Java Programming

Introduction to Java

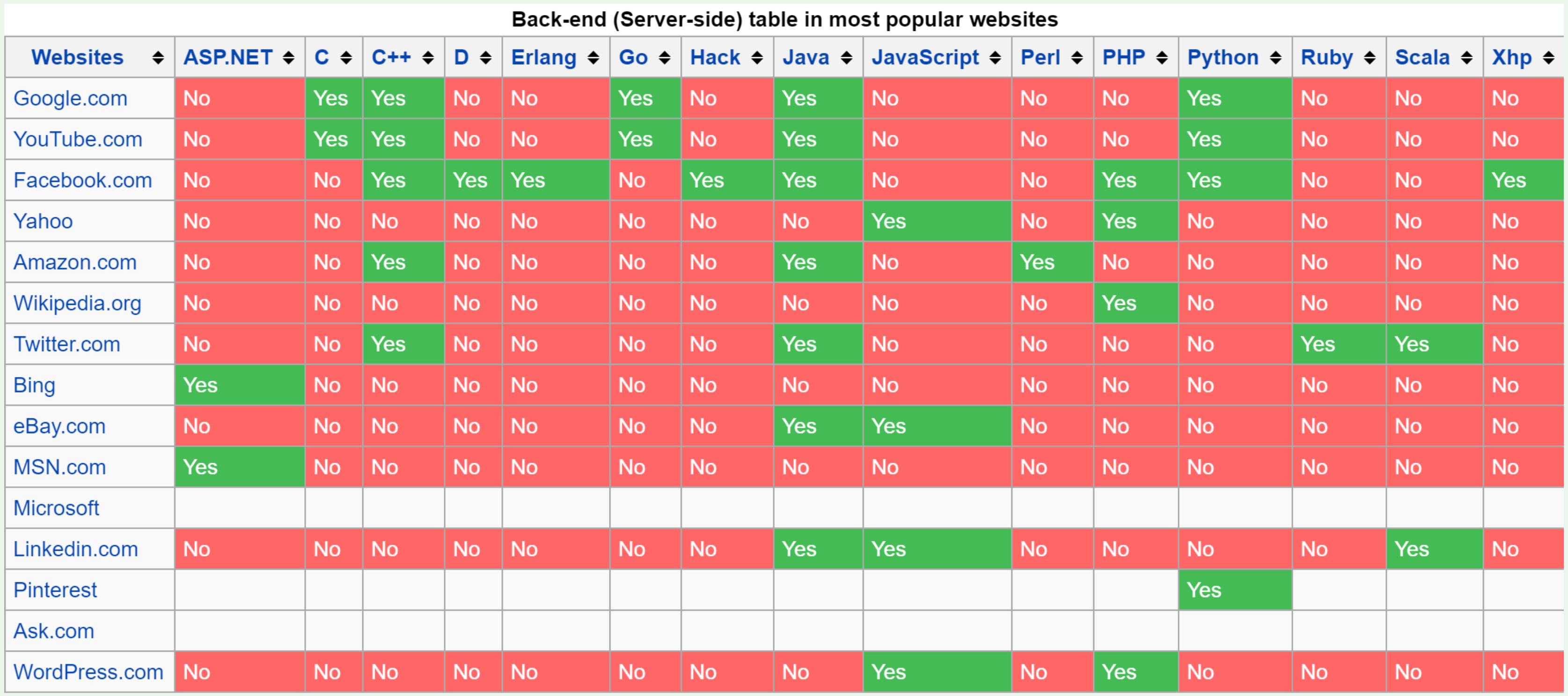
Lesson 1 Objectives

- Explain Java Platform
- Describe Java Programming language
- Write simple Java applications using Eclipse IDE and IntelliJ IDEA Community Edition
  - Eclipse IDE, IntelliJ IDEA
  - Anatomy of a Java Application

Introduction to Java

- Java is the **most popular primary programming language**. (https://www.jetbrains.com/lp/devecosystem-2020/).
- Java has become the language of choice for implementing Internet-based applications and software for devices that communicate over a network.
- There are now billions of Java-enabled mobile phones and handheld devices.
  - Java is used in developing **Android** applications.
  - Java is used in developing **IoT** applications.

