Introduction to Java

Chapter 1 & 2, Core Java, Volume I

Java is a … Programming Language

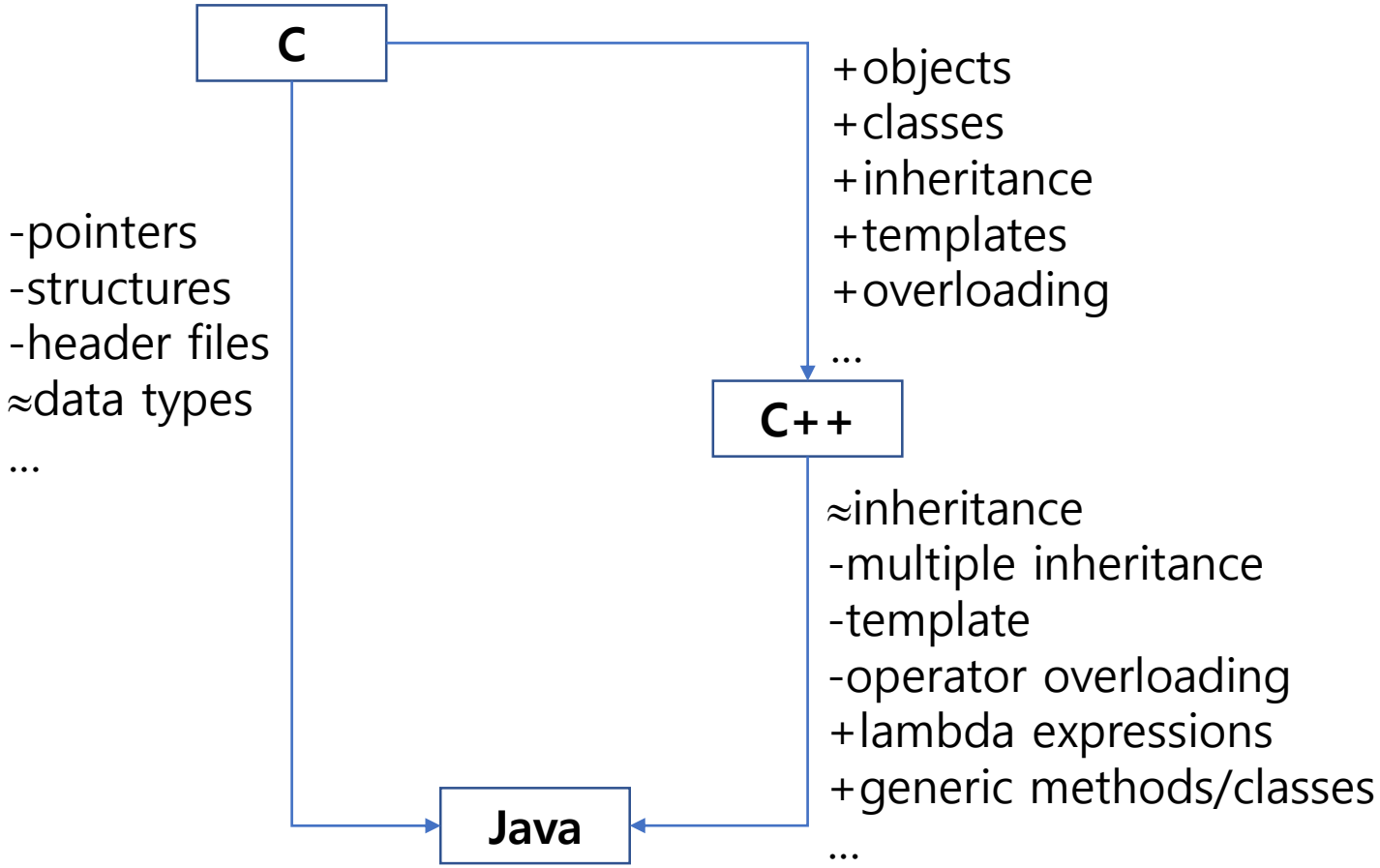
- Simple
- Object-Oriented
- Distributed
- Robust
- Secure
- Architecture-neutral
- Portable
- Interpreted
- High-performance
- Multithreaded
- Dynamic

