JAVA PROGRAMMING

# Week 1: Introduction to Java

Lecturer:

- Hồ Tuấn Thanh, M.Sc.

# Plan

1. History
2. How Java impacted the Internet
3. Compiling and running a Java program

# Plan

1. History
2. How Java impacted the Internet
3. Compiling and running a Java program

# Origins of Java

- Conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems in 1991.
  - Accquired by Oracle, 2010.
- Initially called “Oak” but was renamed “Java” in 1995.

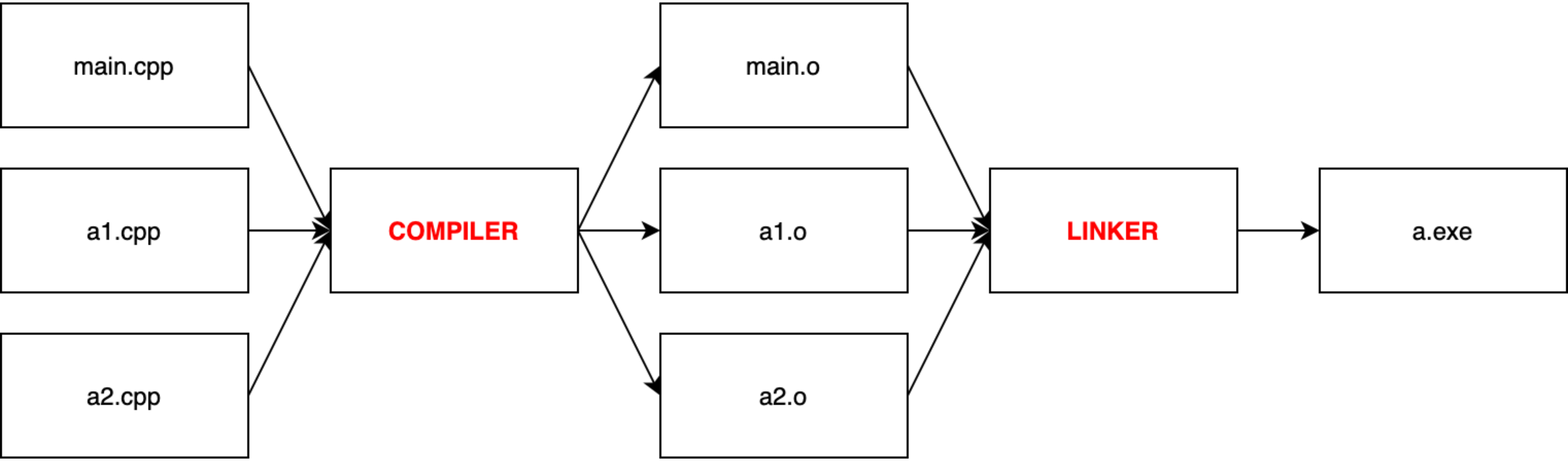
# Origins of Java

- Primary motivation:
  - A platform independent language
  - To create software to be embedded in various consumer electronic devices, such as toasters, microwave ovens, and remote controls.
- Slogan: **Write once, run anywhere**

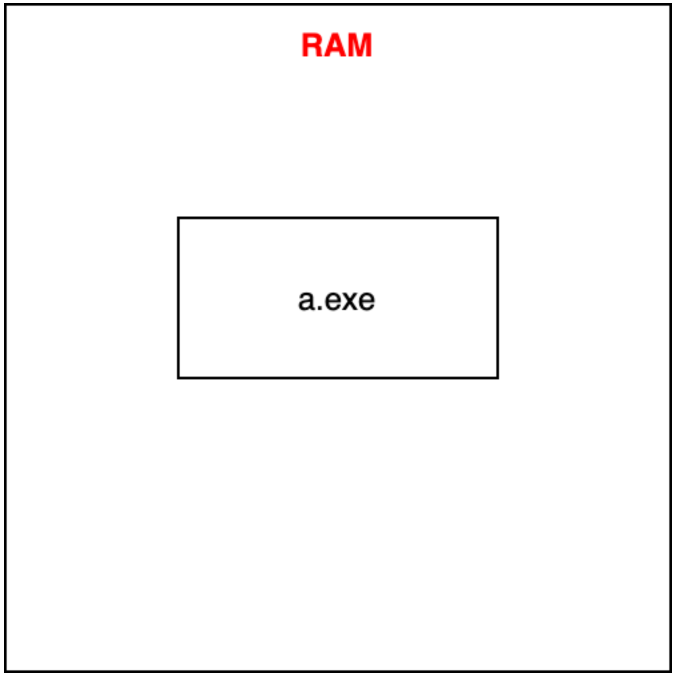
# Problems (at that time)

- Most computer languages were designed to be compiled into machine code that was targeted for a specific type of CPU.
    - For example, consider the C++ language.
- Although it was possible to compile a C++ program for just about any type of CPU, to do so required a full C++ compiler targeted for that CPU → compilers are expensive and time consuming to create.

# Problems (at that time)



# Problems (at that time)



# Problems (at that time)

- In an attempt to find a better solution, Gosling and the others worked on a portable, cross-platform language that could produce code that would run on a variety of CPUs under differing environments → Java

# Java’s Lineage: C and C++ [1]

- C and C++ are among the most important computer languages ever invented and are still in widespread use today.
- Java inherits its syntax from C. Java’s object model is adapted from C++.
- Java’s relationship to C and C++ is important for a number of reasons.
    - Many programmers were familiar with the C/C++ syntax → Java uses a similar syntax, it was relatively easy for a C/C++ programmer to learn Java.
    - Java provides a powerful, logically consistent programming environment by inheriting and building on C/C++.

# Java’s Lineage: C and C++ [2]

- Although C++ and Java are related, especially in their support for object-oriented programming, Java is not simply the “Internet version of C++”.
- Java has significant practical and philosophical differences from C++.
- Java is not an enhanced version of C++.

# Plan

1. History
2. How Java impacted the Internet
3. Compiling and running a Java program
4. Variables
5. Expression

# How Java Impacted the Internet

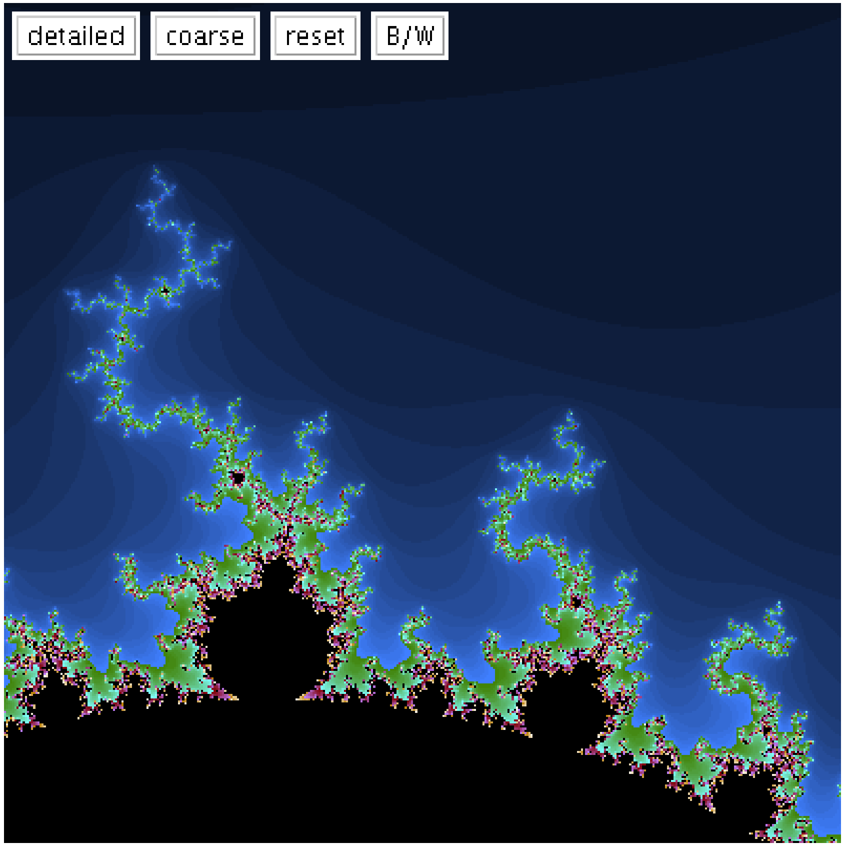
- Java had a profound effect on the Internet
  - The creation of Java acting as a catalyst that drew legions of programmers to the Web.
  - Java innovated a new type of networked program called the *applet* that changed the way the online world thought about content.
  - Java addressed some of the thorniest issues associated with the Internet: portability and security.

