

Lecture 1: Introduction to Java (Chapter 1)

Adapted by Fangzhen Lin for COMP3021 from Y.
Danial Liang's PowerPoints for Introduction to Java
Programming, Comprehensive Version, 9/E,
Pearson, 2013.

Objectives

- ➡ 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 display output using the JOptionPane message dialog boxes (§1.9).
- ➡ To become familiar with Java programming style and documentation (§1.10).
- ➡ To explain the differences between syntax errors, runtime errors, and logic errors (§1.11).

How Data is Stored?

Data of various kinds, such as numbers, characters, and strings, are encoded as a series of bits (zeros and ones). Computers use zeros and ones because digital devices have two stable states, which are referred to as *zero* and *one* by convention. The programmers need not to be concerned about the encoding and decoding of data, which is performed automatically by the system based on the encoding scheme. The encoding scheme varies. For example, character 'J' is represented by 01001010 in one byte. A small number such as three can be stored in a single byte. If computer needs to store a large number that cannot fit into a single byte, it uses a number of adjacent bytes. No two data can share or split a same byte. A byte is the minimum storage unit.

Memory address	Memory content	
.	.	
.	.	
.	.	
2000	01001010	Encoding for character 'J'
2001	01100001	Encoding for character 'a'
2002	01110110	Encoding for character 'v'
2003	01100001	Encoding for character 'a'
2004	00000011	Encoding for number 3

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.

Popular High-Level Languages

Language	Description
Ada	Named for Ada Lovelace, who worked on mechanical general-purpose computers. The Ada language was developed for the Department of Defense and is used mainly in defense projects.
BASIC	Beginner's All-purpose Symbolic Instruction Code. It was designed to be learned and used easily by beginners.
C	Developed at Bell Laboratories. C combines the power of an assembly language with the ease of use and portability of a high-level language.
C++	C++ is an object-oriented language, based on C.
C#	Pronounced "C Sharp." It is a hybrid of Java and C++ and was developed by Microsoft.
COBOL	COmmon Business Oriented Language. Used for business applications.
FORTRAN	FORmula TRANslation. Popular for scientific and mathematical applications.
Java	Developed by Sun Microsystems, now part of Oracle. It is widely used for developing platform-independent Internet applications.
Pascal	Named for Blaise Pascal, who pioneered calculating machines in the seventeenth century. It is a simple, structured, general-purpose language primarily for teaching programming.
Python	A simple general-purpose scripting language good for writing short programs.
Visual Basic	Visual Basic was developed by Microsoft and it enables the programmers to rapidly develop graphical user interfaces.

Why Java?

