CIS2571 – Intro to Java

Chapter 1 → Intro to Computers, Programs, and Java

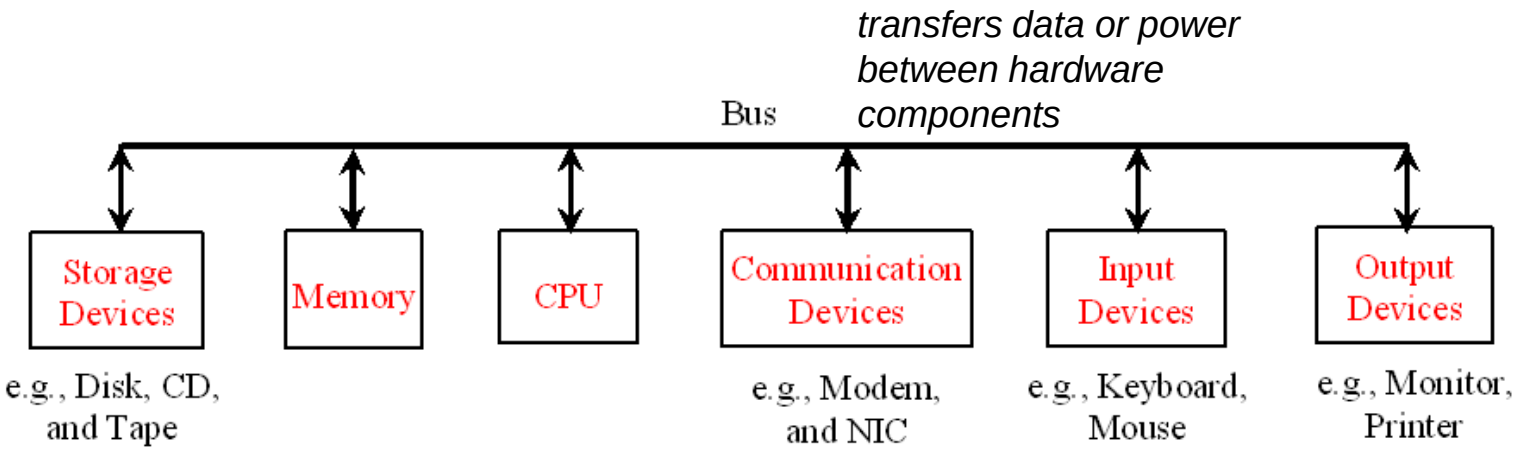
Information consolidated from a variety of textbook and online resources

Topic Objectives

- Know how the components of a computer interact
- Understand the importance of the Operating System
- Understand how data is stored on a computer
- Know the difference between Low-Level and High-Level Languages
- Know the importance of Java for web and application development
- Understand the different Java Language Specifications
- Know how to create and run a simple Java program
  - Programming Style and Documentation
  - Programming Errors

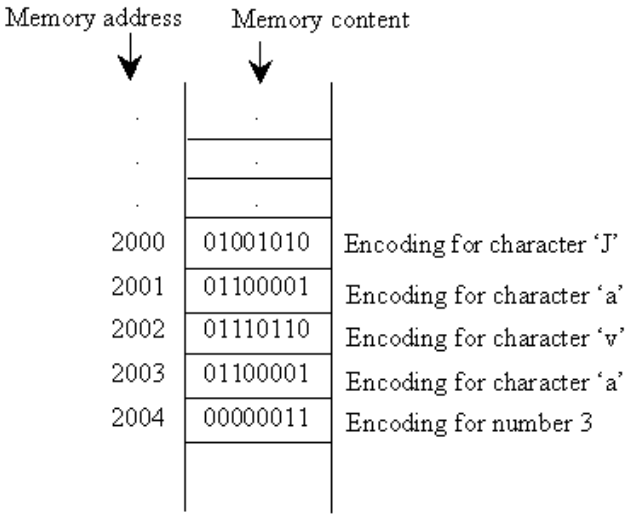
What Is a Computer?

- Central Processing Unit (CPU)
- Memory (main memory)
- Storage Devices
- Input and Output Devices
- Communication Devices



How is Data Stored?

- Data of various kinds, such as numbers, characters, and strings, are encoded as a series of bits
  - 0 OFF False
  - 1 ON True
- A byte is the minimum storage unit.
  - unique memory (RAM) address
- Encoding schemes:
  - ASCII
  - Unicode



What are Computer Programs?