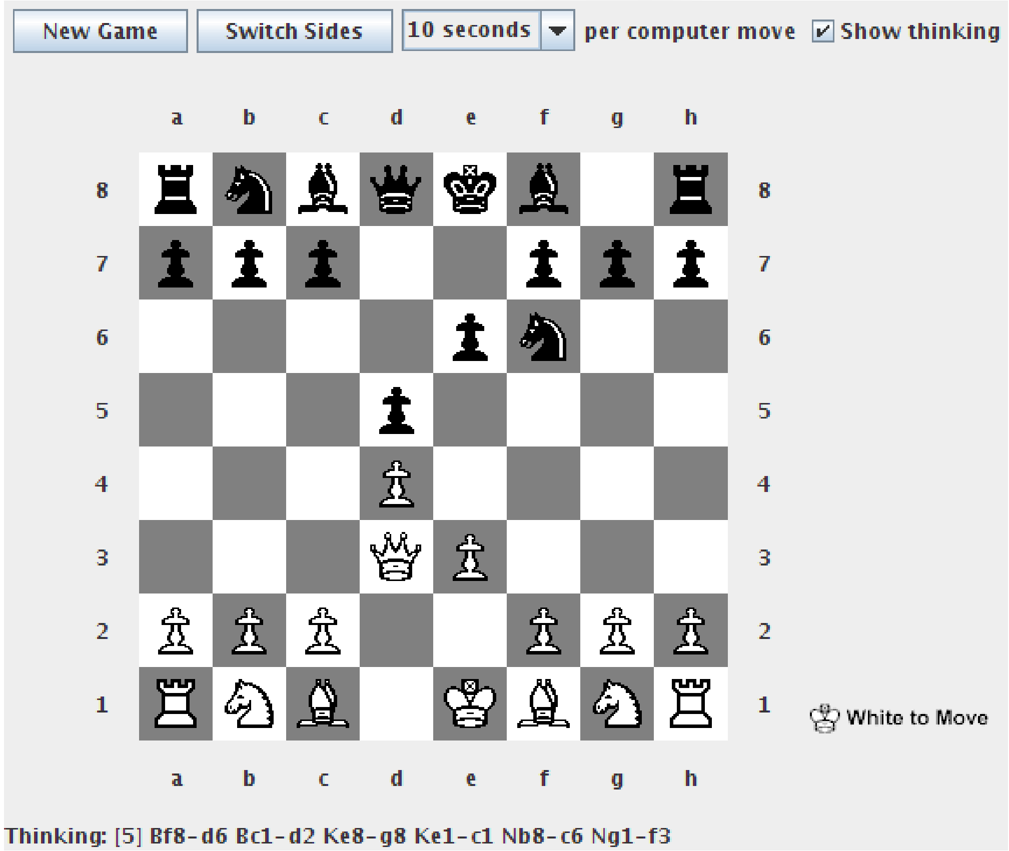
# Java Simplified Web-Based Programming

- Ability to create portable, cross-platform programs.
- Java supports for networking
  - Provide library that enabled programmers to easily write programs that accessed or made use of the Internet.
  - Provided mechanisms that enabled programs to be readily delivered over the Internet.

# Java Applets

- An *applet* is a special kind of Java program designed to be transmitted over the Internet and automatically executed inside a Java-compatible web browser.
  - If the user clicks a link that contains an applet, the applet will download and run in the browser automatically.
  - Applets: small programs, typically used to display data provided by the server, handle user input, or provide simple functions.
  - Key feature: execute locally, rather than on the server.
- The creation of the *applet* was important because it is a dynamic, self-executing program.
- In the early days of Java: *applet*s were a crucial part of Java programming.
- Applet support was removed by JDK 11.

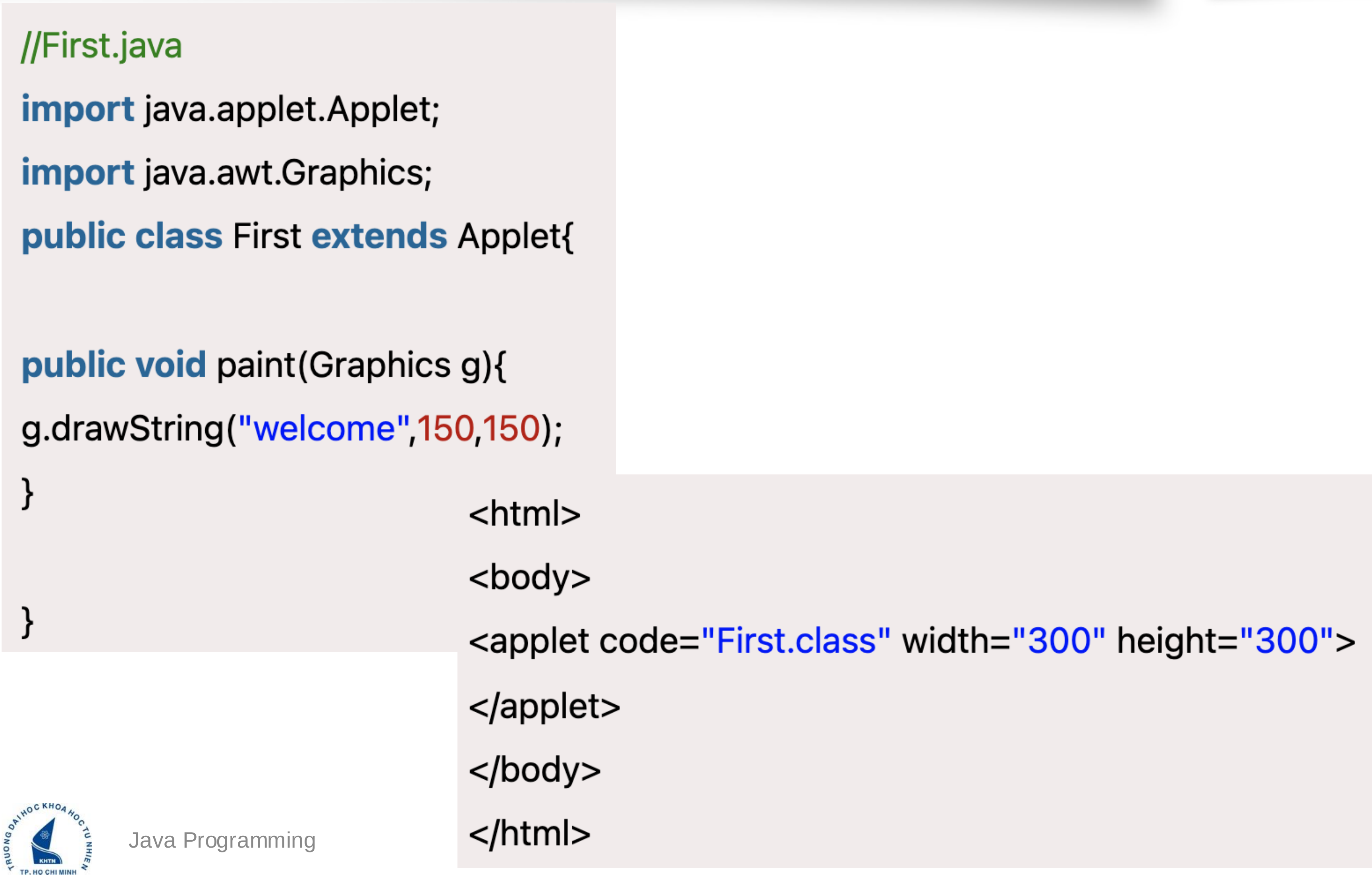




10 seconds per computer move ☑ Show thinking

White to Move

Thinking: [5] Bf8-d6 Bc1-d2 Ke8-g8 Ke1-c1 Nb8-c6 Ng1-f3



```
//First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet{

public void paint(Graphics g){
g.drawString("welcome",150,150);
}


}
```

```
<html>
<body>
<applet code="First.class" width="300" height="300">
</applet>
</body>
</html>
```

