Chapter 1 Introduction to Computers, Programs, and Java

Edited By: Huda Saadeh

Programs

Computer *programs*, known as *software*, are instructions written using programming languages to tell the computer what to do

Computers do not understand human languages, so you need to use computer languages to communicate with them.

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 instructions. Programs in M.L. are highly difficult to read and modify. For example, to add two numbers, you might write an instruction in binary like this:

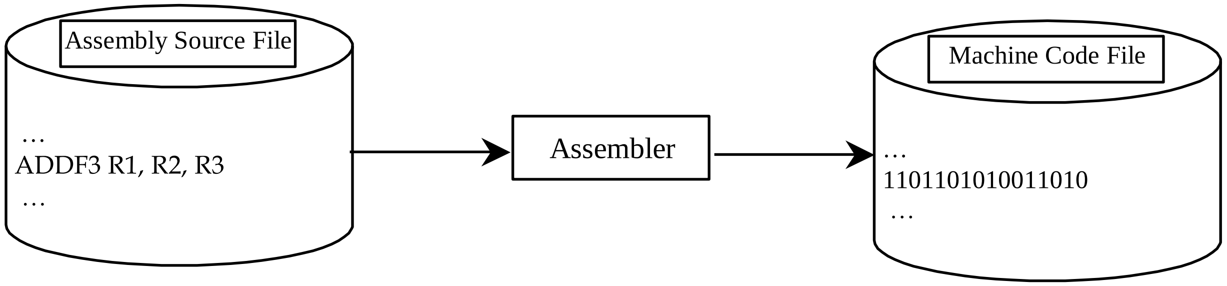
```
1101101010011010
```

Programming Languages

Machine Language **Assembly Language** High-Level Language

Assembly languages were developed to make programming easy. Since the computer cannot understand assembly language, however, a program called assembler is used to convert assembly language programs into machine code. For example, to add two numbers, you might write an instruction in assembly code like this:

```
ADDF3 R1, R2, R3
```