Most Popular Web Sites are using Java



Back-end (Server-side) table in most popular websites

Websites	ASP.NET	C	C++	D	Erlang	Go	Hack	Java	JavaScript	Perl	PHP	Python	Ruby	Scala	Xhp
Google.com	No	Yes	Yes	No	No	Yes	No	Yes	No	No	No	Yes	No	No	No
YouTube.com	No	Yes	Yes	No	No	Yes	No	Yes	No	No	No	Yes	No	No	No
Facebook.com	No	No	Yes	Yes	Yes	No	Yes	Yes	No	No	Yes	Yes	No	No	Yes
Yahoo	No	No	No	No	No	No	No	No	Yes	No	Yes	No	No	No	No
Amazon.com	No	No	Yes	No	No	No	No	Yes	No	Yes	No	No	No	No	No
Wikipedia.org	No	No	No	No	No	No	No	No	No	No	Yes	No	No	No	No
Twitter.com	No	No	Yes	No	No	No	No	Yes	No	No	No	No	Yes	Yes	No
Bing	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No
eBay.com	No	No	No	No	No	No	No	Yes	Yes	No	No	No	No	No	No
MSN.com	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No
Microsoft															
Linkedin.com	No	No	No	No	No	No	No	Yes	Yes	No	No	No	No	Yes	No
Pinterest												Yes			
Ask.com															
WordPress.com	No	No	No	No	No	No	No	No	Yes	No	Yes	No	No	No	No

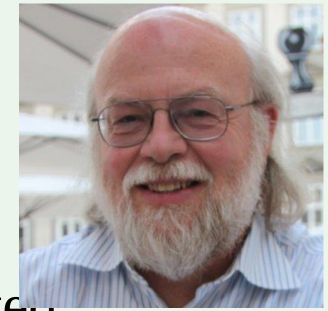
https://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites

Java Editions

- Several Java Editions are available and each of them aims at different use cases:
  - **Java Standard Edition** (Java SE 14.0. 1 is the latest release of Java SE Platform.) - contains the capabilities needed to develop desktop and server applications.
  - **Java Enterprise Edition** (Java EE) - geared toward developing large-scale, distributed networking applications and web-based applications.
  - **Java Micro Edition** (Java ME Embedded) - geared toward developing applications for resource-constrained embedded devices, such as Smartwatches, MP3 players, television set-top boxes, smart meters, etc.

History of Java

- In **1991** Sun Microsystems funded an internal corporate research project led by **James Gosling**.
- The main goal was to create a new language that would allow consumer electronic devices to communicate with each other.



- The project resulted in a C++ -based object-oriented programming language that Sun called **Java**.
  - Key goal of Java is to be able to **write programs that will run on a great variety of computer systems and computer-controlled devices**.
  - This is sometimes called "**write once, run anywhere**."

History of Java

- 1993
  - The web exploded in popularity
  - Sun saw the potential of **using Java to add dynamic content to web pages**.
  - Java drew the attention of the business community because of the phenomenal interest in the web.
- May 23, **1995 -** The **first public release of Java**
- Java is **used to develop large-scale enterprise applications**, to **enhance the functionality of web servers**, to provide **applications for consumer devices** and for many other purposes.

Java Class Libraries

- Java programs consist of pieces called classes.
- Classes include methods that perform tasks and return information when the tasks complete.
- Java class libraries
  - Rich collections of existing classes
  - Also known as the Java APIs (Application Programming Interfaces)
- Two aspects to learning the Java “world.”
  - The Java language it-self
  - The classes in the extensive Java class libraries

Typical Java Development Environment

- Java program development and execution cycle (illustrated in Fig. 1.1).