# Security

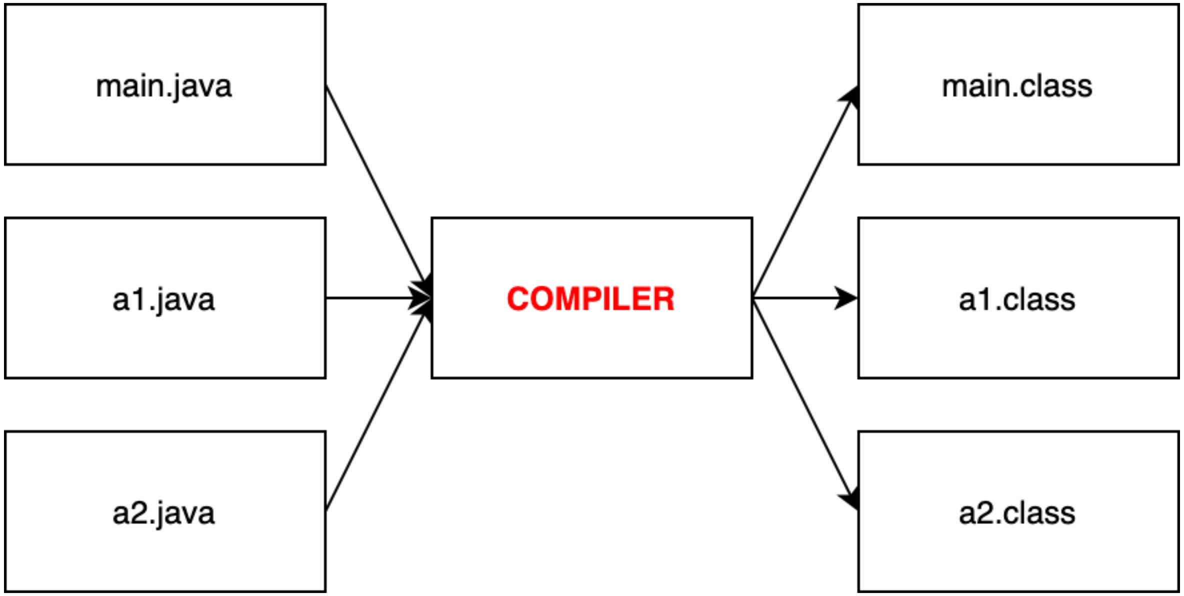
- Java enables you to confine an application to the Java execution environment and prevent it from accessing other parts of the computer.
- The ability to download an application with a high level of confidence that no harm will be done contributed significantly to Java's early success.

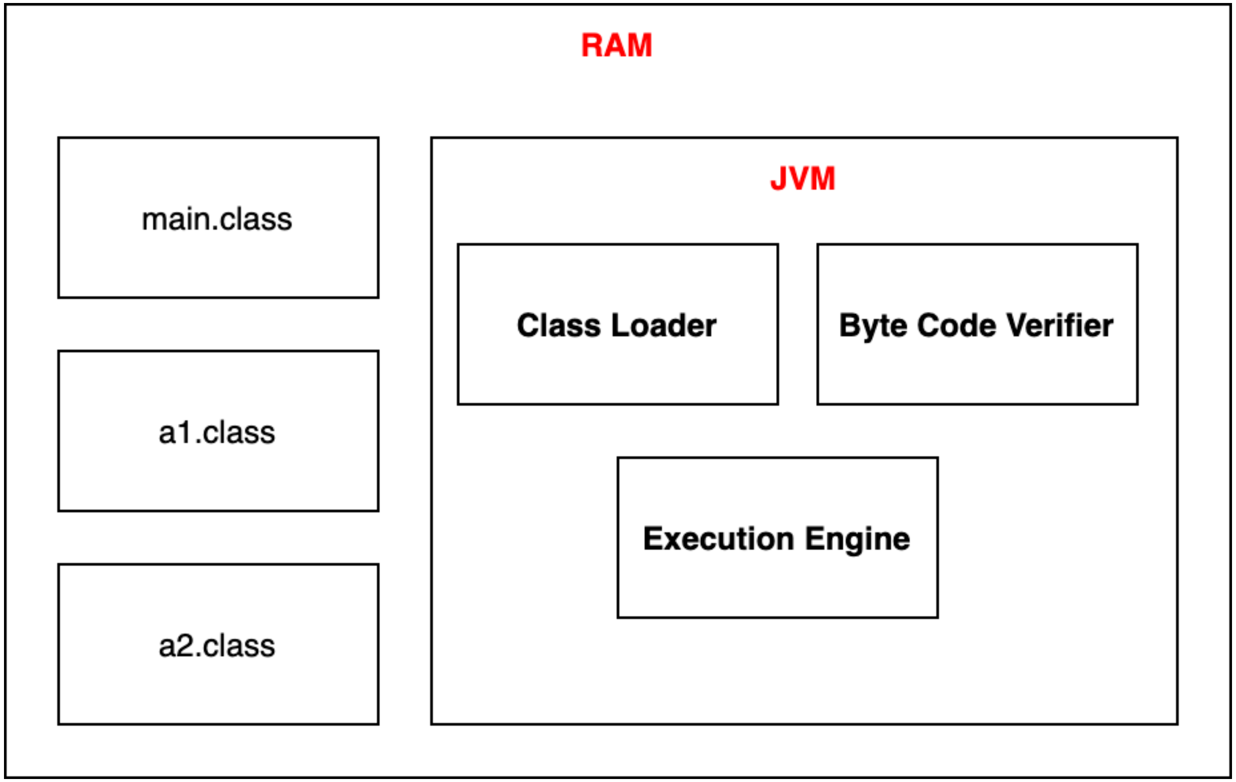
# Portability

- Java provides mechanism that allows the same application to be downloaded and executed by a wide variety of CPUs, operating systems, and browsers is required.

# Java’s Magic: The Bytecode

- The key that allowed Java to address both the security and the portability problems is that the output of a Java compiler is not executable code.
- Rather, it is bytecode.
  - Bytecode is a highly optimized set of instructions designed to be executed by what is called the Java Virtual Machine (JVM), which is part of the Java Runtime Environment (JRE).
  - The original JVM was designed as an interpreter for bytecode.
- Once a JRE exists for a given system, any Java program can run on it.
  - Although the details of the JRE will differ from platform to platform, all JREs understand the same Java bytecode.