The "White Paper" Buzzwords

❖Note: https://www.oracle.com/technetwork/java/index-136113.html

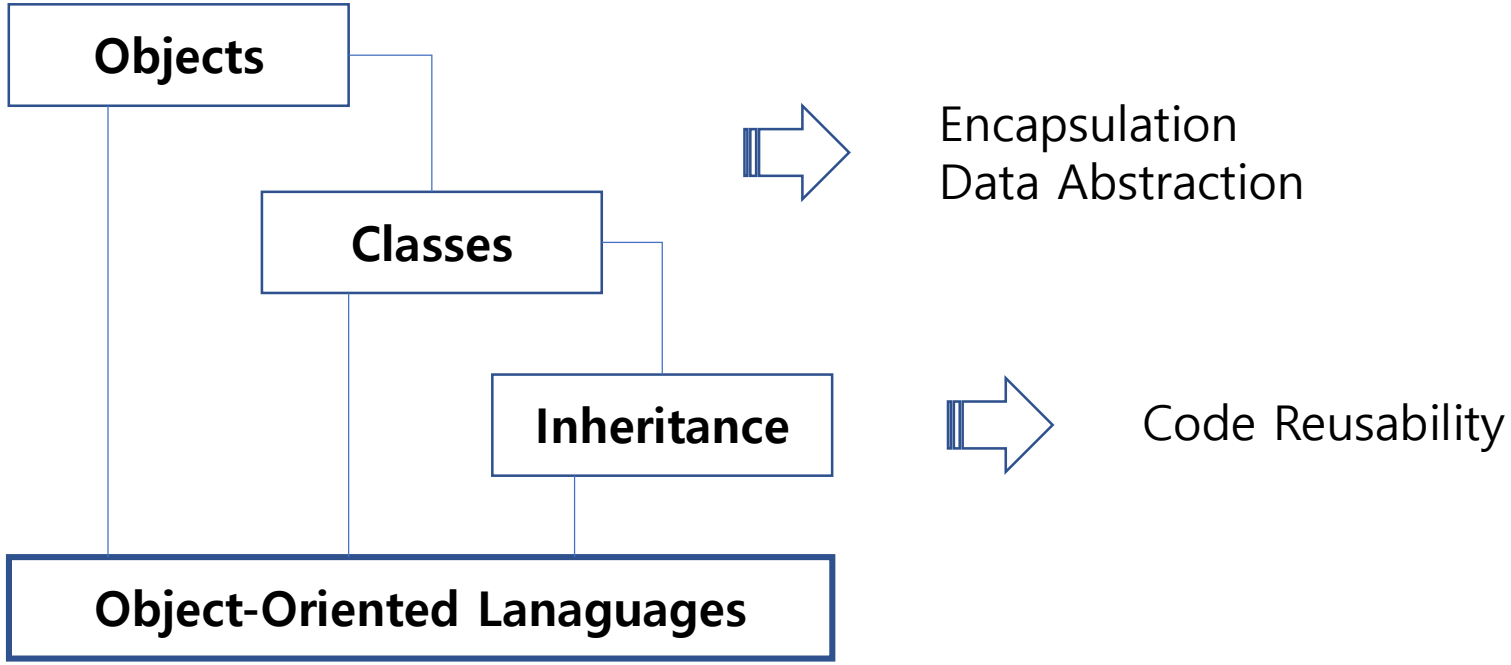
Java is a … Programming Language

- Simple
  - The syntax of Java is indeed a cleaned-up version of C and C++



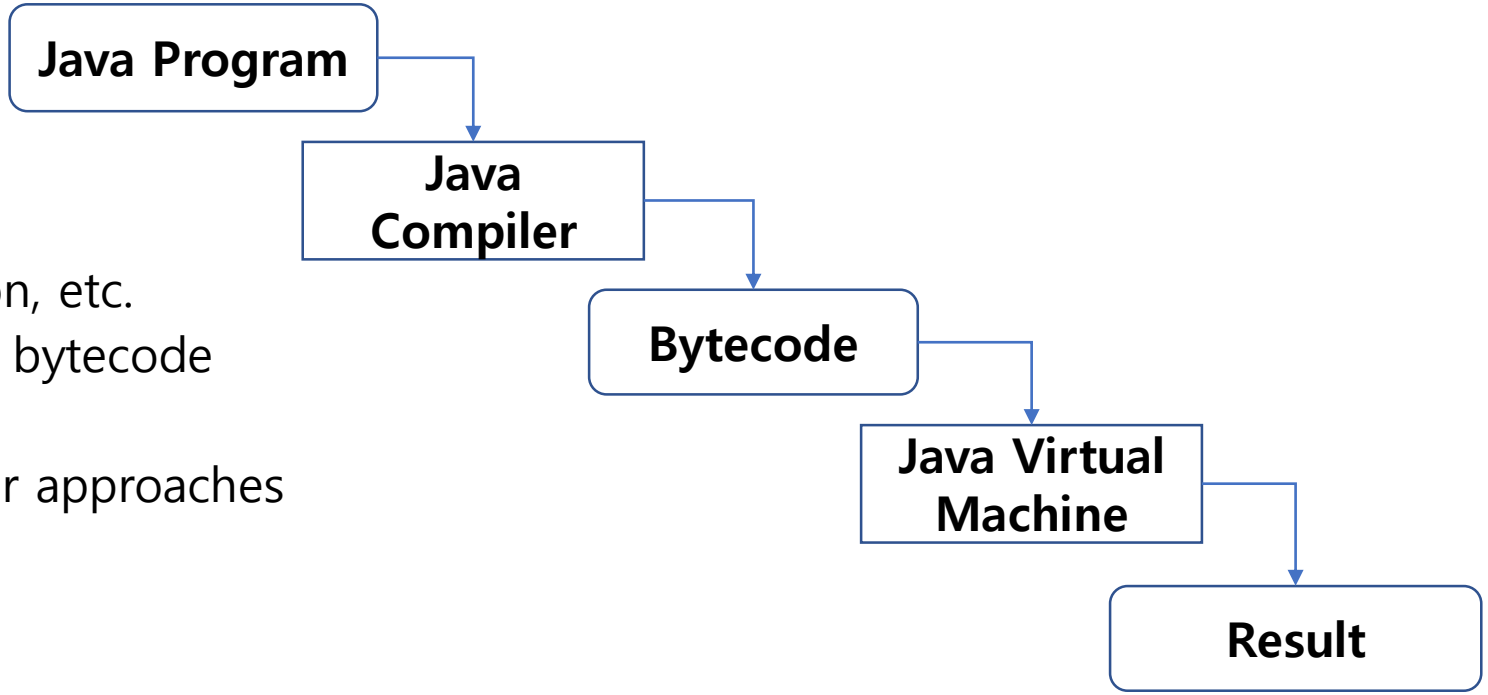
Java is a … Programming Language

- Object-Oriented
  - Programming paradigm based on the concepts of objects
  - An object contains its data (state) and behavior (methods or functions)
  - The execution of program is viewed as a collaboration of objects
- Three Key Elements



Java is a … Programming Language

- Architecture-Neutral
  - "Write-once, run-everywhere"
  - Java Virtual Machine
    - Bytecode
    - Interpreter



- Interpreted
  - Not like Basic, Python, etc.
  - Interprets the whole bytecode
  - Hybird approach
  - Slower than compiler approaches

Why Use Java?

- Java works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.)
- It is one of the most popular programming language in the world(TIOBE ratings)
- It is easy to learn and simple to use