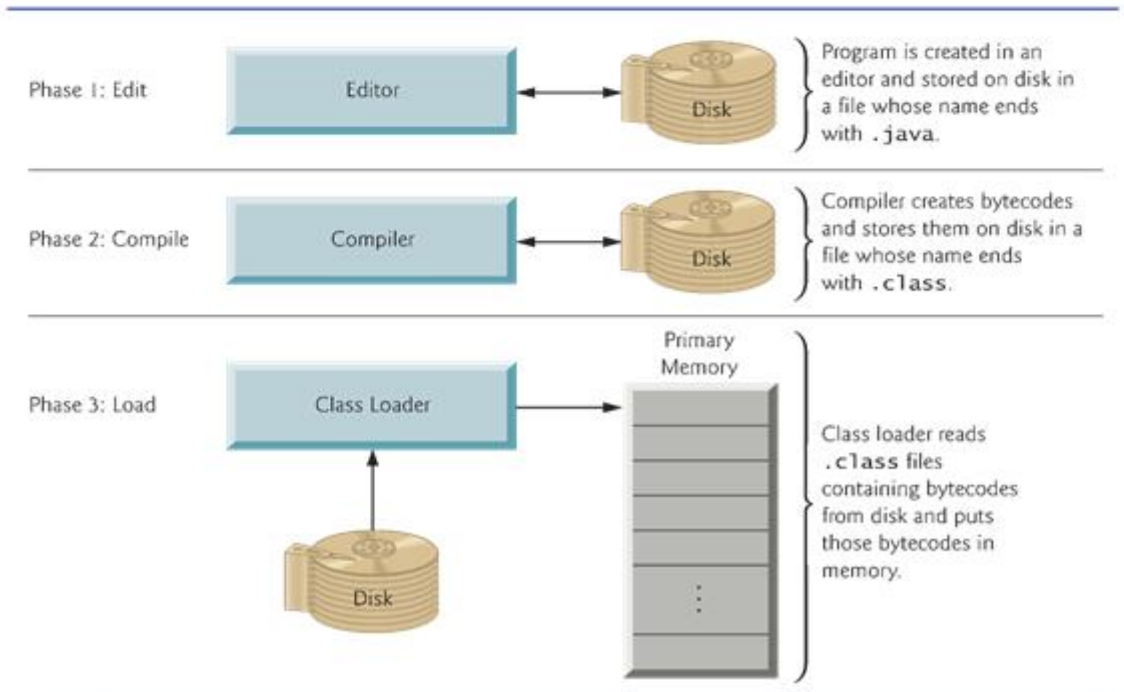

Phase 1: Edit
Editor
Disk
Program is created in an editor and stored on disk in a file whose name ends with .java.
Phase 2: Compile
Compiler
Disk
Compiler creates bytecodes and stores them on disk in a file whose name ends with .class.
Phase 3: Load
Class Loader
Primary Memory
Disk
Class loader reads .class files containing bytecodes from disk and puts those bytecodes in memory.

Fig. 1.1 | Typical Java development environment. (Part 1 of 2.)