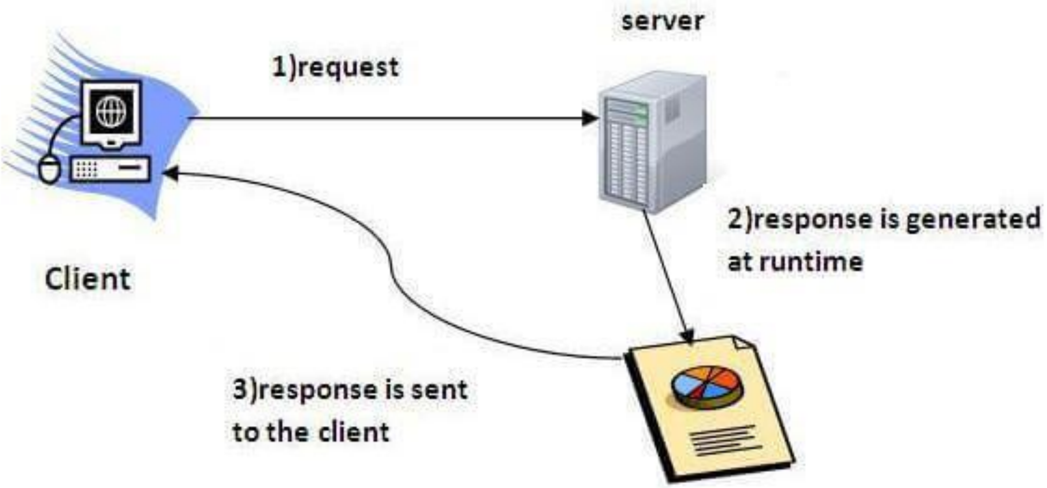

main.java
a1.java
a2.java
COMPILER
main.class
a1.class
a2.class

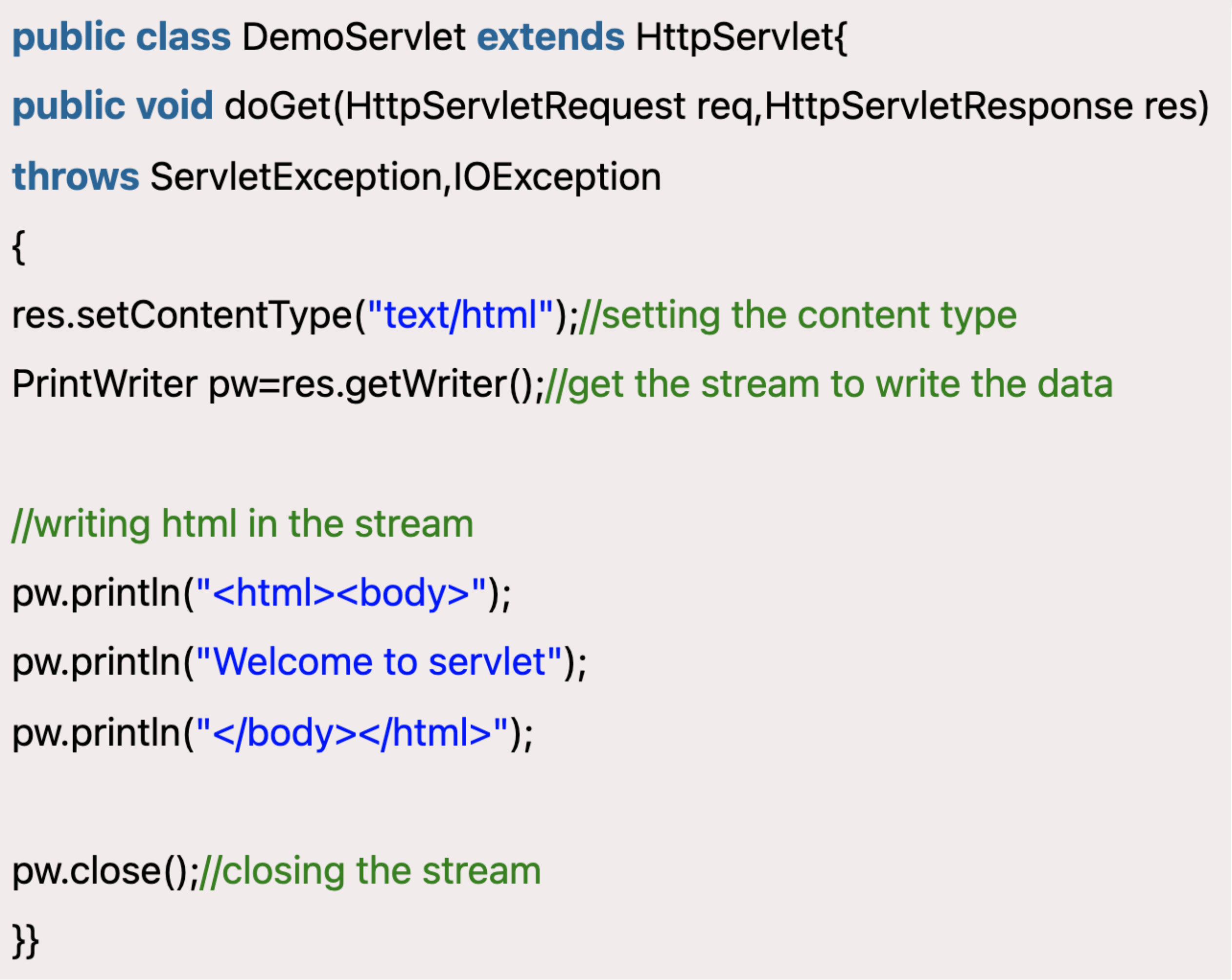

RAM
main.class
a1.class
a2.class
JVM
Class Loader
Byte Code Verifier
Execution Engine

- A Java program is executed by the JVM also helps to make it secure.
  - Because the JVM is in control, it manages program execution.
- It is possible for the JVM to create a restricted execution environment, called the sandbox, that contains the program, preventing unrestricted access to the machine. Safety is also enhanced by certain restrictions that exist in the Java language.
- Because bytecode has been highly optimized, the use of bytecode enables the JVM to execute programs faster.

# Servlet

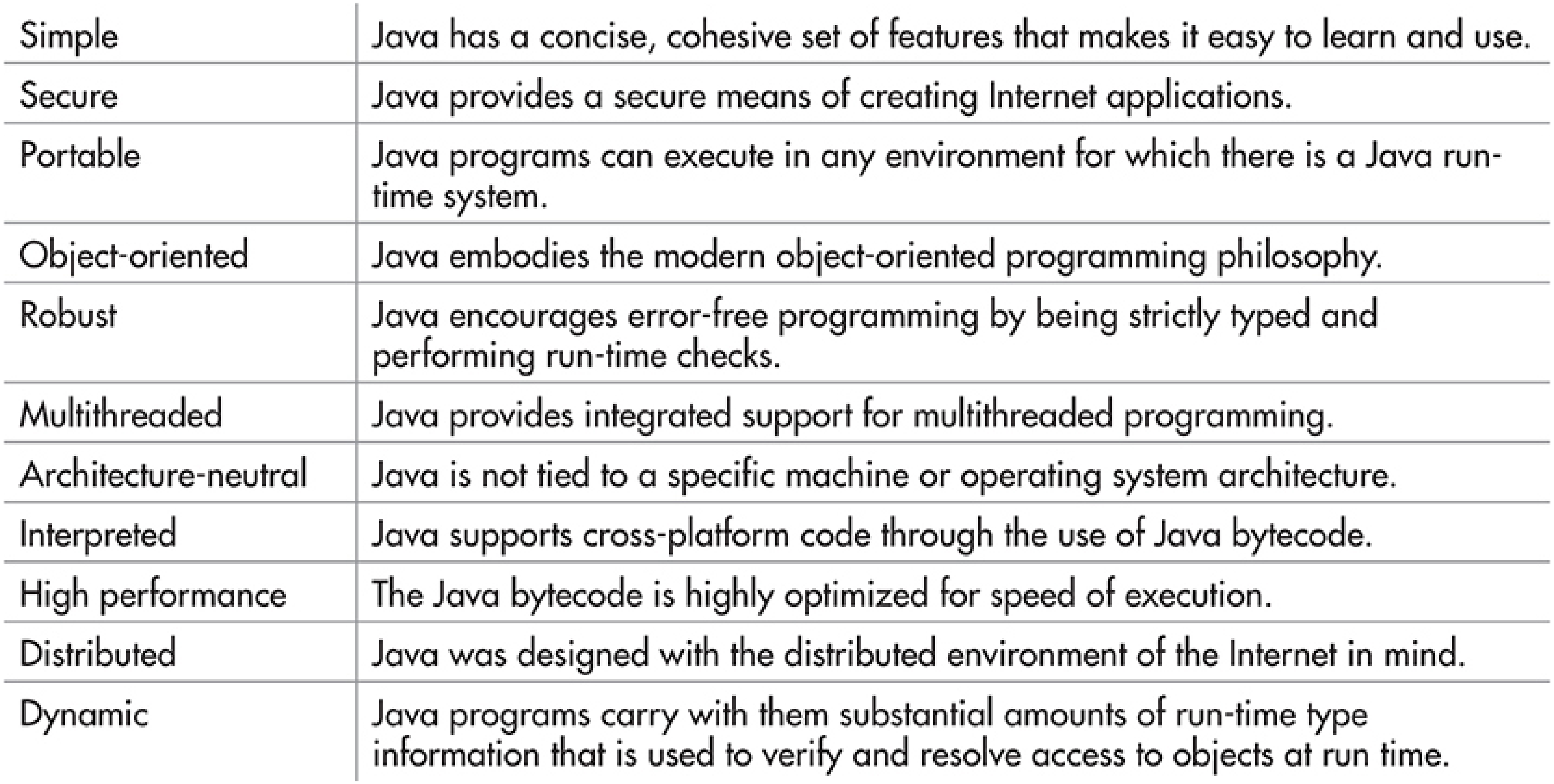
- A Java servlet is a small program that executes on a server.
- Servlets dynamically extend the functionality of a web server.
- With the advent of the servlet, Java spanned both sides of the client/server connection.