PROGRAMMING LANGUAGES

Machine Language   Assembly Language   **High-Level Language**

The high-level languages are English-like and easy to learn and program. 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;
```

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.

Interpreting/Compiling Source Code

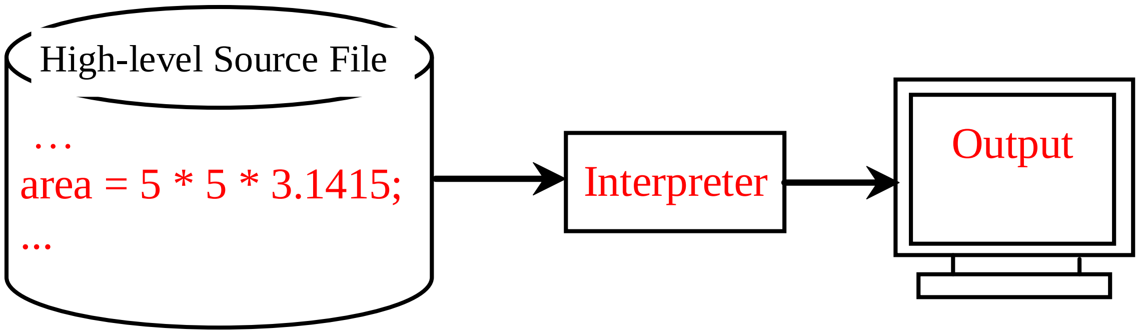
A program written in a high-level language is called a *source program* or *source code*. Because a computer cannot understand a source program, just like assembly programs a source program must be translated into machine code for execution.

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 following figure. Note that a statement from the source code may be translated into several machine instructions.

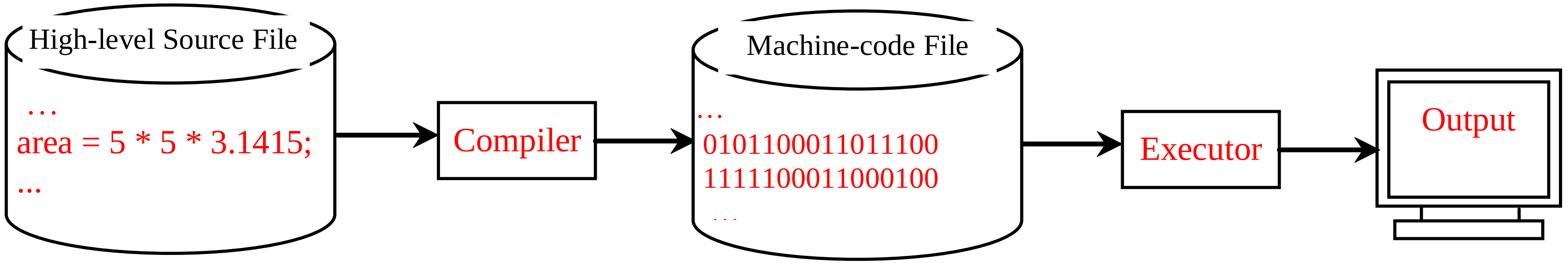