Typical Java Development Environment

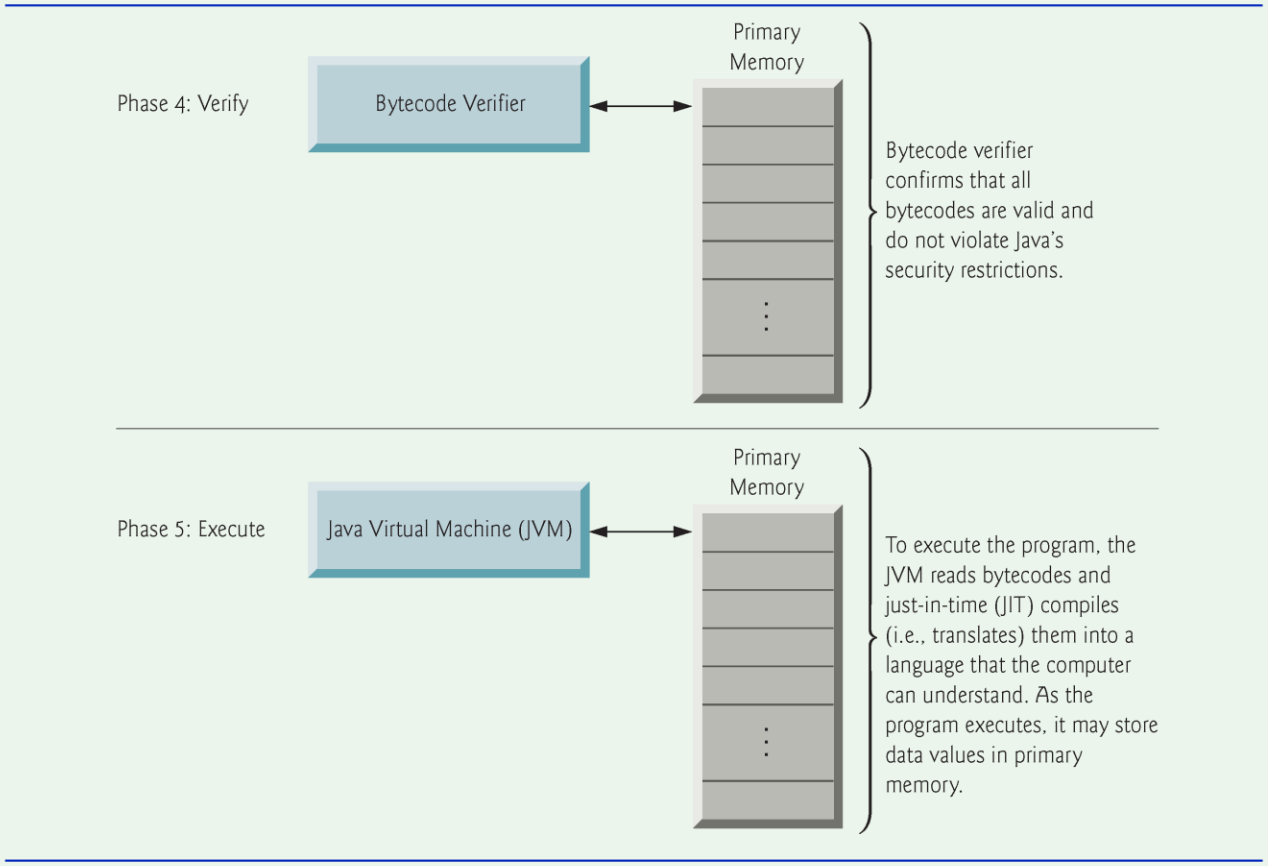
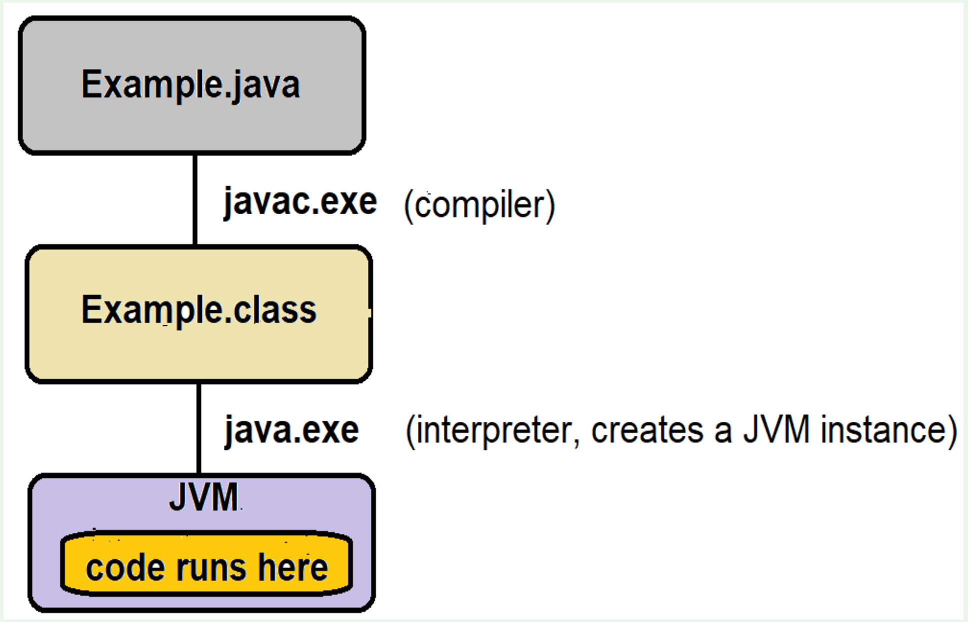

Phase 4: Verify
Bytecode Verifier
Primary Memory
Bytecode verifier confirms that all bytecodes are valid and do not violate Java's security restrictions.
Phase 5: Execute
Java Virtual Machine (JVM)
Primary Memory
To execute the program, the JVM reads bytecodes and just-in-time (JIT) compiles (i.e., translates) them into a language that the computer can understand. As the program executes, it may store data values in primary memory.

Fig. 1.1 | Typical Java development environment. (Part 2 of 2.)

Typical Java Development Environment (Cont.)

- Java programs normally go through five phases
  - **edit**
  - **compile**
  - load
  - verify
  - **execute**



Typical Java Development Environment (Cont.)

- Phase 1 consists of editing a file with an editor program *(normally known simply as an editor)*.
  - Type a Java program (source code) using the editor
  - Make any necessary corrections
  - Save the program
    - A file name ending with the .java extension indicates that the file contains Java source code.
  - **Linux** editors: `vi` and `emacs`.
  - **Windows** editors: Notepad, EditPlus (`www.editplus.com`), TextPad (`www.textpad.com`) and jEdit (`www.jedit.org`).

Typical Java Development Environment (Cont.)

