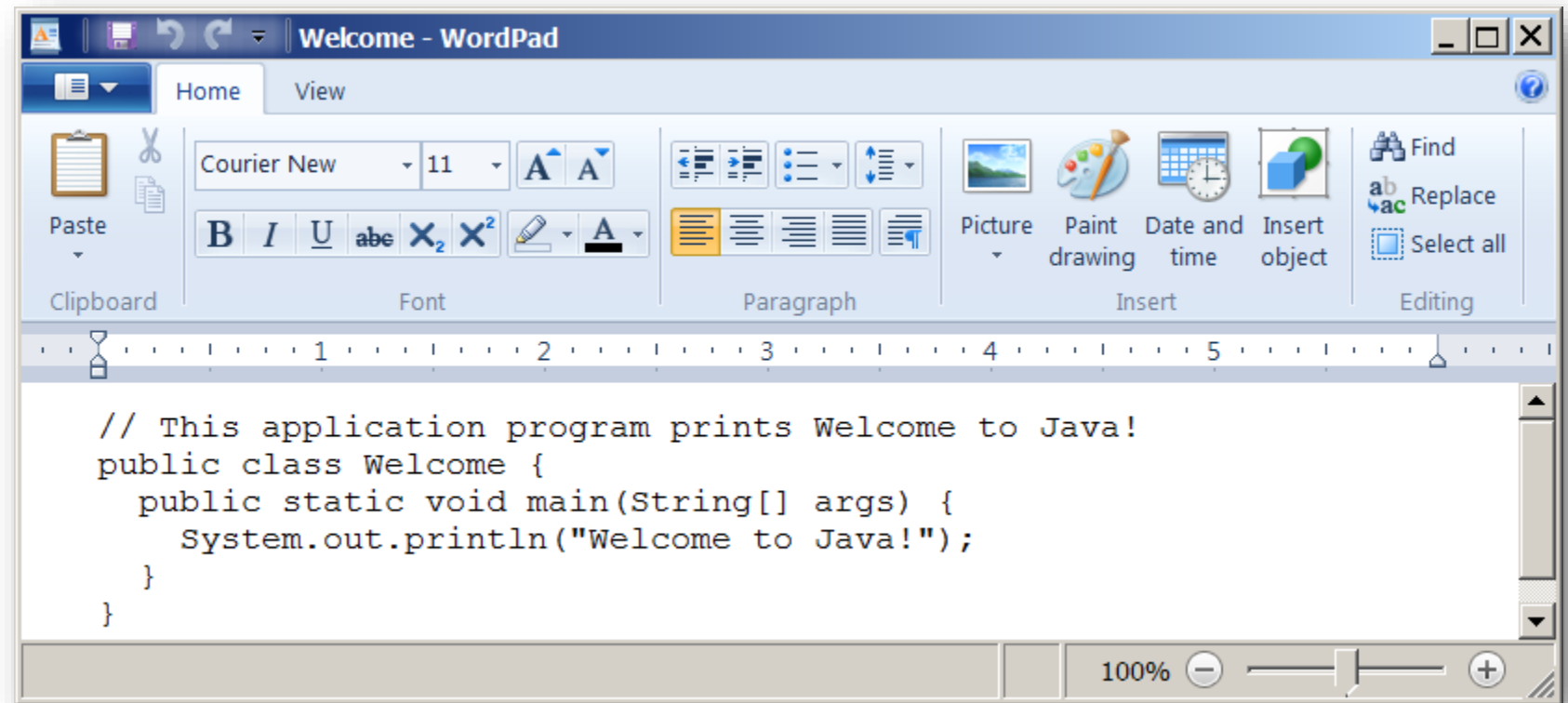
Creating and Editing Using Notepad

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



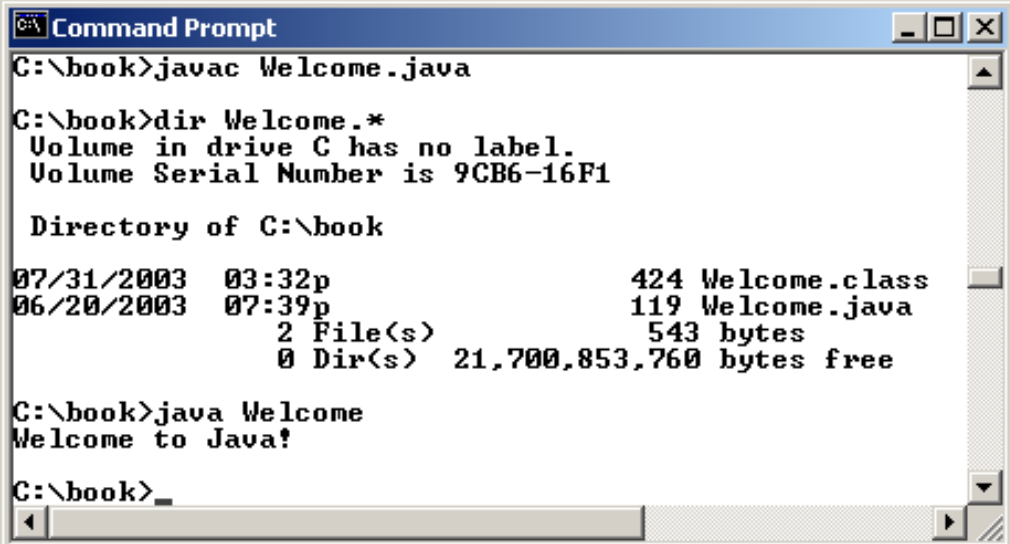
Creating and Editing Using WordPad

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



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

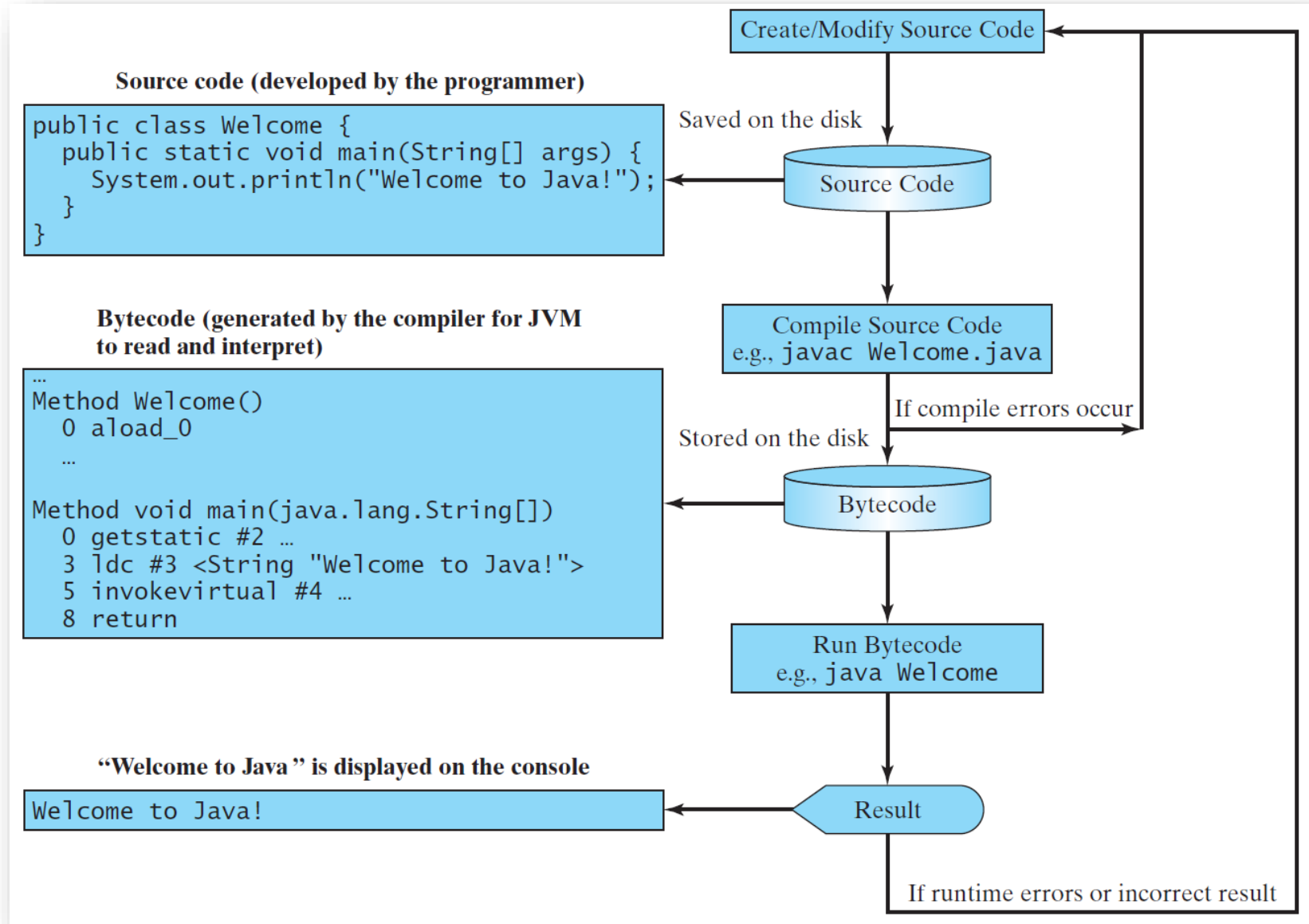
Directory of C:\book

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>
```

Creating, Compiling, and Running Programs



Trace a Program Execution

1. 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

2. 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!");  
    }  
}
```



3. Prints a message to the console

EXERCISE

- Write a program which displays the result of the expression:

$$(15.3 - 5 * 2) / (45 * 2.1)$$

- Your output should look like this:

```
(15.3 - 5 * 2) / (45 * 2.1) = 0.05608465608465609
```

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 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!");  
    }  
}
```

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.

```
// 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.

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

Method Block

Class 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.

Naming Conventions

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

`ComputeExpression`

Proper Indentation and Spacing

- Indentation
 - Indent each component or statement at least two spaces.
- Spacing