(Just one stmt. In a time)



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.

Compilers first check for errors in the source code, and only the error-free version of a source code is translated into M.L.



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

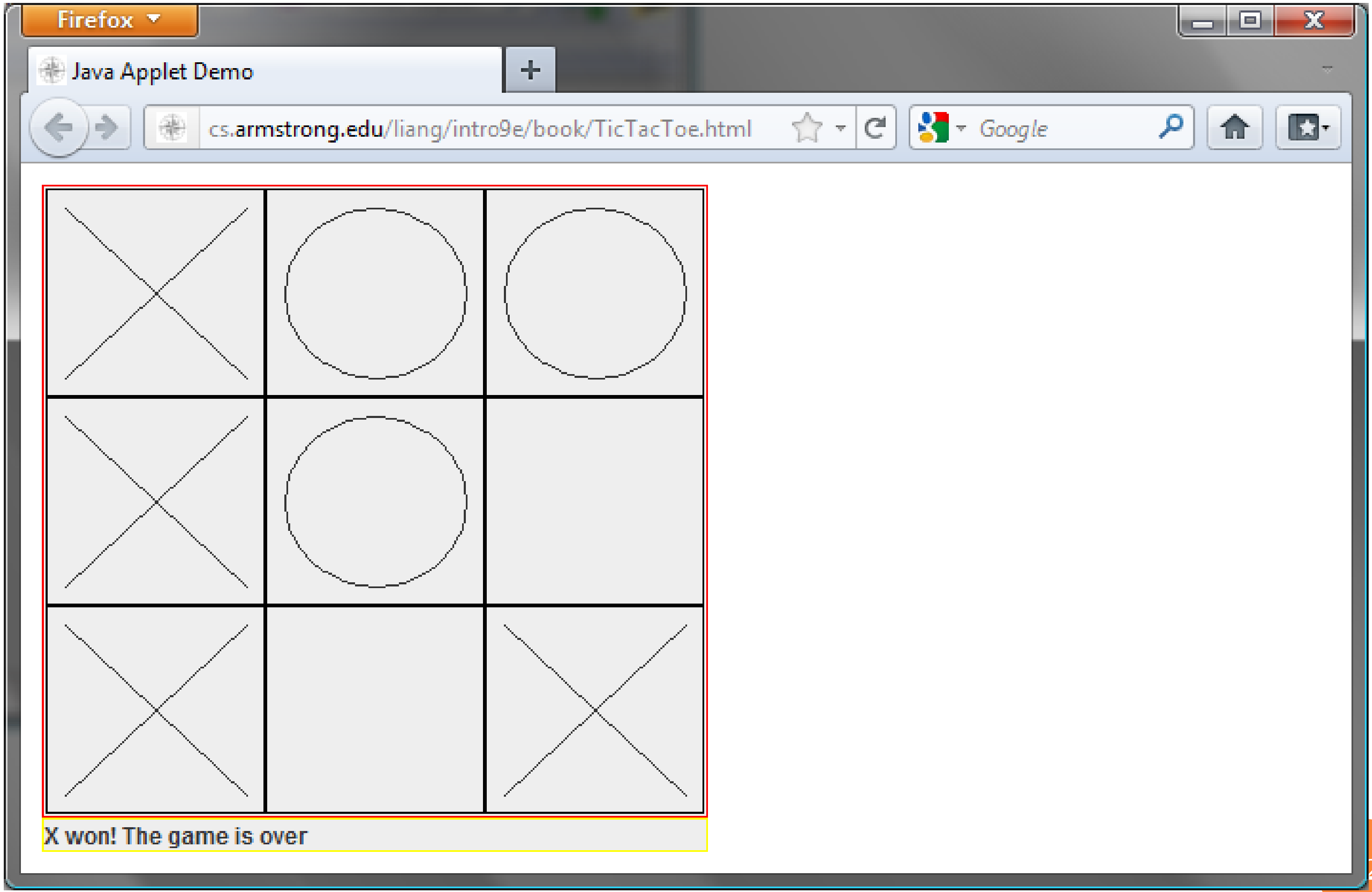
http://www.youtube.com/watch?v=Hkwcv9hf0Wc

- http://www.youtube.com/watch?v=yfcyeY-jQbI

Java, Web, and Beyond

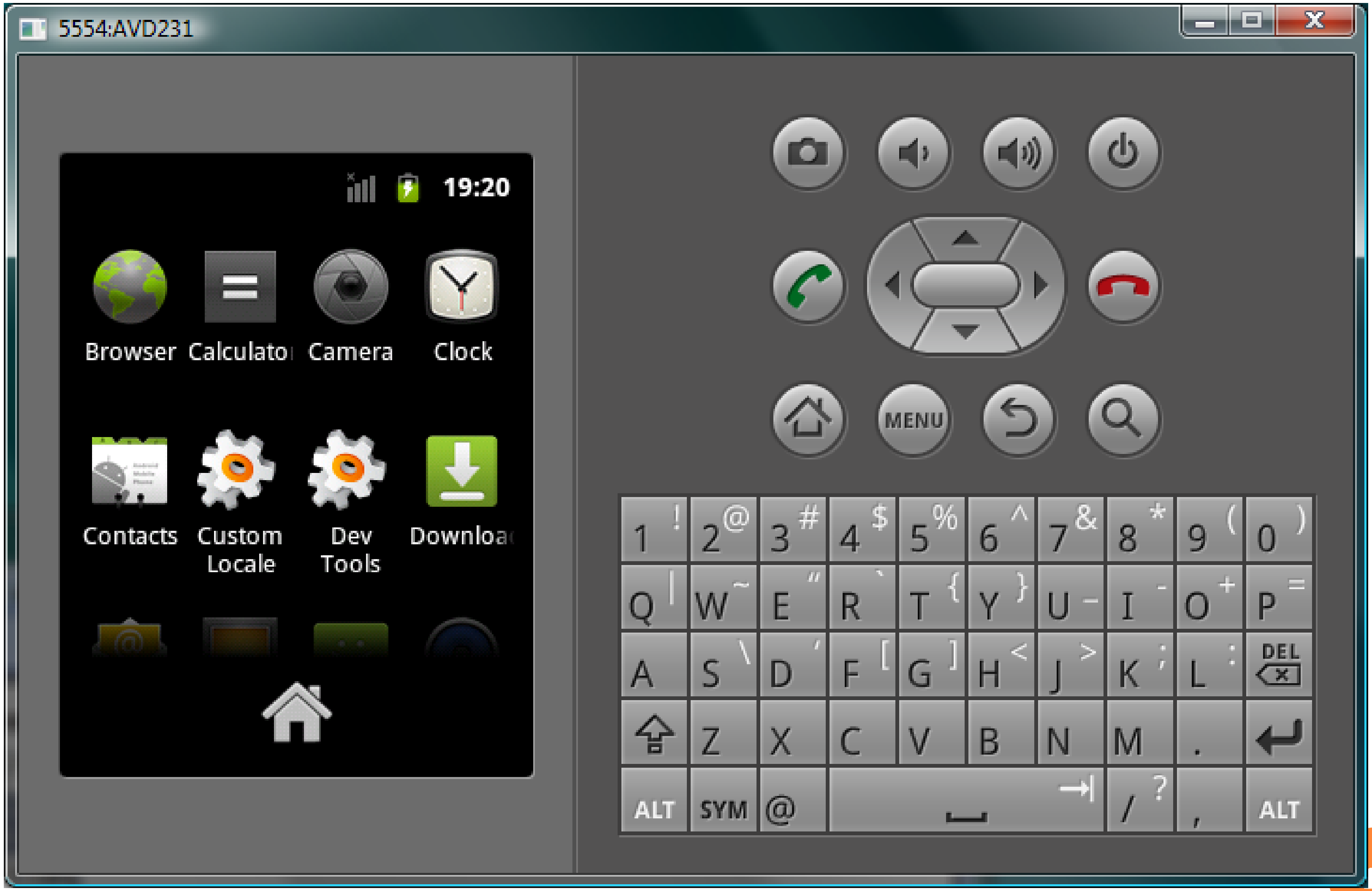
- Java can be used to develop Web applications.
- Java Applets
- Java can also be used to develop applications for hand-held devices such as Palm and cell phones

Examples of Java's Versatility (Applets)



X won! The game is over

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

Companion Website