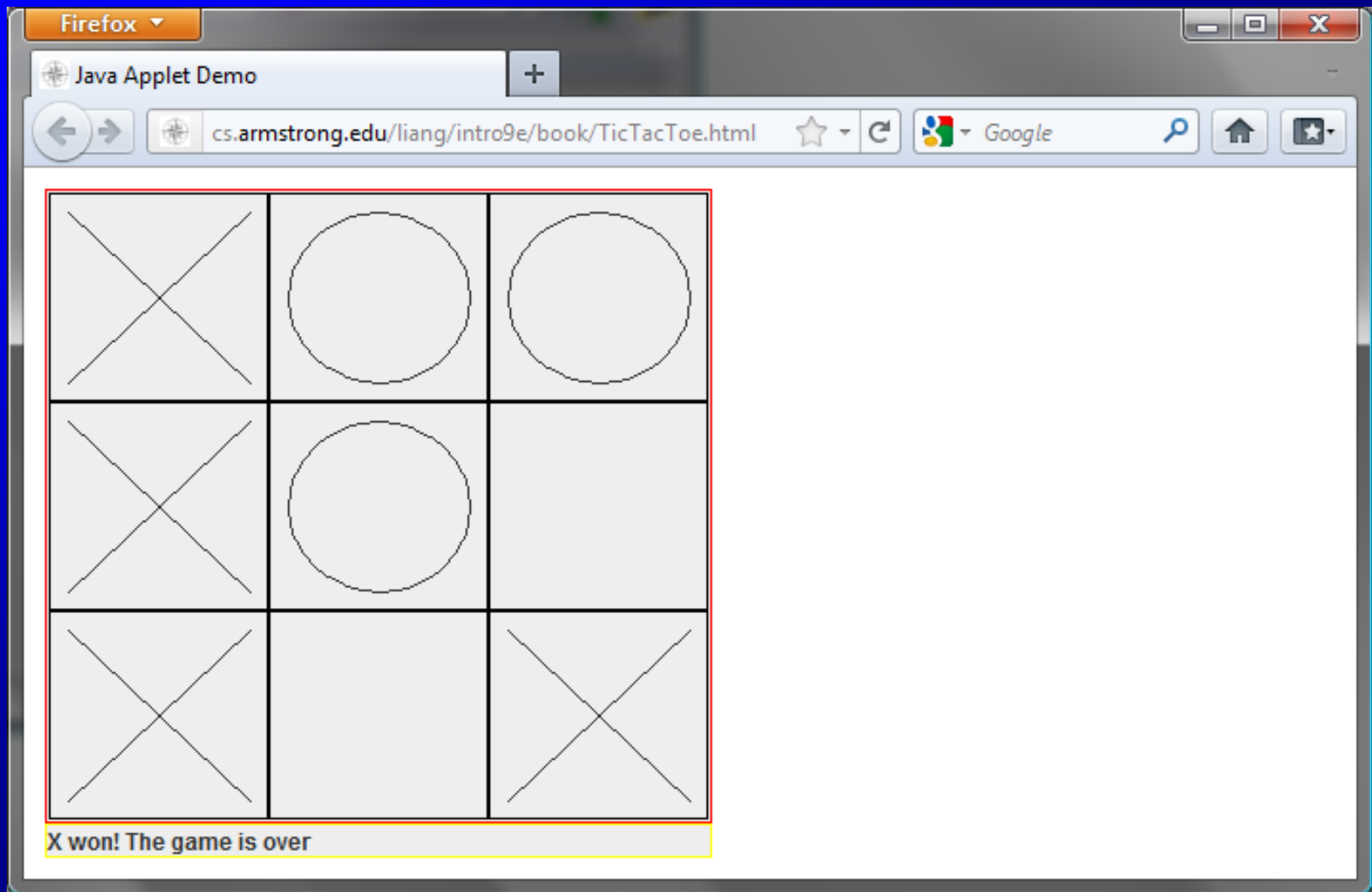
The answer is that Java enables users to develop and deploy applications on the Internet for servers, desktop computers, and small hand-held devices. The future of computing is being profoundly influenced by the Internet, and Java promises to remain a big part of that future. Java is the Internet programming language.

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

Java, Web, and Beyond

- ☞ Java can be used to develop Web applications.
- ☞ Java Applets
- ☞ Java Web Applications
- ☞ Java can also be used to develop applications for hand-held devices such as Palm and cell phones

Examples of Java's Versatility (Applets)



PDA and Cell Phone



Java's History

- ☞ James Gosling and Sun Microsystems

- ☞ Oak

- ☞ Java, May 20, 1995, Sun World

- ☞ HotJava

 - The first Java-enabled Web browser

- ☞ Early History Website:

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

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

www.cs.armstrong.edu/liang/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 implements several security mechanisms to protect your system against harm caused by stray programs.
- ☞ 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
 - ☞ Java Is Secure
 - ☞ **Java Is Architecture-Neutral** Write once, run anywhere
 - ☞ Java Is Portable
 - ☞ Java's Performance
 - ☞ Java Is Multithreaded
 - ☞ Java Is Dynamic
- With a Java Virtual Machine (JVM), you can write one program that will run on any platform.