- Integrated development environments (IDEs)
  - Provide tools that support the software-development process, including editors for writing and editing programs and debuggers for locating logic errors—errors that cause programs to execute incorrectly.
- Popular IDEs
  - **Eclipse** (www.eclipse.org)
  - **IntelliJ IDEA** (www.jetbrains.com)
  - **NetBeans** (www.netbeans.org)

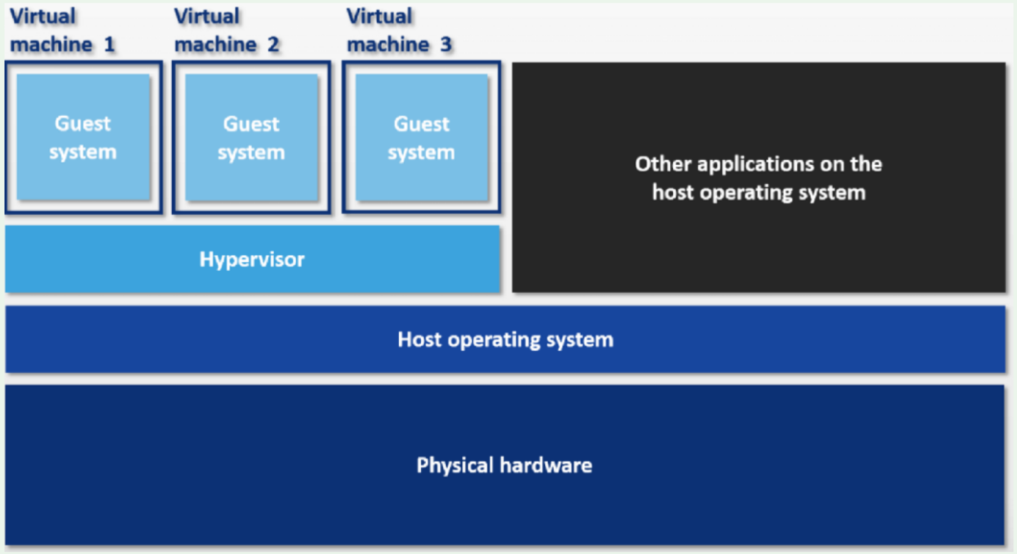
Typical Java Development Environment (Cont.)

- Phase 2
  - Use the command javac (the Java compiler) to compile a program. For example, to compile a program called `Welcome.java`, you'd type

    ```
    javac Welcome.java
    ```
  - If the program compiles, the compiler produces a .class file called `Welcome.class` that contains the compiled version of the program.

Typical Java Development Environment (Cont.)

- Java compiler **translates Java source code into** bytecodes that represent the tasks to execute.
- Bytecodes are executed by the Java Virtual Machine (JVM) - a part of the JDK and the foundation of the Java platform.
- Virtual machine (VM) - a software application that simulates a computer:



Virtual Machine architecture. Source: IONOS

Virtual Machines

- **VM**s are based on computer architectures and provide functionality of a physical computer.
- **System virtual machines** (or full virtualization VMs) provide **a substitute for a real machine**.
  - Hides the underlying operating system and hardware from the programs that interact with it.
  - They provide functionality needed to **execute entire operating systems**.
- **Process virtual machines (**like **JVM)** allow programs to execute in a platform-independent environment:
  - If the **same JVM is implemented on many computer platforms**, applications that it executes can be used on all those platforms.

Typical Java Development Environment (Cont.)

- Bytecodes are **platform independent**
  - They do not depend on a particular hardware platform.
- Bytecodes are portable
  - The same bytecodes can execute on any platform containing a JVM that understands the version of Java in which the bytecodes were compiled.
- The JVM is invoked by the `java` command. For example, to execute a Java application called `Welcome`, you'd type the command

```
java Welcome
```

Typical Java Development Environment (Cont.)

- Phase 3
  - The JVM places the program in memory to execute it
    - This is known as loading.
  - Class loader takes the `.class` files containing the program’s bytecodes and transfers them to primary memory.
  - Also loads any of the `.class` files provided by Java that your program uses.
    - The `.class` files can be loaded from a disk on your system or over a network.

Typical Java Development Environment (Cont.)

- Phase 4
  - As the classes are loaded, the bytecode verifier examines their bytecodes
    - Ensures that they are **valid** and **do not violate** Java’s security restrictions.
  - Java enforces strong security to make sure that Java programs arriving over the network do not damage your files or your system (as computer viruses and worms might).

Typical Java Development Environment (Cont.)

- Phase 5
  - The JVM executes the program’s bytecodes.
  - JVM typically uses a combination of interpretation and just-in-time (JIT) compilation.
  - Analyzes the bytecodes as they are interpreted, searching for hot spots - parts of the bytecodes that execute frequently.
  - A just-in-time (JIT) compiler (the Java HotSpot compiler) translates the bytecodes into the underlying computer’s machine language.
  - When the JVM encounters these compiled parts again, the faster machine-language code executes.