server
1)request
Client
2)response is generated
at runtime
3)response is sent
to the client



```
public class DemoServlet extends HttpServlet{
public void doGet(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException
{
res.setContentType("text/html");//setting the content type
PrintWriter pw=res.getWriter();//get the stream to write the data

//writing html in the stream
pw.println("<html><body>");
pw.println("Welcome to servlet");
pw.println("</body></html>");

pw.close();//closing the stream
}}
```

# Java Buzzwords



| | |
|---|---|
| Simple | Java has a concise, cohesive set of features that makes it easy to learn and use. |
| Secure | Java provides a secure means of creating Internet applications. |
| Portable | Java programs can execute in any environment for which there is a Java run-time system. |
| Object-oriented | Java embodies the modern object-oriented programming philosophy. |
| Robust | Java encourages error-free programming by being strictly typed and performing run-time checks. |
| Multithreaded | Java provides integrated support for multithreaded programming. |
| Architecture-neutral | Java is not tied to a specific machine or operating system architecture. |
| Interpreted | Java supports cross-platform code through the use of Java bytecode. |
| High performance | The Java bytecode is highly optimized for speed of execution. |
| Distributed | Java was designed with the distributed environment of the Internet in mind. |
| Dynamic | Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve access to objects at run time. |

# Object-Oriented Programming

- To support the principles of OOP, Java have three traits in common
  - encapsulation,
  - polymorphism, and
  - inheritance.

# Encapsulation

- A programming mechanism that binds together code and the data it manipulates, and that keeps both safe from outside interference and misuse.
- Within an object, code, data, or both may be *private* to that object or *public*.
- Java's basic unit of encapsulation is the class.
  - A class defines the form of an object.
  - It specifies both the data and the code that will operate on that data. Java uses a class specification to construct objects.
  - Objects are instances of a class.
- The code and data that constitute a class are called members of the class.

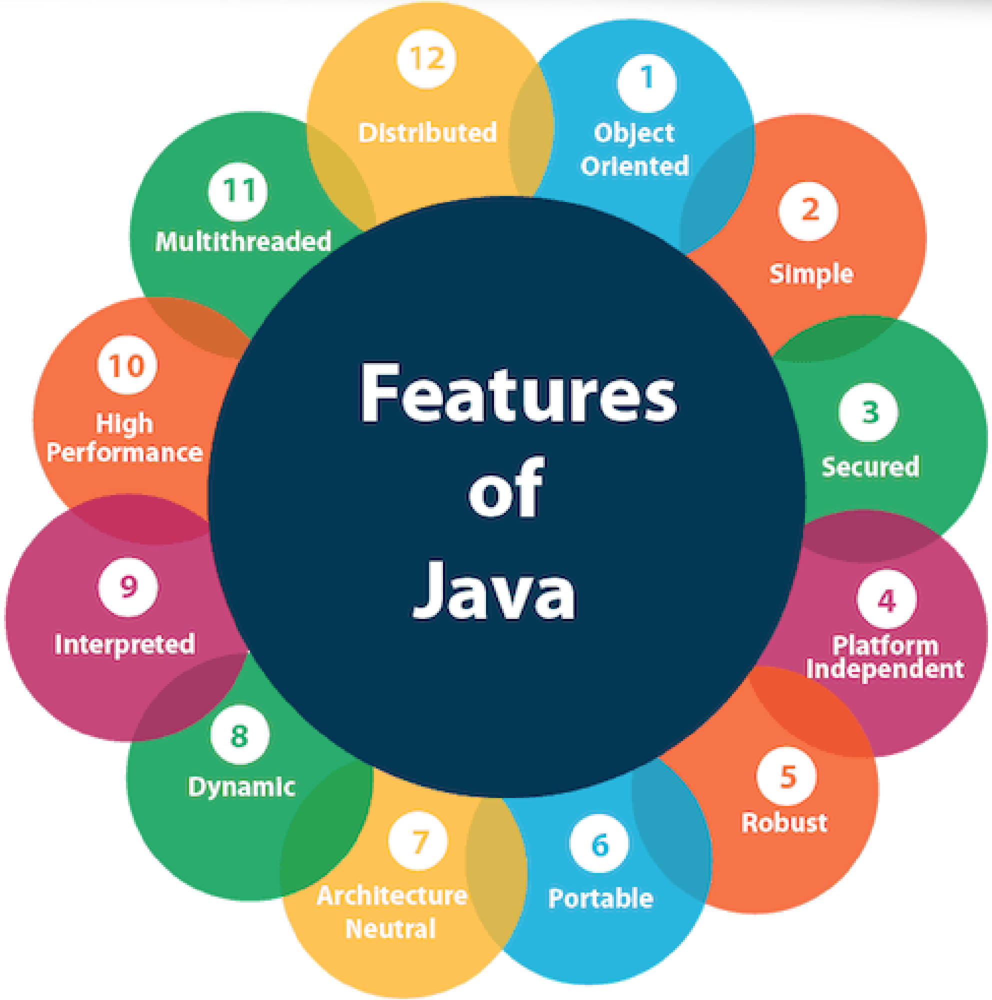
# Polymorphism

- Polymorphism = “many forms”
- The quality that allows one interface to access a general class of actions.
  - The specific action is determined by the exact nature of the situation.
- Polymorphism helps reduce complexity by allowing the same interface to be used to specify a general class of action.
  - It is the compiler’s job to select the specific action (i.e., method) as it applies to each situation.
  - The programmer don’t need to do this selection manually.

# Inheritance

- The process by which one object can acquire the properties of another object.
- This is important because it supports the concept of hierarchical classification.
- Using inheritance, an object need only define those qualities that make it unique within its class.
  - It can inherit its general attributes from its parent.

  →It is the inheritance mechanism that makes it possible for one object to be a specific instance of a more general case.