- 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

Companion Website

- 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

Companion Website

- 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

Companion Website

- 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

Companion Website

- 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

Companion Website

- 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

Companion Website

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

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.

This book uses J2SE to introduce Java programming.

Popular Java IDEs

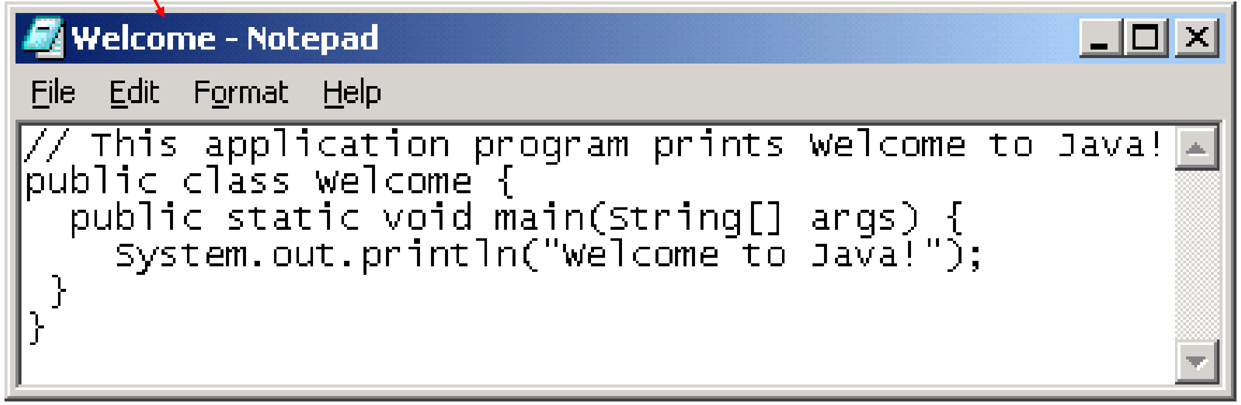
- NetBeans
- Eclipse
- TextPad
- JCreator
- JBulder
- JBlue

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!");
  }
}
```