- It is open-source and free
- It is also a programming platform including language, environment, and huge library
- It has a huge community support (tens of millions of developers)
- It is used for various applications:
  - Desktop applications
  - Mobile applications (specially Android apps)
  - Web applications
  - Games
  - Etc.

❖ Note: https://www.w3schools.com/java/java_intro.asp

A Short History of Java

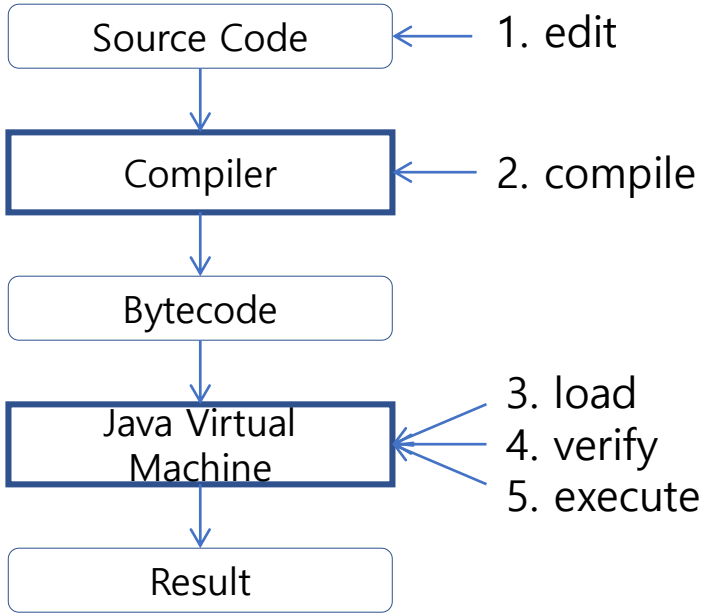
- 1991: James Gosling worked on “Project Green”, a system for consumer devices.
- He designed a programming language, originally called “Oak”.
- That name was trademarked, so it was renamed to “Java”.
- 1992: The first project was released, a TV switchbox called “*7”.
- Nobody cared, and the project was renamed “First Person, Inc.”
- 1994: Still nobody cared, and Gosling realized that they could build a “really cool browser…architecture-neutral, real-time, reliable, secure.”
- 1995: HotJava was released.
- 1996: Java 1.0 was released.
- 1998: Java grows up with Java 2 release. SE, ME, EE editions.