- Computer *programs*, known as *software*, are instructions to the computer telling it what to do.
- Programs are written using programming languages.
  - Machine Language
    - Set of primitive instructions built into every computer
      - `1101101010011010`
  - Assembly Language
    - Developed to make programming easier
      - ADDF3 R1, R2, R3
  - High-Level Language
    - English-like, easy to learn and program
      - X = 10 + 20

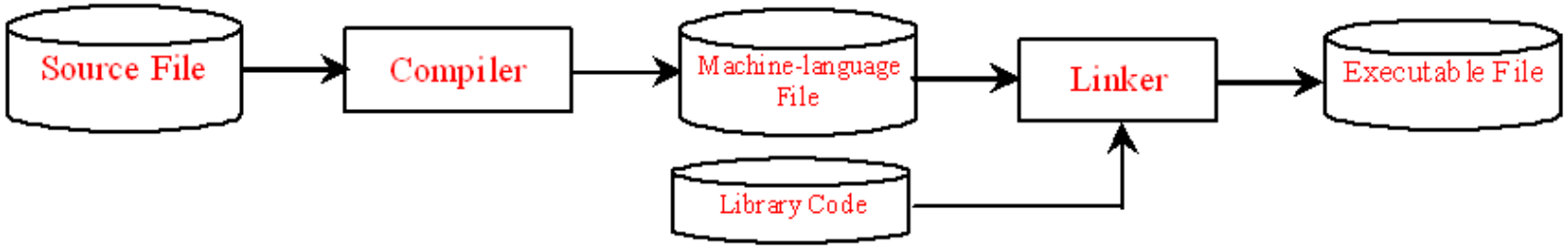
Popular High-Level Languages

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

TIOBE Index

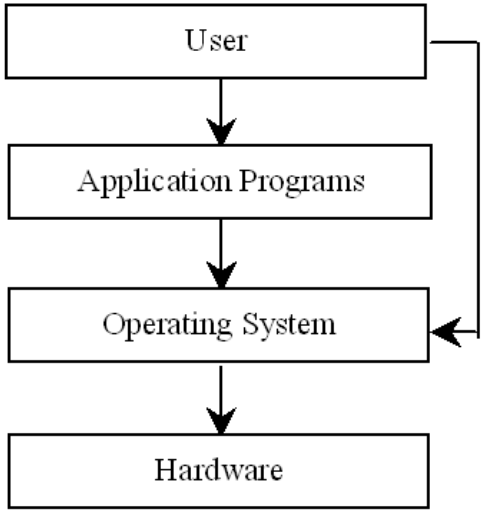
Compiling Source Code

- A program written in a high-level language is called a source program.
- A compiler is used to translate the source program into a machine language program called an object program.
- The object program is often then linked with other supporting library code before the object can be executed.



Operating Systems

- The operating system (OS) is a program that manages and controls a computer's activities.
- Application programs such as an Internet browser and a word processor cannot run without an operating system.
- Major tasks of OS
  - Controlling/monitoring system activities
  - Allocating/assigning system resources
  - Scheduling operations
    - multiprogramming
    - multithreading
    - multiprocessing



Why Java?

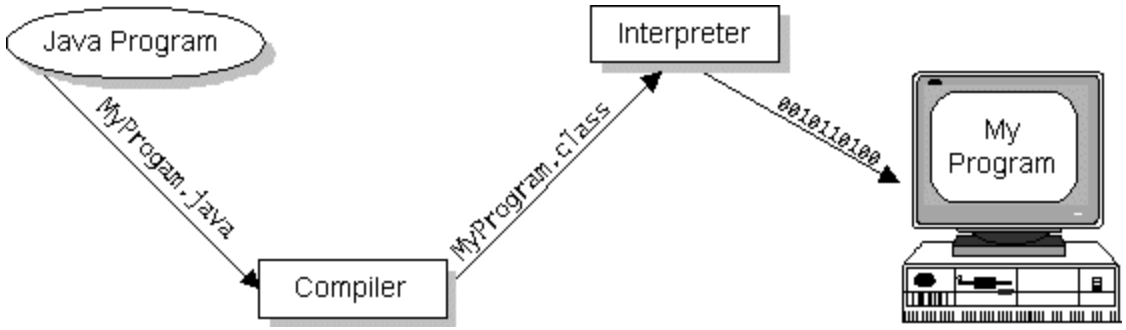
- Java enables users to develop and deploy applications on the Internet for servers, desktop computers, and small hand-held devices.
  - applets run from browsers
- History
  - Java was created by Sun Microsystems in 1991. At that time, it was called "Oak".
  - Created for internal use at Sun because they had many different computers, and they wanted a language that could work on all of them.
  - In 1995 "Oak" was released to the general public under the name that we know today - "Java".
- API (Application Programming Interface) contains pre-defined classes and interfaces