Creating and Editing Using NotePad

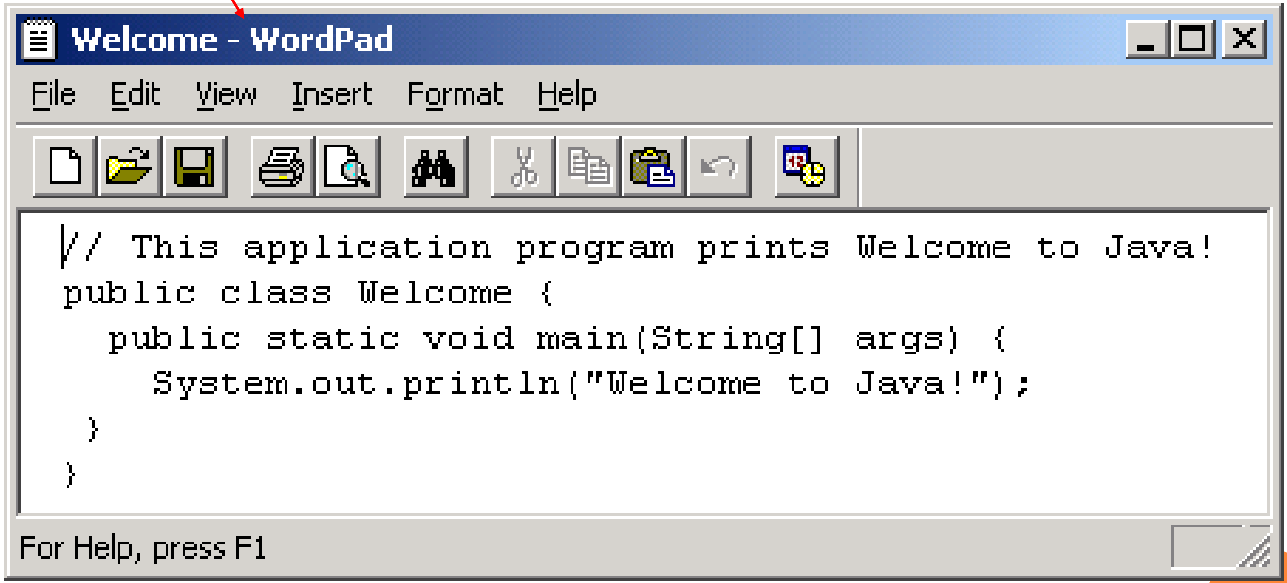
Open Notepad program and type java code you want