Typical Java Development Environment (Cont.)

- To summarize all this, Java programs actually go through two compilation phases:
  - One in which source code is **translated into bytecodes** (for portability across computer platforms) – done by javac.exe compiler
  - A second in which, during execution, the bytecodes are **translated into machine language** for the actual computer on which the program executes – done by JIT compiler
    - Every platform has its own Java interpreter which handles platform-specific operations – java.exe.

Java Platform

- The platform consists of two essential pieces of software:
  - The Java Runtime Environment (**JRE**), which is **needed to run** Java applications – JVM is part of it.
  - The Java Development Kit (**JDK**), which is **needed to develop** those Java applications.
    - If you have installed the JDK, you should know that it comes with a JRE as well.
    - If you have not installed JDK, download and install it from this link: https://www.oracle.com/java/technologies/javase-jdk14-downloads.html

Java Application Development

- Java application programming
- Use tools from the JDK to compile and run programs.
- Videos at www.deitel.com/books/jhtp10/
  - https://www.youtube.com/watch?v=pKJMWpMKev8
  - https://www.youtube.com/watch?v=dyvEg0sJ_3M
- Help you get started with Eclipse and IntelliJ IDEA integrated development environments.

Your First Program in Java: Printing a Line of Text

- Java application
  - A computer program that executes when you use the **java** command to launch the Java Virtual Machine (JVM).

```
// Fig. 2.1: Welcome1.java
// Text-printing program.

public class Welcome1
{
  // main method begins execution of Java application
  public static void main(String[] args)
  {
    System.out.println("Welcome to Java Programming!");
  } // end method main
} // end class Welcome1
```

Your First Program in Java: Printing a Line of Text (Cont)

Commenting Your Programs

- Comments

```
// Fig. 2.1: Welcome1.java
```

  - // indicates that the line is a comment.
  - Used to document programs and improve their readability.
  - Compiler ignores comments.
  - A comment that begins with // is an end-of-line comment - it terminates at the end of the line on which it appears.
- Traditional comment, can be spread over several lines as in

```
/* This is a traditional comment. It
   can be split over multiple lines */
```

  - This type of comment begins with /* and ends with */.
  - All text between the delimiters is ignored by the compiler.

Your First Program in Java: Printing a Line of Text (Cont)

- Javadoc comments
  - Delimited by /** and */.
  - All text between the Javadoc comment delimiters is ignored by the compiler.
  - Enable you to embed program documentation directly in your programs.
  - The `javadoc` utility program (online Appendix G) reads Javadoc comments and uses them to prepare program documentation in HTML format.

Your First Program in Java: Printing a Line of Text (Cont)

- Syntax errors – The compiler detects code that violates Java’s language rules
- Eliminate all syntax errors before the application compiles properly

Your First Program in Java: Printing a Line of Text (Cont)

Using Blank Lines

- Blank lines, space characters and tabs
  - Make programs easier to read.
  - Together, they're known as white space (or whitespace).
  - White space is ignored by the compiler.

Your First Program in Java: Printing a Line of Text (Cont)

Declaring a class

- Class declaration

```
public class Welcome1
```

  - Every Java program consists of at least one class that you define.
  - `class` keyword introduces a class declaration and is immediately followed by the class name.
  - Keywords (Appendix C) are reserved for use by Java and are always spelled with all lowercase letters.

Your First Program in Java: Printing a Line of Text (Cont)

Filename for a public Class

- A public class must be placed in a file that has a filename of the form `*ClassName*.java`, so class `Welcome1` is stored in the file `Welcome1.java`.

Your First Program in Java: Printing a Line of Text (Cont)

Class Names and Identifiers

- By convention, begin with a capital letter and capitalize the first letter of each word they include (e.g., SampleClassName).
- A class name is an identifier—a series of characters consisting of letters, digits, underscores (_) and dollar signs ($) that does not begin with a digit and does not contain spaces.
- Java is case sensitive—uppercase and lowercase letters are distinct—so a1 and A1 are different (but both valid) identifiers.

Your First Program in Java: Printing a Line of Text (Cont)

Class Body

- A left brace, {, begins the body of every class declaration.
- A corresponding right brace, }, must end each class declaration.

Your First Program in Java: Printing a Line of Text (Cont)

Declaring a Method

```
public static void main( String[] args )
```

- Starting point of every Java application.
- Parentheses after the identifier main indicate that it's a program building block called a method.
- Java class declarations normally contain one or more methods.
- main must be defined as shown; otherwise, the JVM will not execute the application.
- Methods perform tasks and can return information when they complete their tasks.
- Keyword void indicates that this method will not return any information.