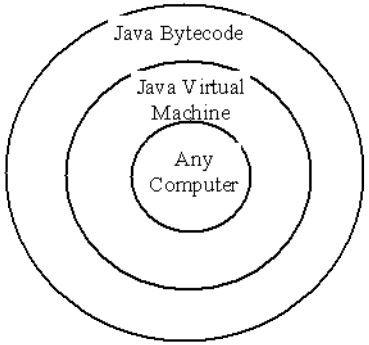
JDK Editions

- 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 and Java ServerPages.
- Java Micro Edition (J2ME).
  - J2ME can be used to develop applications for mobile and other embedded devices such as mobile phones.

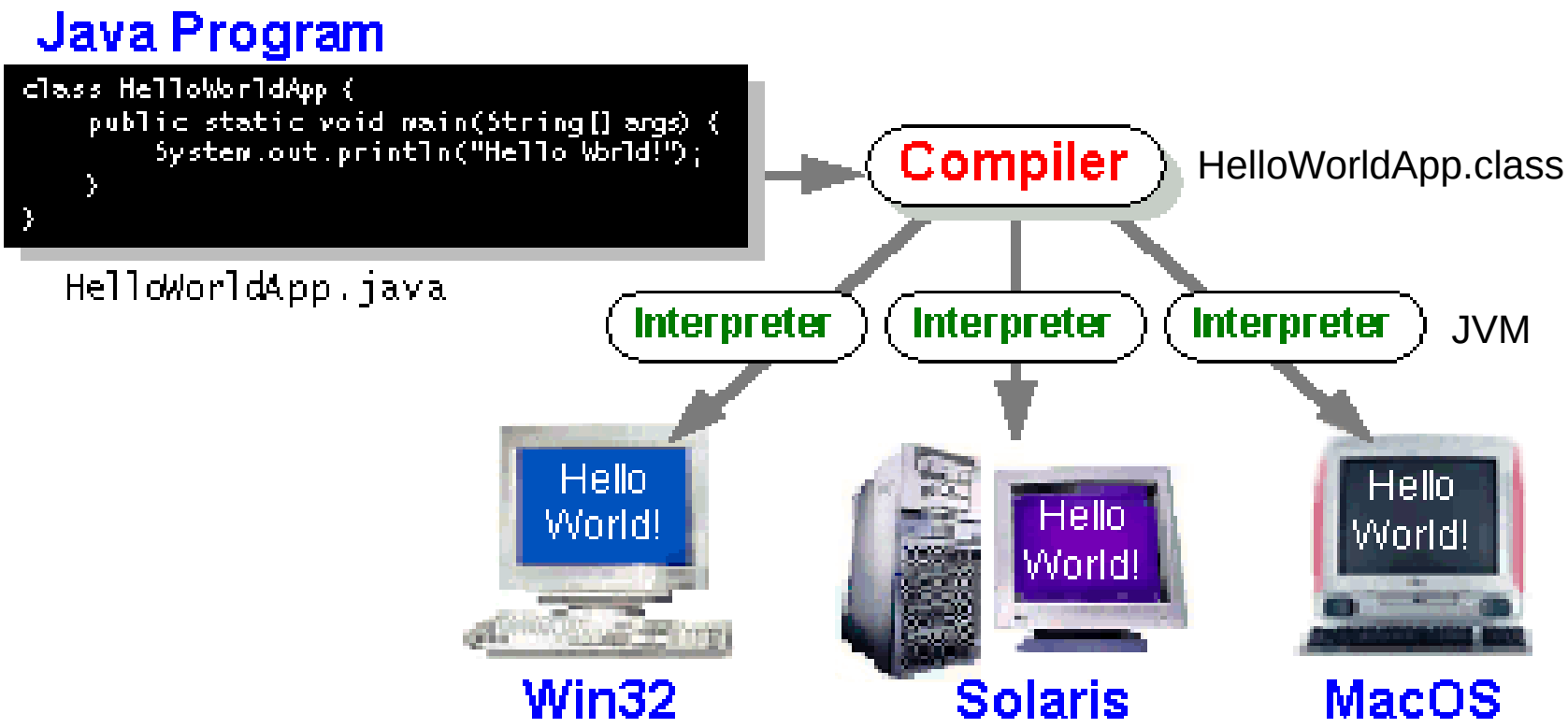
(Some) Java Characteristics

- High-level, object-oriented, portable, interpreted programming language
- Java bytecode makes "write once, run anywhere" possible
- JVM must be installed on your computer before you can run Java programs (the specific version of JVM depends on your operating system).