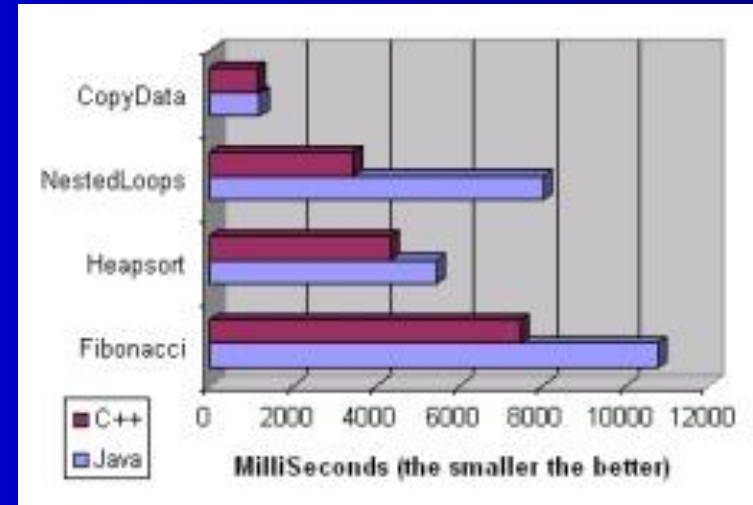
Characteristics of Java

- ☞ 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



Extracted from: **C++ vs. Java 1.6 - A fair benchmark**
<http://www.irrlicht3d.org/pivot/entry.php?id=446>

Java runs slightly slower than C++

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 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 use J2SE to introduce Java programming.

Popular Java IDEs

☞ NetBeans

☞ Eclipse: used for this course and taught in the lab.

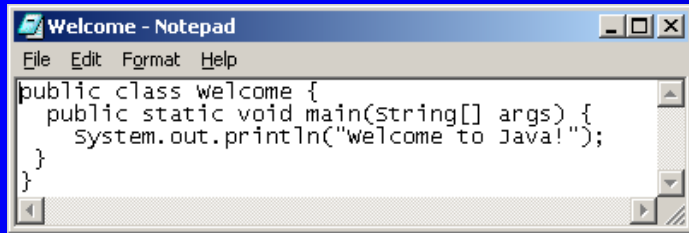
A Simple Java Program

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

Creating, Compiling, and Running Programs



```
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 ...  
  8 return
```

Saved on the disk

Create/Modify Source Code

Source Code

Compile Source Code
i.e., `javac Welcome.java`

If compilation errors

stored on the disk

Bytecode

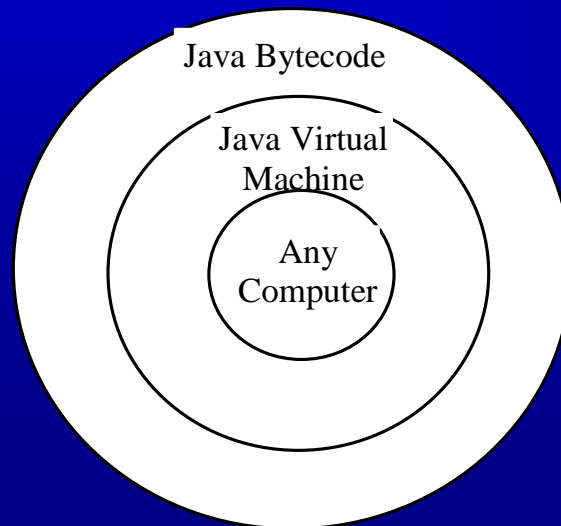
Run Bytecode
i.e., `java Welcome`

Result

If runtime errors or incorrect result

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.



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!");  
    }  
}
```

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.

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

Class block

Method block

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.”

WelcomeInMessageDialogBox

The showMessageDialog Method

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



Two Ways to Invoke the Method

There are several ways to use the showMessageDialog method. For the time being, all you need to know are two ways to invoke it.

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

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

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);
```

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

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");  
    }  
}
```

Runtime Errors

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

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);  
    }  
}
```

Supplements on the Companion Website

- ☞ See Supplement I.B for installing and configuring JDK
- ☞ See Supplement I.C for compiling and running Java from the command window for details

www.cs.armstrong.edu/liang/intro9e/supplement.html