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

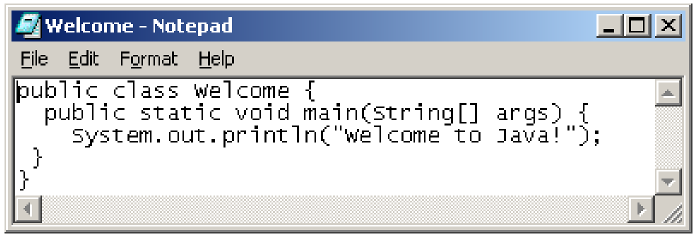
Creating and Editing Using WordPad

Open **WordPad** program and type java code you want

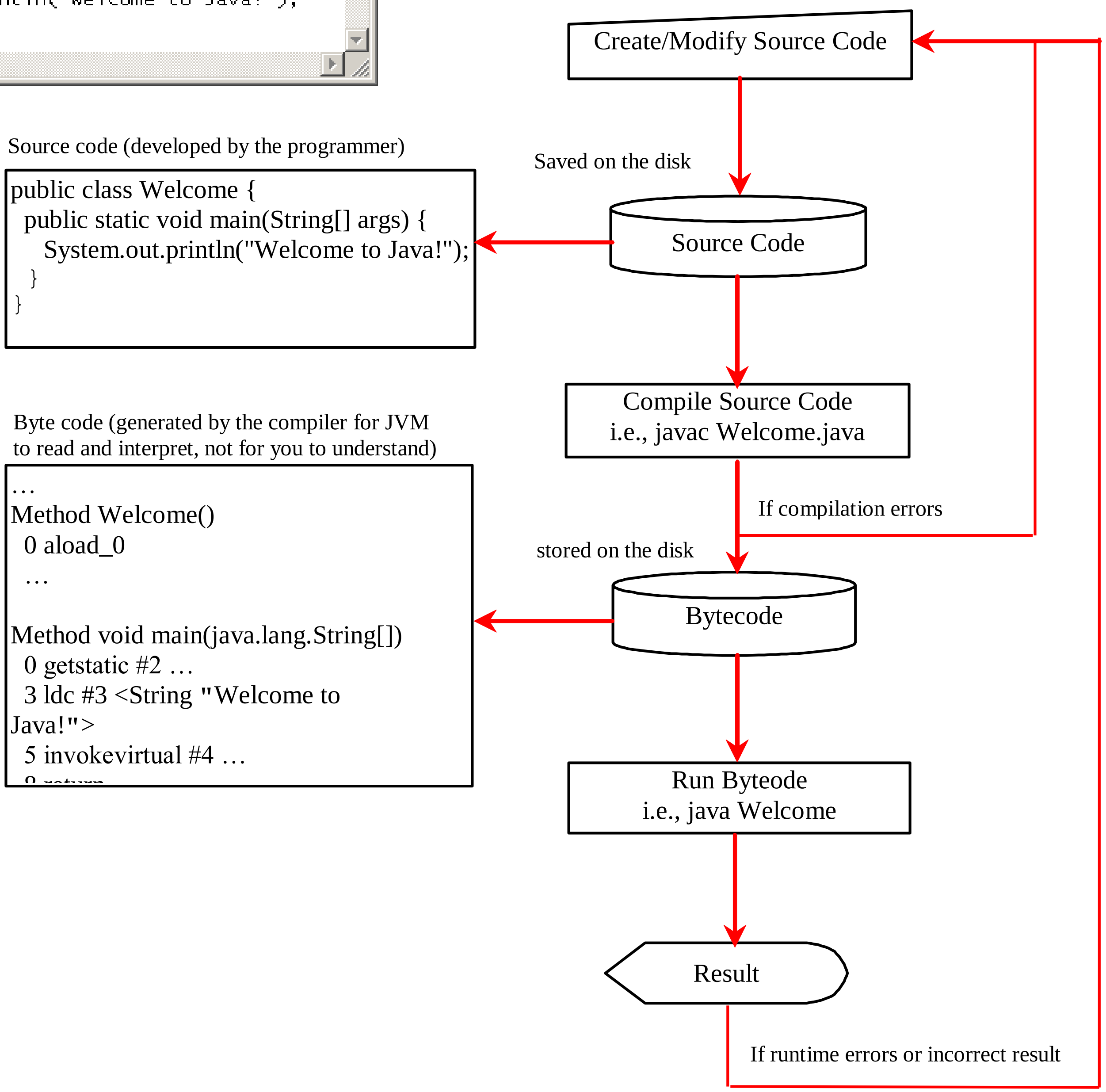


```
// This application program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

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)

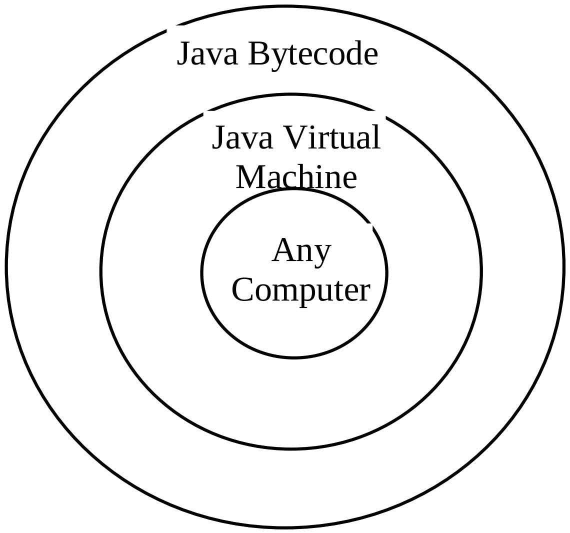
```
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

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 …
```

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.



How Java Works
Java Platform, JVM

- See the following movies:
- http://www.youtube.com/watch?v=aa7Y07Wv1-M
- http://www.youtube.com/watch?v=1D9Q5qiu6Zo
- http://www.youtube.com/watch?v=jQzO1xjkb0E

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!");
  }
}
```

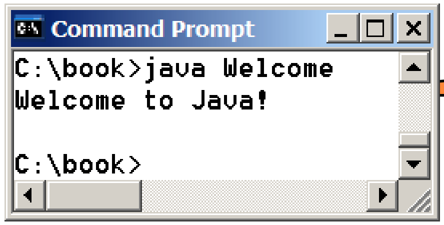
Trace a Program Execution

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!");
  }
}
```



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

C:\book>
```