Your First Program in Java: Printing a Line of Text (Cont)

- Body of the method declaration
  - Enclosed in left and right braces.
- Statement

```
System.out.println("Welcome to Java Programming!");
```

  - Instructs the computer to perform an action
    - Display the characters contained between the double quotation marks.
  - Together, the quotation marks and the characters between them are a string - also known as a character string or a string literal.
  - White-space characters in strings are *not* ignored by the compiler.
  - Strings *cannot* span multiple lines of code.

Your First Program in Java: Printing a Line of Text (Cont)

- `System.out` object
  - Standard output object.
  - Allows a Java application to display information in the command window from which it executes.
- `System.out.println` method
  - Displays (or prints) a line of text in the command window.
  - The string in the parentheses the argument to the method.
  - Positions the output cursor at the beginning of the next line in the command window.
- Most statements end with a semicolon.

Your First Program in Java: Printing a Line of Text (Cont)

Compiling Your First Java Application

- Open a command window and change to the directory where the program is stored. **Make sure Java path is set properly**.
- Many operating systems use the command cd to change directories.
- To compile the program, type

```
javac Welcome1.java
```

- If the program contains no compilation errors, preceding command creates a.class file (known as the class file) containing the platform-independent Java bytecodes that represent the application.
- When we use the java command to execute the application on a given platform, these bytecodes will be translated by the JVM into instructions that are understood by the underlying operating system.

Your First Program in Java: Printing a Line of Text (Cont)

Executing the Welcome1 Application

- To execute this program in a command window, change to the directory containing Welcome1.java - C:\examples\ch02\fig02_01 on Microsoft Windows or ~/Documents/examples/ch02/fig02_01 on Linux/OS X.
- Next, type java `Welcome1`.
- This launches the JVM, which loads the `Welcome1.class` file.
- The command *omits* the .class file-name extension; otherwise, the JVM will *not* execute the program.
- The JVM calls class `Welcome1`'s main method.

Modifying Your First Java Program

- Class Welcome2, shown in Fig. 2.3, uses two statements to produce the same output as that shown in Fig. 2.1.
- New and key features in each code listing are highlighted.
- System.out’s method print displays a string.
- Unlike println, print does not position the output cursor at the beginning of the next line in the command window.
  - The next character the program displays will appear immediately after the last character that print displays.

Modifying Your First Java Program

```
// Fig. 2.3: Welcome2.java
// Printing a line of text with multiple statements.

public class Welcome2
{
   // main method begins execution of Java application
   public static void main(String[] args)
   {
      System.out.print("Welcome to ");
      System.out.println("Java Programming!");
   } // end method main
} // end class Welcome2
```

Modifying Your First Java Program

- Newline characters indicate to System.out's print and println methods when to position the output cursor at the beginning of the next line in the command window.
- Newline characters are whitespace characters.
- The backslash (\) is called an escape character.
  - Indicates a "special character"
- Backslash is combined with the next character to form an escape sequence - \n represents the newline character.
- Complete list of escape sequences

  http://docs.oracle.com/javase/specs/jls/se7/html/

  jls-3.html#jls-3.10.6.

Modifying Your First Java Program

```
// Fig. 2.4: Welcome3.java
// Printing multiple lines with a single statement.

public class Welcome3
{
   // main method begins execution of Java application
   public static void main(String[] args)
   {
      System.out.println("Welcome\nto\nJava\nProgramming!");
   } // end method main
} // end class Welcome3
```

Displaying Text with printf

- System.out.printf method
  - f means “formatted”
  - displays *formatted* data
- Multiple method arguments are placed in a comma-separated list.
- Method printf’s first argument is a format string
  - May consist of fixed text and format specifiers.
  - Fixed text is output as it would be by print or println.
  - Each format specifier is a placeholder for a value and **specifies the type of data to output**.
- Format specifiers begin with a percent sign (%) and are followed by a character that represents the data type.
- Format specifier %s is a placeholder for a string.

Displaying Text with printf

```
// Fig. 2.6: Welcome4.java
// Displaying multiple lines with method System.out.printf.

public class Welcome4
{
   // main method begins execution of Java application
   public static void main(String[] args)
   {
      System.out.printf("%s%n%s%n",
         "Welcome to", "Java Programming!");
   } // end method main
} // end class Welcome4
```

Another Application: Adding Integers

- Integers
  - Whole numbers, like –22, 7, 0 and 1024
- Programs remember numbers and other data in the computer’s memory and access that data through program elements called variables.
- The program of Fig. 2.7 demonstrates these concepts.