Java Versions

Version	Year	New Language Features	Number of Classes and Interfaces
1.0	1996	The language itself	211
1.1	1997	Inner classes	477
1.2	1998	Swing GUI framework, the strictfp modifier	1,524
1.3	2000	None	1,840
1.4	2002	Assertions	2,723
5.0	2004	Generic classes, "for each" loop, varargs, autoboxing, metadata, enumerations, static import	3,279
6	2006	None	3,793
7	2011	Switch with strings, diamond operator, binary literals, exception handling enhancements	4,024
8	2014	Lambda expressions, interfaces with default methods, stream and date/time libraries	4,240
9	2017	Modules, miscellaneous language and library	6,005

The Java Programming Environment

- Java Develement and Execution Phases
  1. Edit
  2. Compiler
  3. Load
  4. Verify
  5. Execute



- Two Types of Programming Environment
  - Using Command Line Tools
  - Using Integrated Development Environment (e.g. Eclipse)

Installing Java Development Kit

- Download the Java Software Development Kit (JDK) from http://www.oracle.com/technetwork/java/javase/downloads.
  - Navigating the Oracle site requires mastery of some Java jargon.
  - Also download and unzip the documentation of the application programming interface (API).
- Install the JDK, using the provided installer.
- Set the PATH environment variable.
- Download the sample code for these lessons.
- Install an integrated development environment (IDE) such as Eclipse, NetBeans, or IntelliJ.

Using the Command-Line Tools

- Open a terminal window.
- Change to the directory where you unzipped the sample code for these lessons.
- Run the following commands
  - cd corejava/v1ch02/Welcome
  - javac Welcome.java
  - java Welcome
- Congratulations!

  You have just compiled and run your first Java program.

```
Terminal
~$ cd corejava/v1ch02/Welcome
~/corejava/v1ch02/Welcome$ javac Welcome.java
~/corejava/v1ch02/Welcome$ java Welcome
Welcome to Core Java!
======================
~/corejava/v1ch02/Welcome$
```

