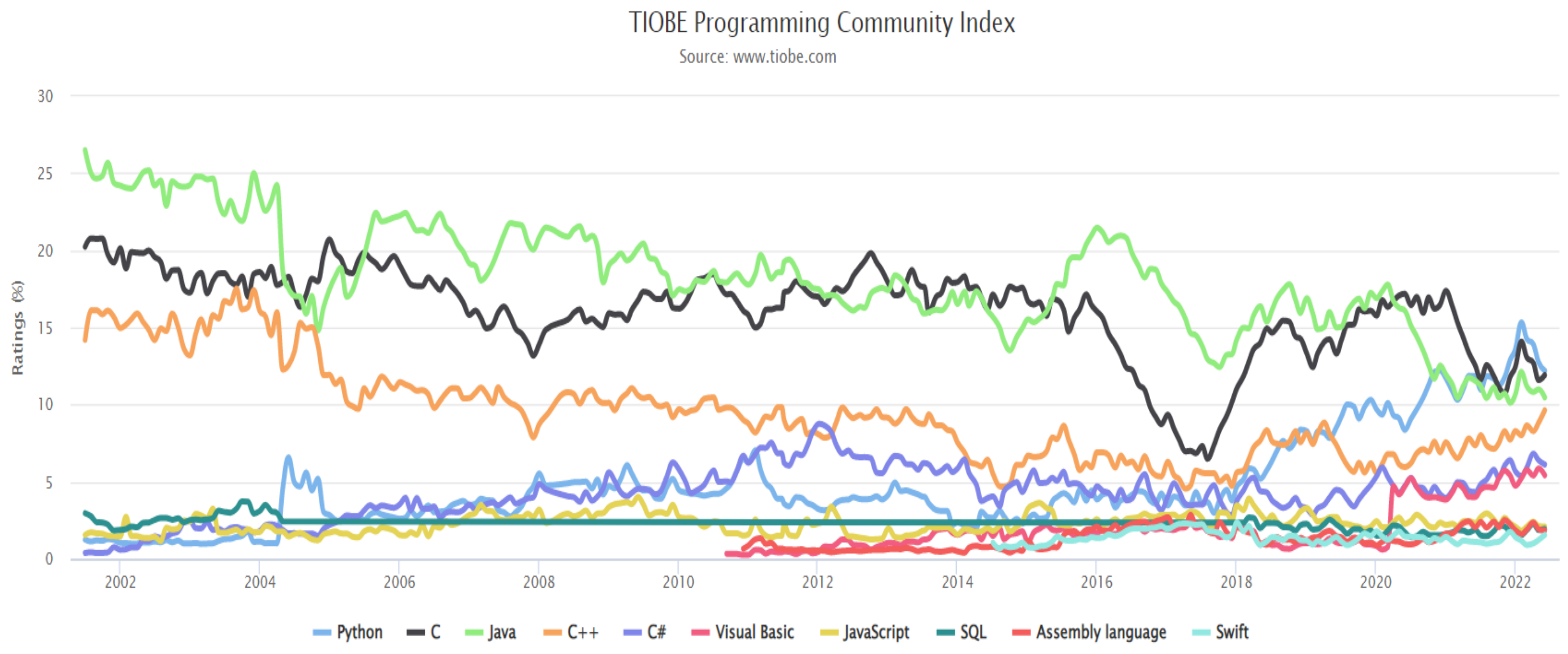
Basic Concepts of Java Programming Part 1

@UD

(Java features, Hello World)

Jul-2022

Java in IT Industry (Jun'2022)



Background

- Object-oriented language developed by Sun in mid 1990s.
  - Originally called Oak
  - Originally intended for embedded systems and consumer electronics

Java Version History

1993 Oak project at Sun

- small, robust, architecture independent, Object-Oriented, language to control interactive TV.
- didn’t go anywhere

1995 Oak becomes Java

- Focus on the web

1996 Java 1.0 available

1997 (March) Java 1.1 - some language changes, much larger library, new event handling model

1997 (September) Java 1.2 beta – huge increase in libraries including Swing, new collection classes, J2EE

1998 (October) Java 1.2 final (Java2!)