print a message to the console

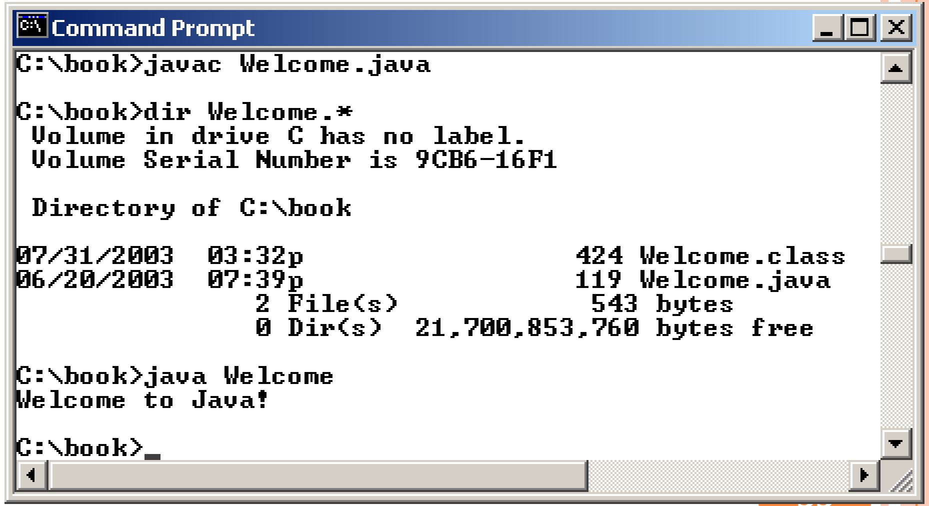
Supplements on the Companion Website

- 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/intro8e

Compiling and Running Java from the Command Window

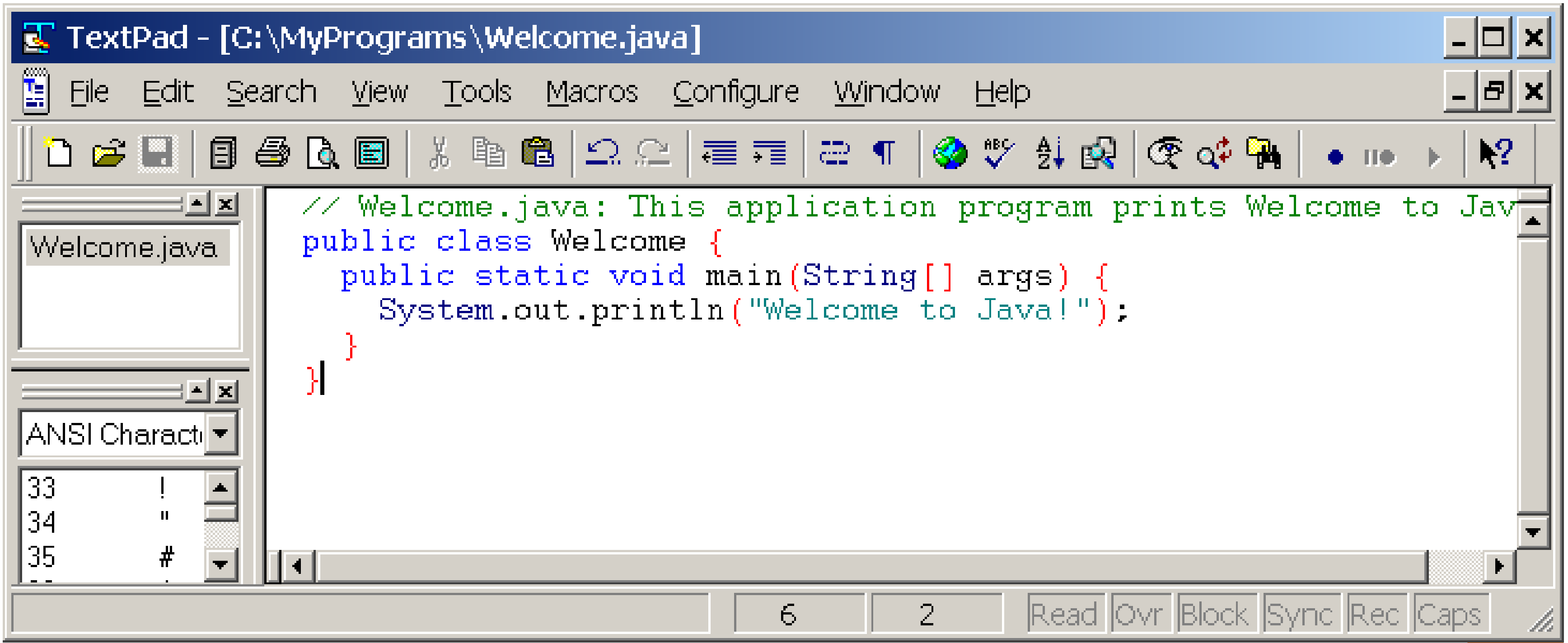
- Set path to JDK bin directory
  - set path=c:\Program Files\java\jdk1.6.0\bin
- Set classpath to include the current directory
  - set classpath=.
- Compile
  - javac Welcome.java
- Run
  - java Welcome



```
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>_
```