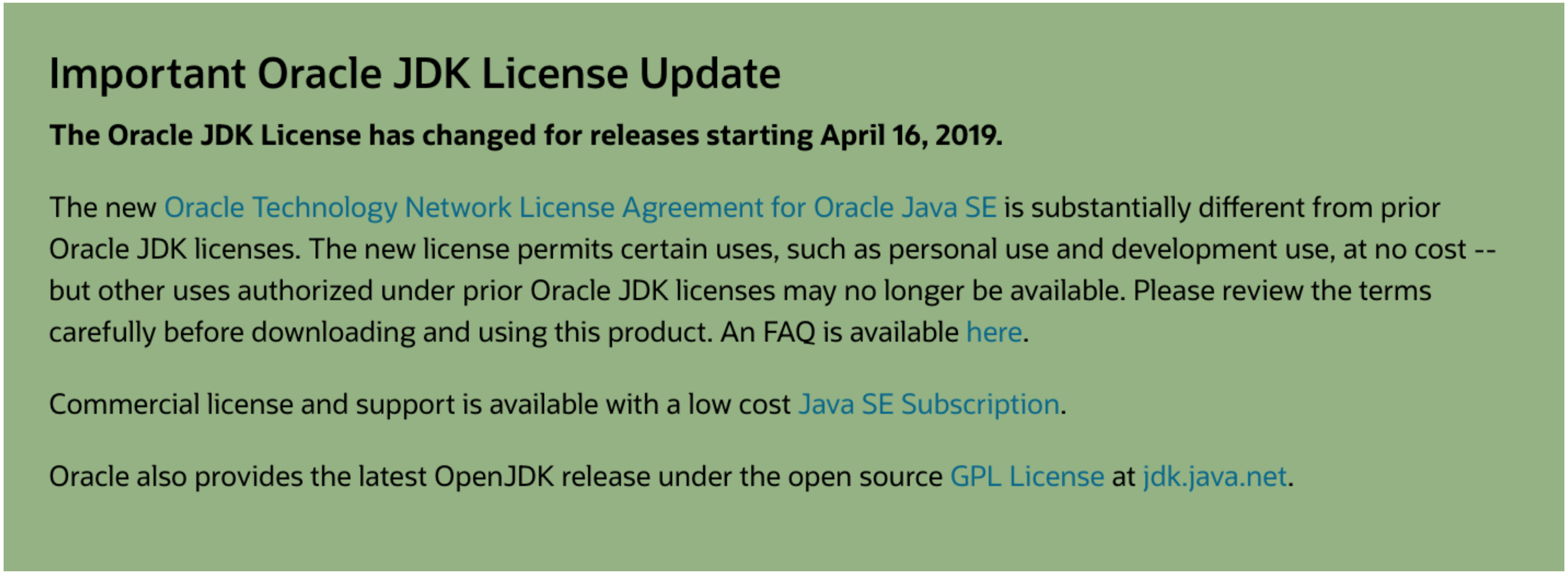

Features of Java
12
Distributed
1
Object Oriented
2
Simple
3
Secured
4
Platform Independent
5
Robust
6
Portable
7
Architecture Neutral
8
Dynamic
9
Interpreted
10
High Performance
11
Multithreaded

# Plan

1. History
2. How Java impacted the Internet
3. Compiling and running a Java program
4. Variables
5. Expression

# Java Development Kit [1]

- Before you can compile and run those programs, you must have a Java Development Kit (JDK).
- The current release of the JDK is JDK 15.
  - This is the version for Java SE 15. (SE stands for Standard Edition)
- If you need to install the JDK on your computer:
  - Be aware that for modern versions of Java, both Oracle JDKs and open source OpenJDKs are available for download.



## Important Oracle JDK License Update

**The Oracle JDK License has changed for releases starting April 16, 2019.**

The new Oracle Technology Network License Agreement for Oracle Java SE is substantially different from prior Oracle JDK licenses. The new license permits certain uses, such as personal use and development use, at no cost -- but other uses authorized under prior Oracle JDK licenses may no longer be available. Please review the terms carefully before downloading and using this product. An FAQ is available here.

Commercial license and support is available with a low cost Java SE Subscription.

Oracle also provides the latest OpenJDK release under the open source GPL License at jdk.java.net.

Source: https://www.oracle.com/java/technologies/javase-jdk14-downloads.html

# Java Development Kit [2]

- The JDK supplies two primary programs
  - **javac**: the Java compiler.
  - **java** : the standard Java interpreter and is also referred to as the application launcher.
- The JDK runs in the command-prompt environment and uses command-line tools.
  - It is not a windowed application.
  - It is also not an integrated development environment (IDE).
- There are several high-quality IDEs available for Java, such as NetBeans and Eclipse.

# My first example

```
public class HelloWorld {
   public static void main(String[] args) {
                  System.out.print("Hello, world!");
   }
}
```

Three steps:

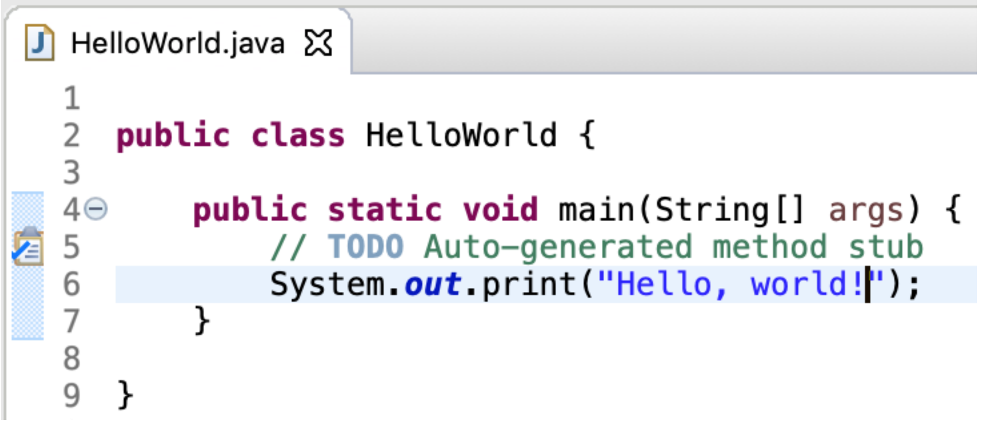
1. Enter the program.
2. Compile the program.
3. Run the program.

# My first example: Command line

Compile: javac HelloWorld.java

Run : java HelloWorld

Output: Hello, world



```

public class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.print("Hello, world!");
    }

}
```

# Comment

```
// A single-line comment

/*
 * This is a simple Java program
 * Multiline comment
 */
```

```
public class HelloWorld2 {
        public static void main(String[] args) {
                if (args.length >0) {
                        System.out.print("Hello, " + args[0]);
                }else {
                        System.out.println("Hello new user!");
                }
        }
}
```

```
/* This demonstrates a variable
 * Call this file Example2.java
 */
public class Example2{
        public static void main(String args[]) {
                int myVar1; // this declares a variable
                int myVar2; // this declares another variable

                myVar1 = 1024; // this assigns 1024 to myVar1
                System.out.println("myVar1 contains " + myVar1);

                myVar2 = myVar1 / 2;
                System.out.print("myVar2 contains myVar1 / 2: ");
                System.out.println(myVar2);
        }
}
```

# Arithmetic operators

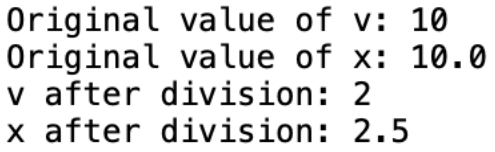
| | |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |