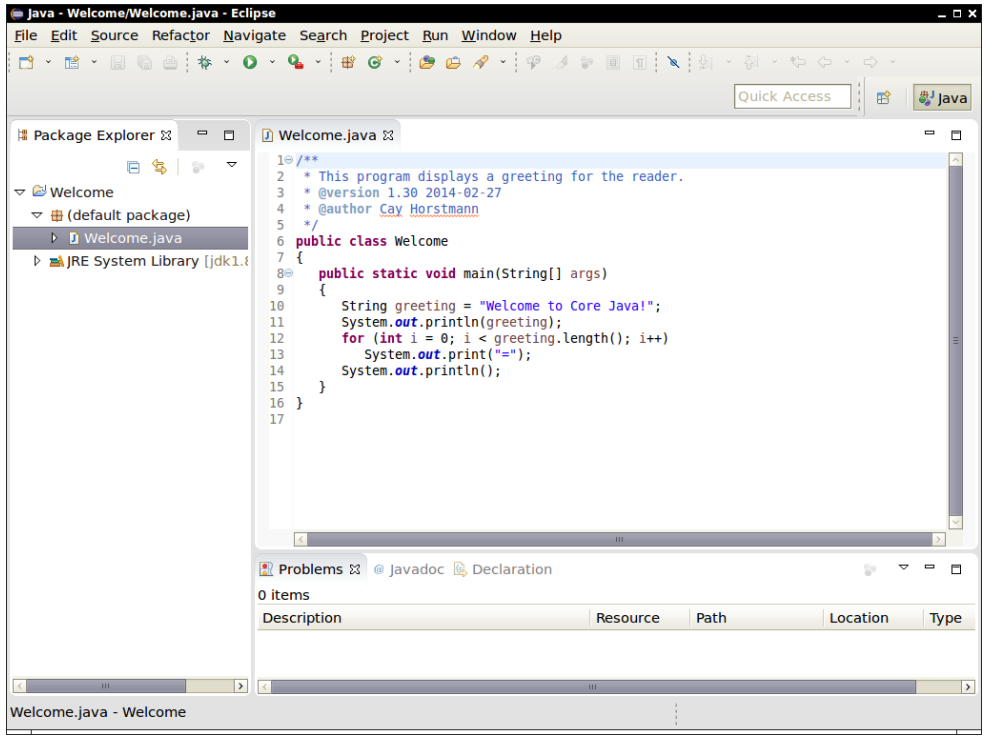
Using the Command-Line Tools

- Welcome.java

```
/**
This program displays a greeting for the reader.
author Cay Horst@version 1.30 2014-02-27
* @mann */
public class Welcome
{
    public static void main(String[] args)
    {
        String greeting = "Welcome to Core Java!";
        System.out.println(greeting);
        for (int i = 0; i < greeting.length(); i++)
            System.out.print("=");
       System.out.println();
    }
}
```

Using an Integrated Development Environment

- For day-to-day work, integrated development environments are more convenient.
- Excellent choices are the freely available Eclipse, NetBeans, and IntelliJ IDEA.
- Eclipse will be used in this lecture
  - Download from http://eclipse.org.
  - Unzip, then launch Eclipse.
  - Select File → New → Project from the menu.
  - Make a Java project with existing sources at corejava/v1ch02/Welcome.
  - Run the program.
  - Introduce an error: Change String to string
    - Note the wiggly underline.
    - Note the error in the Problems pane.

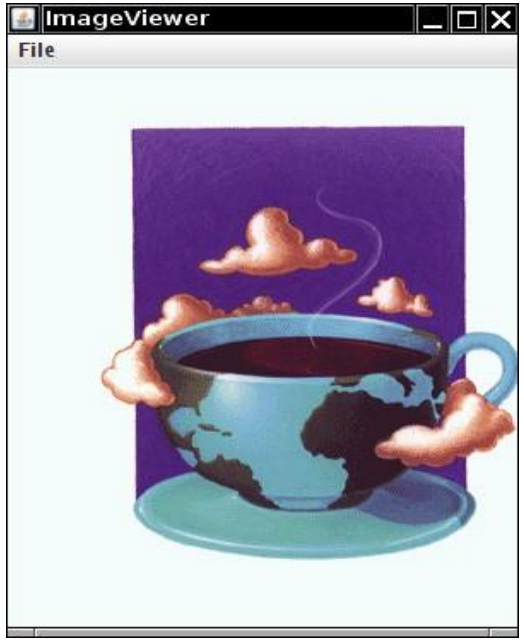


Running a Graphical Application

- The console application was rather basic.
- It is easy in Java to write applications with a graphical user interface(GUI).
- Example: Image viewer.
- Do this in Eclipse or on the command line:

```
cd corejava/v1ch02/ImageViewer
javac ImageViewer.java
java ImageViewer
```

- Select File → Open from the menu and look for an image file to open.
- Select File → Exit from the menu.



Running a Graphical Application

- **ImageViewer.java**

```
import java.awt.*;
import java.io.*;
import javax.swing.*;
public class ImageViewer
{
   public static void main(String[] args)
   {
      EventQueue.invokeLater(() -> {
         JFrame frame = new ImageViewerFrame();
         frame.setTitle("ImageViewer");
         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         frame.setVisible(true);
      });
   }
}
class ImageViewerFrame extends JFrame
{
   private JLabel label;
   private JFileChooser chooser;
   private static final int DEFAULT_WIDTH = 300;
   private static final int DEFAULT_HEIGHT = 400;
   public ImageViewerFrame()
   {
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

      // use a label to display the images
      label = new JLabel();
      add(label);

      // set up the file chooser
      chooser = new JFileChooser();
      chooser.setCurrentDirectory(new File("."));

      // set up the menu bar
      JMenuBar menuBar = new JMenuBar();
      setJMenuBar(menuBar);

      JMenu menu = new JMenu("File");
      menuBar.add(menu);

      JMenuItem openItem = new JMenuItem("Open");
      menu.add(openItem);
      openItem.addActionListener(event -> {
         // show file chooser dialog
            int result = chooser.showOpenDialog(null);

            // if file selected, set it as icon of the label
            if (result == JFileChooser.APPROVE_OPTION)
            {
               String name = chooser.getSelectedFile().getPath();
               label.setIcon(new ImageIcon(name));
            }
         });

      JMenuItem exitItem = new JMenuItem("Exit");
      menu.add(exitItem);
      exitItem.addActionListener(event -> System.exit(0));
   }
}
```