Compiling and Running Java from TextPad

- See Supplement II.A on the Website for details


TextPad - [C:\MyPrograms\Welcome.java]
File Edit Search View Tools Macros Configure Window Help
Welcome.java
ANSI Charact
33 !
34 "
35 #
6
2
Read Ovr Block Sync Rec Caps

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

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 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!");
  }
}
```

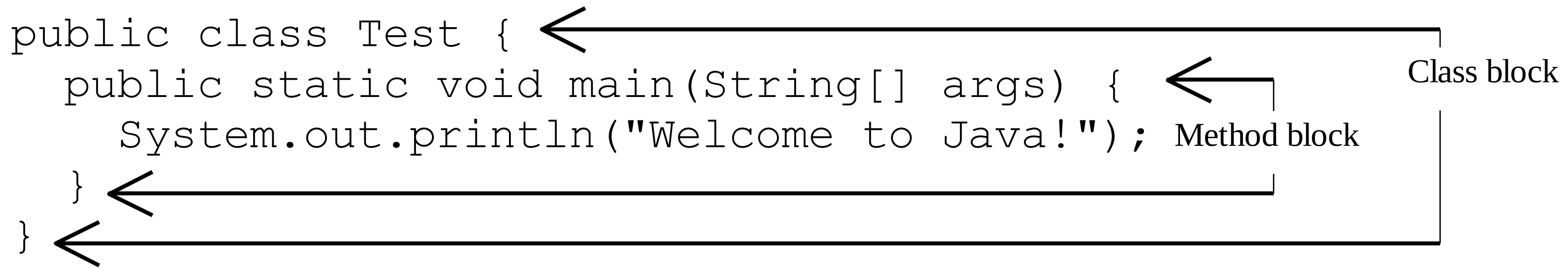
Reserved words

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. And they are written in small letters

```
//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 (set of statements) of a program.



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

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.

Ex.

Try to find the previous symbols in the following code:

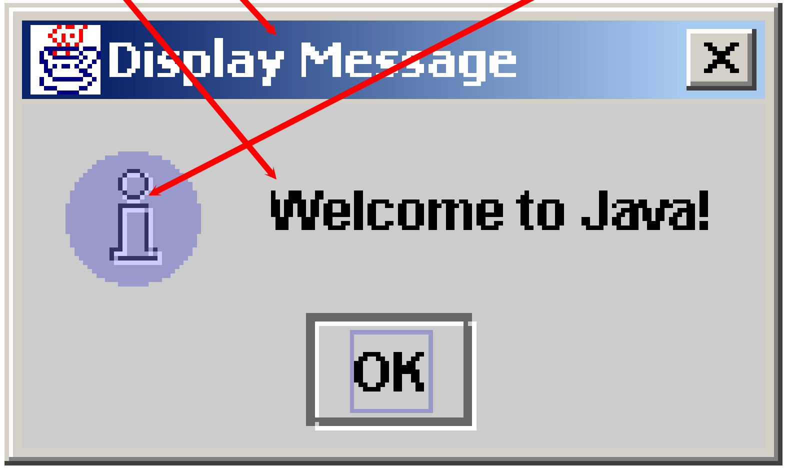
```
//This program prints Welcome to Java!
public class Welcome {
  public static void main(String[]
  args) {
    System.out.println("Welcome to
  Java!");
  }
}
```

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.”

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.

Welcome to Java in a Dialog Box

```
//This program prints Welcome to Java!
import javax.swing.*;
public class Welcome {
  public static void main(String[]
  args) {
      JOptionPane.showMessageDialog(null,
      "Welcome to Java!",
      "Display Message",
      JOptionPane.INFORMATION_MESSAGE);

  }
}
```

Programming Style and Documentation

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

Appropriate Comments

Notes at the beginning of the program to explain what the program does.

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

Find syntax errors?

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

Runtime Errors

Find runTime errors?

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

Logic Errors

Find 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);
  }
}
```