# Another data type

- To allow numbers with fractional components, Java defines two floating-point types: float and double
  - represent single and double-precision values, respectively.
  - double is the most commonly used.
- To declare a variable of type double:

```
double x;
```

```
/* This program illustrates the differences between int
 * and double; Call this file Example3.java
 */
public class Example3 {
        public static void main(String args[]) {
                int v; // this declares an int variable
                double x; // this declares a floating-point variable
                v = 10; // assign v the value 10
                x = 10.0;  // assign x the value 10.0

                System.out.println("Original value of v: " + v);
                System.out.println("Original value of x: " + x);

                // now, divide both by 4
                v = v / 4;
                x = x / 4;
                System.out.println("v after division: " + v);
                System.out.println("x after division: " + x);
        }
}
```



```
Original value of v: 10
Original value of x: 10.0
v after division: 2
x after division: 2.5
```

# Exercise: Converting Gallons to Liters

1 gallon = 3.7854 liters

1. Create a new file called GalToLit.java.
2. Implement the conversion of gallons to liters in GalToLit.java
3. Compile the program:

```
javac GalToLit.java
```

4. Run the program:

```
java GalToLit.java
```

# GalToLit.java

```java
/* This program converts gallons to liters
 * Call this program GalToLit.java
 */
public class GalToLit {

	public static void main(String[] args) {
		double gallons; // holds the number of gallons
		double liters; // holds conversion to liters
		gallons = 10; // start with 10 gallons
		liters = gallons * 3.7854; // convert to liters
		System.out.println(gallons + " gallons is " + liters + " liters.");
	}
}
```

# Two control statements

- if statement
- for loop

# if statement

if(condition) statement;

- condition is a Boolean expression
- Example:

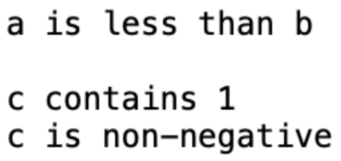
```
if (10 < 11) System.out.println("10 is less than 11");
if (10 < 9) System.out.println("this won't be displayed");
```

# Operators

| Operator | Meaning |
|---|---|
| < | Less than |
| <= | Less than or equal |
| > | Greater than |
| >= | Greater than or equal |
| == | Equal to |
| != | No equal |

# Example

```
/* Demonstrate the if
 * Call this file IfDemo.java
 */
public class IfDemo {
    public static void main(String[] args) {
        int a, b, c;
        a = 2;
        b = 3;
        if(a < b) System.out.println("a is less than b");
        //this won't display anything
        if(a == b) System.out.println("you won't see anything");
        System.out.println();
        c = b - a; // c contains 1
        System.out.println("c contains 1");
        if(c >= 0) System.out.println("c is non-negative");
        if(c < 0) System.out.println("c is negative");
    }
}
```



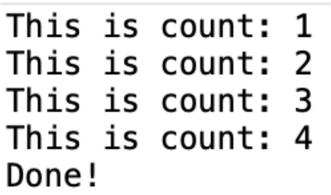
```
a is less than b

c contains 1
c is non-negative
```

# for Loop

for(initialization; condition; iteration) statement;

- Example:

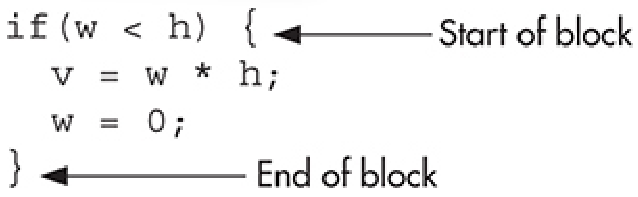
```
/* Demonstrate the for loop.
 * Call this file ForDemo.java.
 */
public class ForDemo {

    public static void main(String[] args) {
        int count;
        for(count = 0; count < 5; count = count + 1)
        System.out.println("This is count: " + count);
        System.out.println("Done!");
    }
}
```



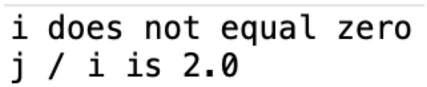
```
This is count: 1
This is count: 2
This is count: 3
This is count: 4
Done!
```

# Create blocks of code

```
/* Demonstrate a block of code
 * Call this file BlockDemo.java.
 */
public class BlockDemo {

	public static void main(String[] args) {
		double i, j, d;
		i = 5;
		j = 10;
		// the target of this if is a block
		if(i != 0) {
			System.out.println("i does not equal zero");
			d = j / i;
			System.out.println("j / i is " + d);
		}
	}
}
```



```
if(w < h) {
  v = w * h;
  w = 0;
}
```



```
i does not equal zero
j / i is 2.0
```

# Semicolons and positioning

- In Java, the semicolon is a separator
- It is often used to terminate a statement.
- In essence, the semicolon indicates the end of one logical entity.

```
x = y;
y = y + 1;
System.out.println(x + " " + y);
```

- Java does not recognize the end of the line as a terminator.

```
x = y; y = y + 1; System.out.println(x + " " + y);
System.out.println("This is a long line of output"
                                        + x + y + x + "more output");
```

# Indentation practices

- Java is a freeform language, meaning that it does not matter where you place statements relative to each other on a line.
- However, over the years, a common and accepted indentation style has developed that allows for very readable programs.
- Using this style, you indent one level after each opening brace, and move back out one level after each closing brace.

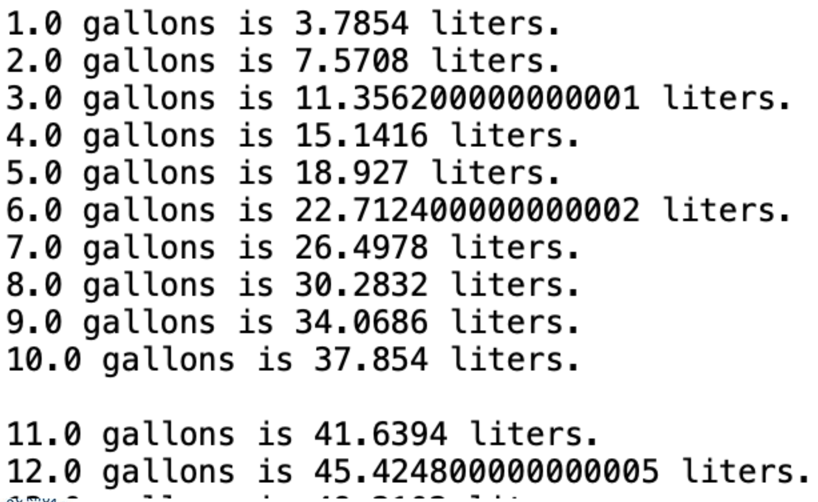
# Exercise: Improving the Gallons-to-Liters Converter

1. Create a new file called GalToLitTable.java.
2. Implement program into the file GalToLitTable.java
3. Compile the program:

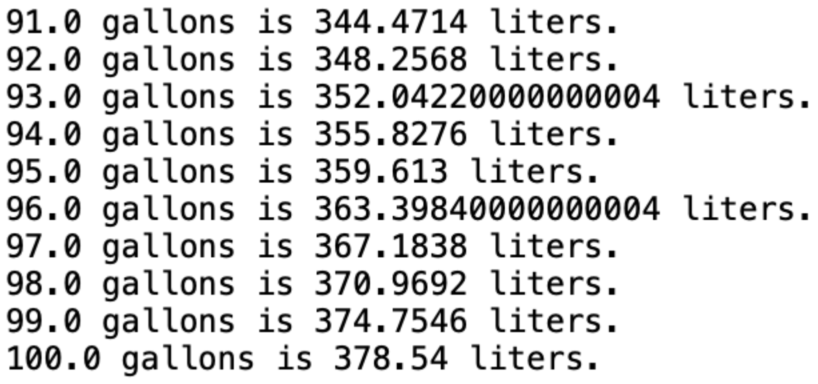
   javac GalToLitTable.java

5. Run the program:

   java GalToLitTable



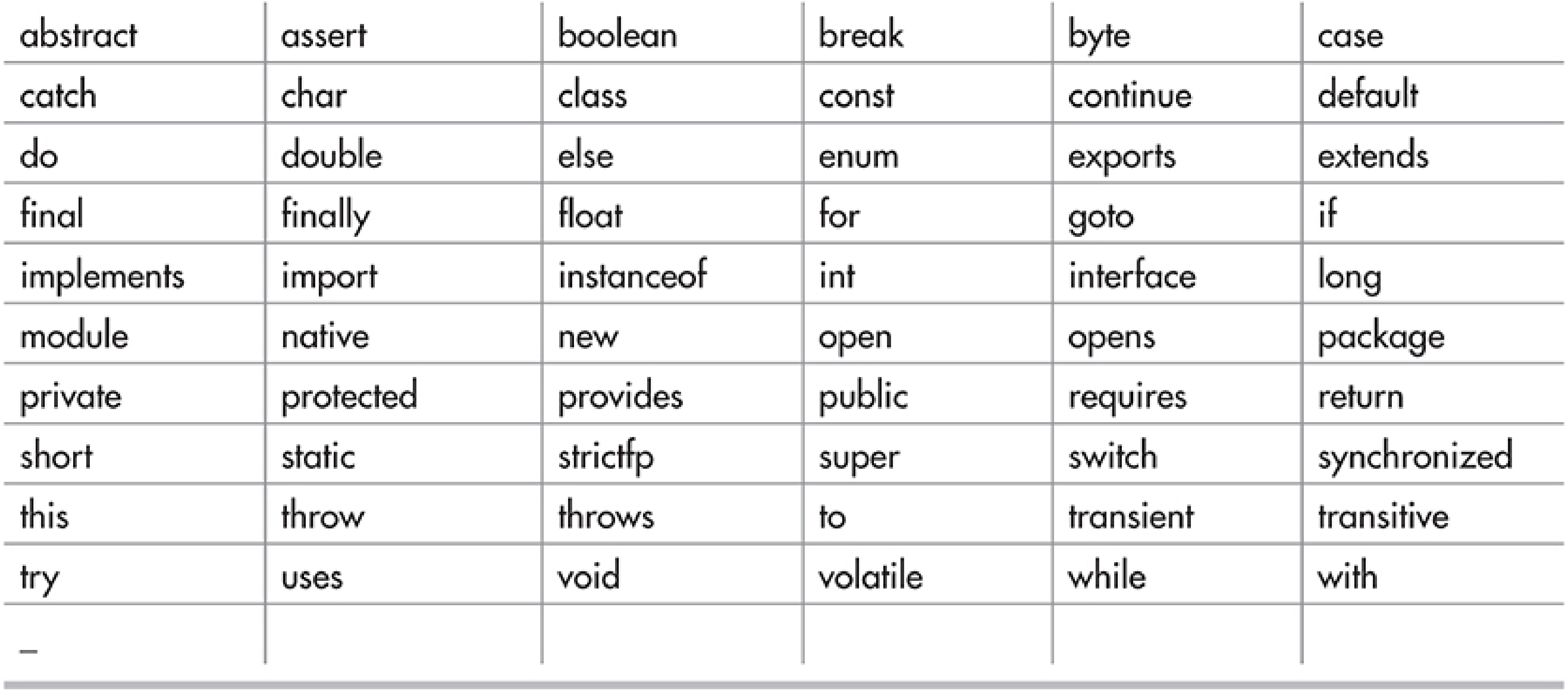
```
1.0 gallons is 3.7854 liters.
2.0 gallons is 7.5708 liters.
3.0 gallons is 11.356200000000001 liters.
4.0 gallons is 15.1416 liters.
5.0 gallons is 18.927 liters.
6.0 gallons is 22.712400000000002 liters.
7.0 gallons is 26.4978 liters.
8.0 gallons is 30.2832 liters.
9.0 gallons is 34.0686 liters.
10.0 gallons is 37.854 liters.

11.0 gallons is 41.6394 liters.
12.0 gallons is 45.424800000000005 liters.
```



```
91.0 gallons is 344.4714 liters.
92.0 gallons is 348.2568 liters.
93.0 gallons is 352.04220000000004 liters.
94.0 gallons is 355.8276 liters.
95.0 gallons is 359.613 liters.
96.0 gallons is 363.39840000000004 liters.
97.0 gallons is 367.1838 liters.
98.0 gallons is 370.9692 liters.
99.0 gallons is 374.7546 liters.
100.0 gallons is 378.54 liters.
```

```
class GalToLitTable {
    public static void main (String args[]) {
        double liters;
        int counter = 0;
        for (double gallons = 1; gallons <= 100; gallons++) {
            liters = gallons * 3.7854;
            System.out.println(gallons + " gallons is "
                                                                   + liters + " liters.");
            counter++;
            if (counter == 10) {
                System.out.println();
                counter = 0;
            }
        }
    }
}
```

# Java keywords



| | | | | | |
|---|---|---|---|---|---|
| abstract | assert | boolean | break | byte | case |
| catch | char | class | const | continue | default |
| do | double | else | enum | exports | extends |
| final | finally | float | for | goto | if |
| implements | import | instanceof | int | interface | long |
| module | native | new | open | opens | package |
| private | protected | provides | public | requires | return |
| short | static | strictfp | super | switch | synchronized |
| this | throw | throws | to | transient | transitive |
| try | uses | void | volatile | while | with |
| _ | | | | | |

# Identifiers in Java

- An identifier is a name given to a method, a variable, or any other user-defined item.
- Identifiers can be from one to several characters long.
- Variable names may start with any letter of the alphabet, an underscore, or a dollar sign. Next may be either a letter, a digit, a dollar sign, or an underscore.
- The underscore can be used to enhance the readability of a variable name, as in line_count.
- Uppercase and lowercase are different; that is, to Java, myvar and MyVar are separate names.

# Java class libraries

- The Java environment relies on several built-in class libraries that contain many built-in methods that provide support for such things as I/O, string handling, networking, and graphics.
- The standard classes also provide support for a graphical user interface (GUI).
- Thus, Java as a totality is a combination of the Java language itself, plus its standard classes.

# References

- Java Programming, NGUYỄN Thị Minh Tuyền, Ph.D.
- Java Programming, NGUYỄN Văn Khiết, M.Sc.
- Java Programming, NGUYỄN Hoàng Anh, M.Sc.
- **Herbert Schildt. (2019). Java: A Beginner's Guide, 8th Edition, McGraw-Hill Education.**
- Herbert Schildt. (2019). Java: The Complete Reference, 11th Edition. McGraw-Hill Education.
- https://docs.oracle.com/javase/tutorial/
- https://www.javatpoint.com/

# Question?

# Exercise Time

- Work in groups of 3 members.
- Exercise 1: create a Google Slide to answer the following questions.
  - For each question, remember to write down the ID – Full name who answered this question.
  - Convert to Exercise01.pdf
- Exercise 2: write code in Java
  - For each source code file, let me know who wrote this code.
  - Exercise2A.java
  - Exercise2B.java
  - Exercise2C.java
- Submit: StudentID1_StudentID2_StudentID3.zip
  - StudentID1 < StudentID2 < StudentID3

# Excercise 1

1. What types of Java applications are?
2. What are the differences between JavaSE, JavaEE, and JavaME?
3. What are the differences between JVM, JRE, and JDK?
4. Suppose that we write Java desktop application,
    1. and then we deploy it on a non-programming desktop,
    2. what should we install?
5. What are the differences of JDK and OpenJDK?
    1. What is the latest release version of JDK?
    2. What is the latest release version of OpenJDK?
6. Besides Java, which programming languages have their intepreters?

# Exercise 2

Write a program:

1. Ask user to enter personal information (first name, last name, date of birth, adress etc.) and display information on screen.
2. Display personal information received from aguments passed through the function main.
    1. Command line arguments
3. Display the square root of floating points numbers (double) from 1 to 100 and its rounding error.