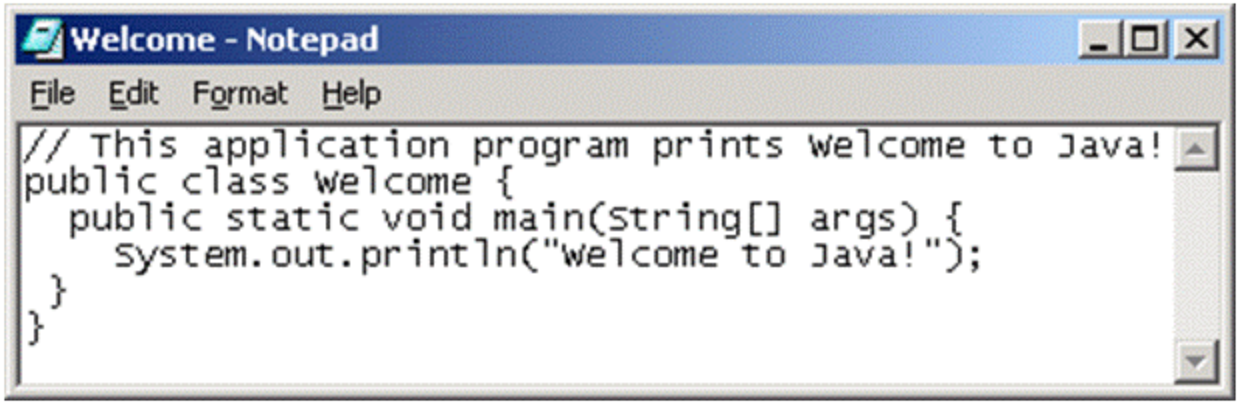




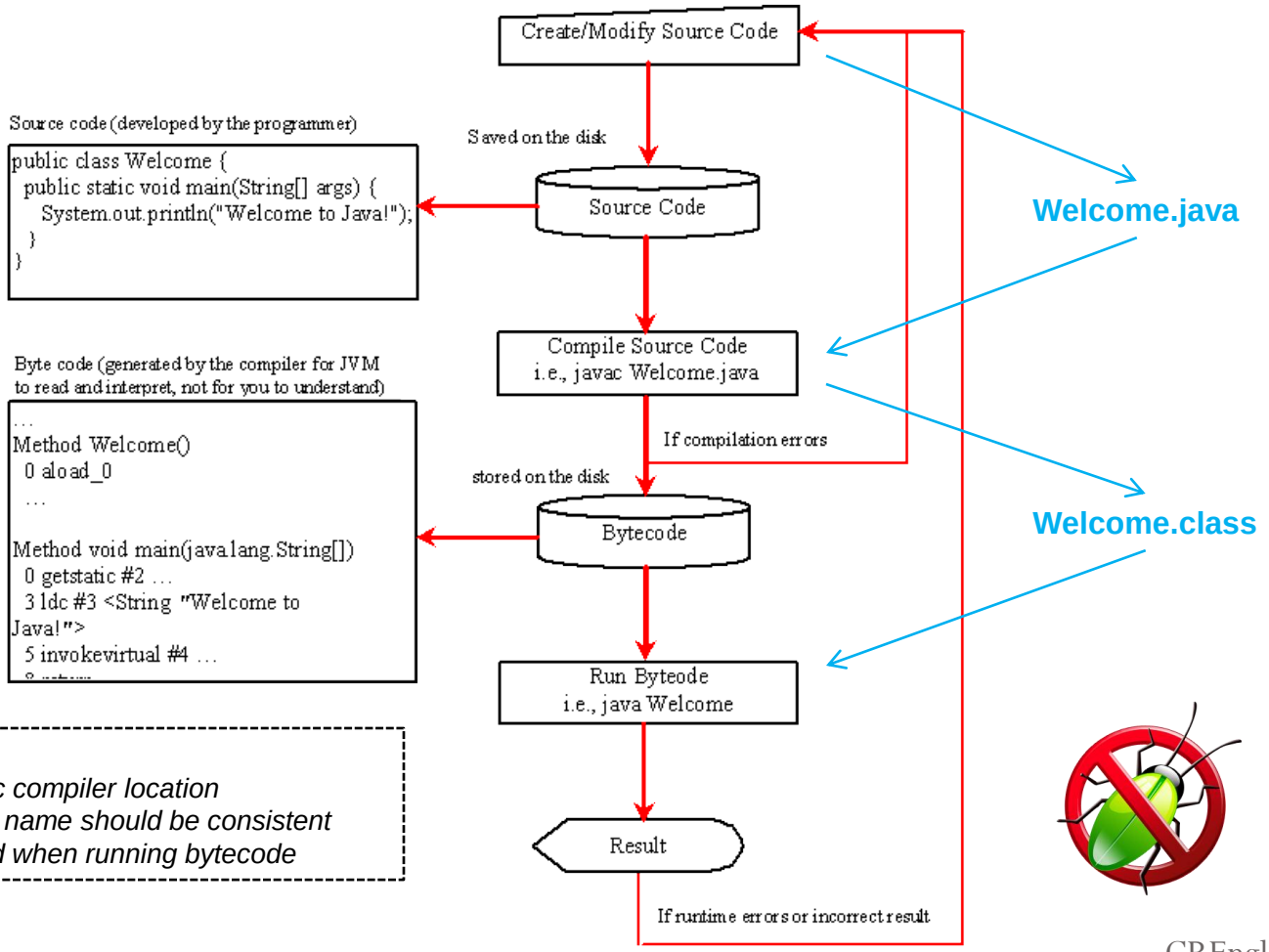
Java Program



Creating and Editing Using Notepad



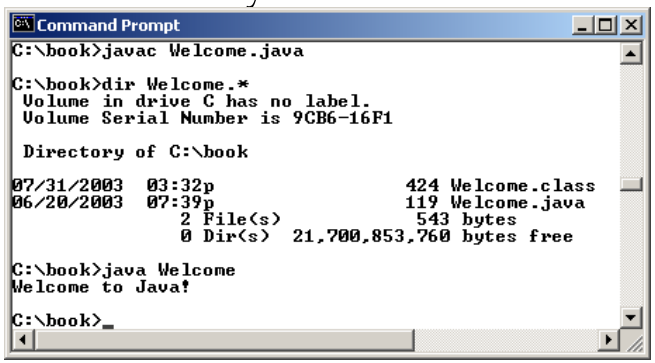
Compiling and Running Java Programs



Notes:
- *path should include javac compiler location*
- *Java class name and file name should be consistent*
- *.class extension not used when running bytecode*

Compiling and Running Java Programs**

- From the Command Window
  - Set path to JDK bin directory
    - set path=C:\Program Files\Java\jdk1.7.0_XX\bin;%path%
  - Set classpath to include the current directory
    - set classpath= .
  - Compile
    - javac Welcome.java
  - Run
    - java Welcome



- IDE (Integrated Development Environments)
  - NetBeans
  - Eclipse
  - Textpad

***C:\Program Files\Java\jdk1.7.0_XX\bin included on path to run compiler and bytecode interpreter directly on command line*

Anatomy of a Java Program

- Comments
  - Line comment
    - `//`
  - Block comment
    - Enclosed between /* and */
- Reserved words
  - Also called keywords
  - Have specific meaning to compiler
    - *class, public, static, void, etc.*
- Modifiers
  - Specifies properties of data, methods, and classes
  - *public, private, protected, static, etc.*

```
/*  This application program displays
    Welcome to Java! Message.
*/

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

Anatomy of a Java Program

- Statements
  - Action or sequence of actions
  - Ends with semicolon ;
- Blocks
  - Groups program components within braces
    - { }
- Classes
  - Essential template or blueprint for objects

```
/*  This application program displays
    Welcome to Java! Message.
*/

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

Anatomy of a Java Program

- Methods
  - Collection of statements that performs a sequence of operations
- The main method
  - Invoked by java interpreter
  - Provides control of program flow

```
/*  This application program displays
    Welcome to Java! Message.
*/

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

(GUI) Displaying Text in a Message Dialog Box

- showMessageDialog is pre-defined method in the JOptionPane class used to display text in message dialog box

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

Programming Style and Documentation

- Good programming style and appropriate documentation reduce the chance of errors and make programs easier to read.
- Include your name, class, instructor, date, file/class name, and brief description at beginning of program to explain what program does, key features, and any unique techniques used
  - Line comment //
  - Block comment /* and */
  - Javadoc comment /** and */
    - Can be extracted into HTML file using JDK's javadoc command
    - Used for commenting on entire class or an entire method

Javadoc Tool Home Page

Programming Style and Documentation

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

- Use consistent indentation
  - Single space on both sides of binary operator
  - Blank lines to separate segments of code
- Use consistent block styles
  - Next-line style
    - Easy to read
  - End-of-line style
    - Saves space

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
  - Also called compile errors
  - Errors that occur during compilation
  - Good practice to debug from top and work downward

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

Programming Errors

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

- Runtime
  - Causes program to terminate abnormally
- Logic
  - Produces incorrect results

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