2000 (April) Java 1.3 final

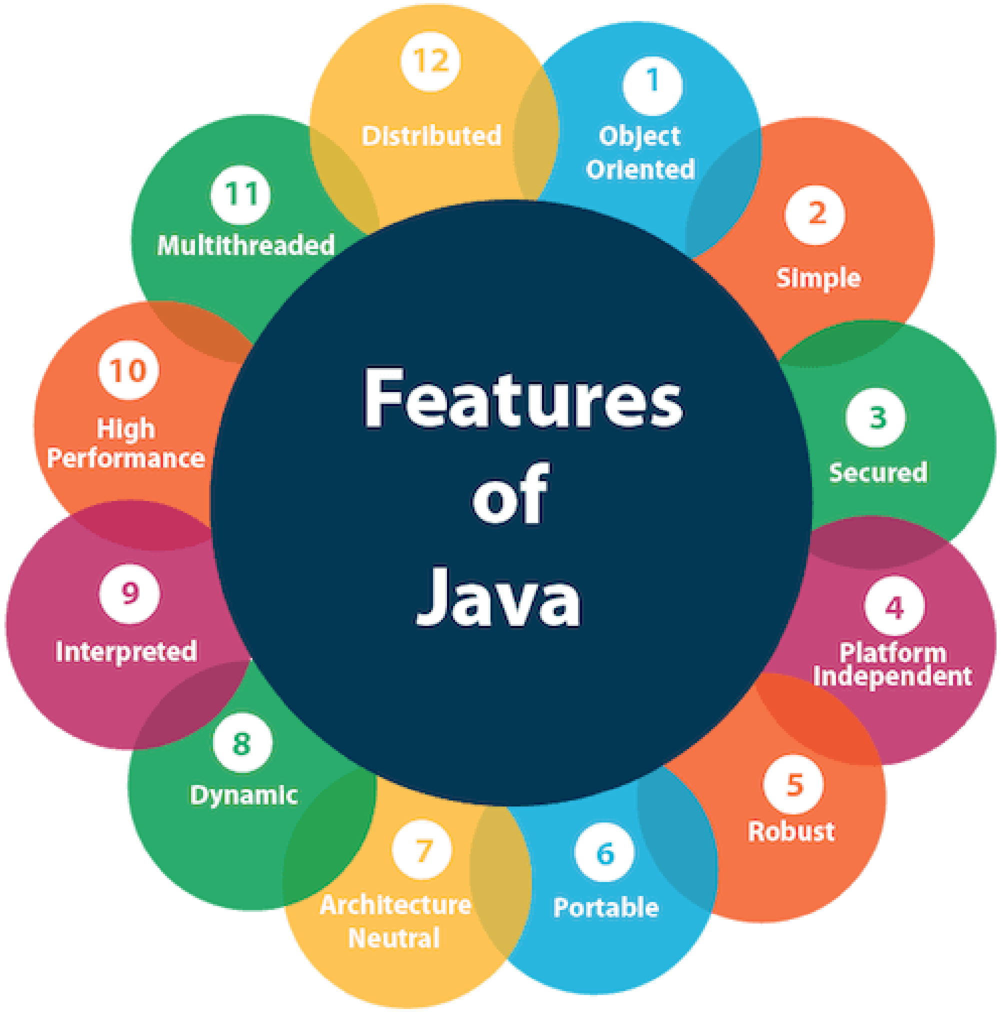
2001 Java 1.4 final (assert)

2004 Java 1.5 (parameterized types, enum, …) (Java5!)

2005 J2EE 1.5

…

What is Java? (cont)



- Object-Oriented
  - Designed to support Object Oriented concepts
  - Contains non-Object Oriented primitive data types

What is Java? (cont)

- Distributed
  - Applications constructed using objects.
  - Objects distributable in multiple locations (within a network environment).
  - Extensive integration with TCP/IP