 - A single space should be added on both sides of a binary operator.

```
System.out.println(3+4*4);
```

Bad style

```
System.out.println(3 + 4 * 4);
```

Good style

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**

- Easy to detect because the compiler tells you where they are and what caused them.
- Syntax errors result from errors in code construction
 - mistyping a keyword,
 - omitting some necessary punctuation,
 - using an opening brace without a corresponding closing brace.

Syntax Errors

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

Syntax Errors

[Intro to Java Programming, Y. Daniel Liang - ShowSyntaxErrors.java \(pearsoncmg.com\)](#)

Programming Errors

- **Runtime Errors**

- Causes the program to terminate abnormally
- They occur while a program is running if the environment detects an operation that is impossible to carry out.
- Input mistakes typically cause runtime errors
 - if the program expects to read in a number, but instead the user enters a string
 - division by zero

Runtime Errors

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

Runtime Errors

[Intro to Java Programming, Y. Daniel Liang - ShowRuntimeErrors.java \(pearsoncmg.com\)](#)

Programming Errors

- **Logic Errors**
 - Occurs when a program does not perform the way it was intended to.
 - Produces incorrect result

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);  
    }  
}
```

Show Logic Errors

[Intro to Java Programming, Y. Daniel Liang - ShowLogicErrors.java \(pearsoncmg.com\)](http://pearsoncmg.com)