Another Application: Adding Integers

```
// Fig. 2.7: Addition.java
// Addition program that displays the sum of two numbers.
import java.util.Scanner; // program uses class Scanner

public class Addition
{
   // main method begins execution of Java application
   public static void main(String[] args)
   {
      // create a Scanner to obtain input from the command
window
      Scanner input = new Scanner(System.in);

      int number1; // first number to add
      int number2; // second number to add
      int sum; // sum of number1 and number2

      System.out.print("Enter first integer: "); // prompt
      number1 = input.nextInt(); // read first number from
user

      System.out.print("Enter second integer: "); // prompt
      number2 = input.nextInt(); // read second number
from user

      sum = number1 + number2; // add numbers, then
store total in sum

      System.out.printf("Sum is %d%n", sum); // display
sum
   } // end method main
} // end class Addition
```

import Declarations

- Helps the compiler locate a class that is used in this program.
- Rich set of predefined classes that you can reuse rather than “reinventing the wheel.”
- Classes are grouped into *packages*—*named groups of related classes*—and are collectively referred to as the Java class library, or the Java Application Programming Interface (Java API).
- You use import declarations to identify the predefined classes used in a Java program.
- Place them before class declaration

Declaring and Creating a Scanner to Obtain User Input from the Keyboard

- Variable declaration statement

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

  - Specifies the name (input) and type (Scanner) of a variable that is used in this program.
- Variable
  - A location in the computer's memory where a value can be stored for use later in a program.
  - *Must* be declared with a name and a type before they can be used.
  - A variable's *name* enables the program to access the value of the variable in memory.
  - The name can be any valid identifier.
  - A variable's type specifies what kind of information is stored at that location in memory.

Another Application: Adding Integers (Cont.)

- **Scanner**
- Enables a program to read data for use in a program.
- Data can come from many sources, such as the user at the keyboard or a file on disk.
- Before using a Scanner, you must create it and specify the source of the data.
- The equals sign (=) in a declaration indicates that the variable should be initialized (i.e., prepared for use in the program) with the result of the expression to the right of the equals sign.
- The new keyword creates an object.
- Standard input object, **System.in**, enables applications to read bytes of data typed by the user.
- Scanner object translates these bytes into types that can be used in a program.

Declaring Variables to Store Integers

- Variable declaration statements

```
int number1; // first number to add
int number2; // second number to add
int sum; // sum of number1 and number2
```

  declare that variables number1, number2 and sum hold data of type int
  - They can hold integer.
  - Range of values for an int is –2,147,483,648 to +2,147,483,647.
  - The int values you use in a program may not contain commas.
- Several variables of the same type may be declared in one declaration with the variable names separated by commas.
- Use **camel case naming** convention

Prompting the User for Input

- Prompt
  - Output statement that directs the user to take a specific action.
- Class System
  - Part of package java.lang.
  - Class System is not imported with an import declaration at the beginning of the program.

Obtaining an int as Input from the User

- Scanner method nextInt:

  ```
  number1 = input.nextInt(); // read first number from user
  ```

  - Obtains an integer from the user at the keyboard.
  - Program *waits* for the user to type the number and press the *Enter* key to submit the number to the program.
- The result of the call to method nextInt is placed in variable number1 by using the assignment operator, =.
  - "number1 *gets* the value of input.nextInt()."
  - Operator = is called a binary operator- it has *two* operands.
  - Everything to the *right* of the assignment operator, =, is always evaluated *before* the assignment is performed.

Another Application: Adding Integers (Cont.)

- Arithmetic

  sum = number1 + number2; // add numbers then store total in sum

  - Assignment statement that calculates the sum of the variables number1 and number2 then assigns the result to variable sum by using the assignment operator, =.
  - “sum *gets* the value of number1 + number2.”
  - Portions of statements that contain calculations are called expressions.
  - An expression is any portion of a statement that has a value associated with it.

Displaying the Result of the Calculation

- Integer formatted output

```
System.out.printf( "Sum is %d%n", sum );
```

  - Format specifier %d is a *placeholder* for an int value
  - The letter d stands for “decimal integer.”

Eclipse example

- Run Addition.java in Eclipse
- Modify the application to allow the user to enter the name.

References

- Textbook
- Java documentation, https://docs.oracle.com/en/java/
- https://docs.oracle.com/javase/8/docs/technotes/guides/index.html
- https://en.wikibooks.org/wiki/Java_Programming/The_Java_Platform