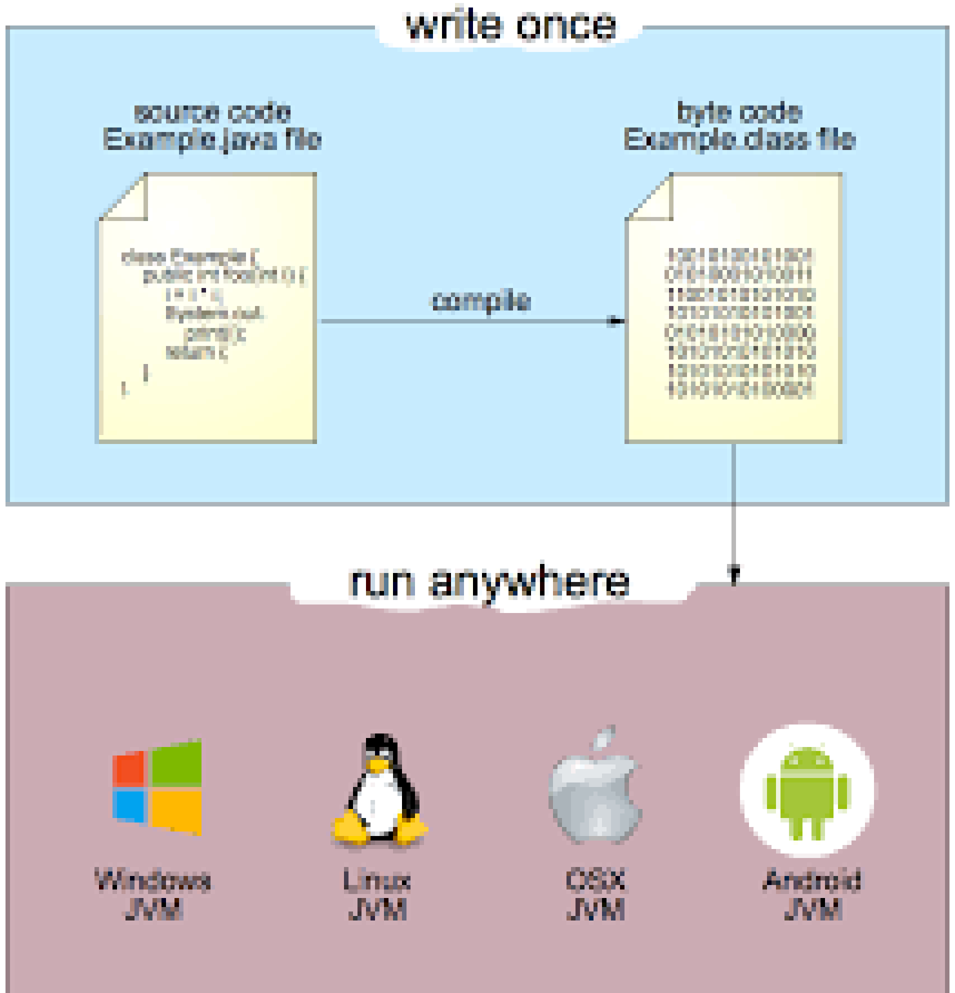
- Interpreted
  - Compiles to byte-code (not machine code). Byte code is interpreted
  - Most Java versions after 1.2 include JIT (Just-In-Time) compiler (which compiles byte code to machine code)

- Robust
  - Memory management done automatically
  - Use of pointers limited

What is Java? (cont)

- Secure
  - All Java code subject to security model.

- Architecture-Neutral/Portable
  - Compiled Java (byte code) will run on any platform having JVM (Java Virtual Machine)
  - The Java Virtual Machine is available for almost all platforms (even mainframes)



What is Java? (cont)

- High-Performance
  - Originally, Java's performance was poor
  - Now, Java's performance rivals C++
- Multi-Threaded
  - Processes contain multiple threads of execution.
  - Similar to multi-tasking but all threads share the same memory space
- Dynamic
  - Makes heavy use of dynamic memory allocation.
  - Classes can be dynamically loaded at any time.

Platform Independence - How Java does it?

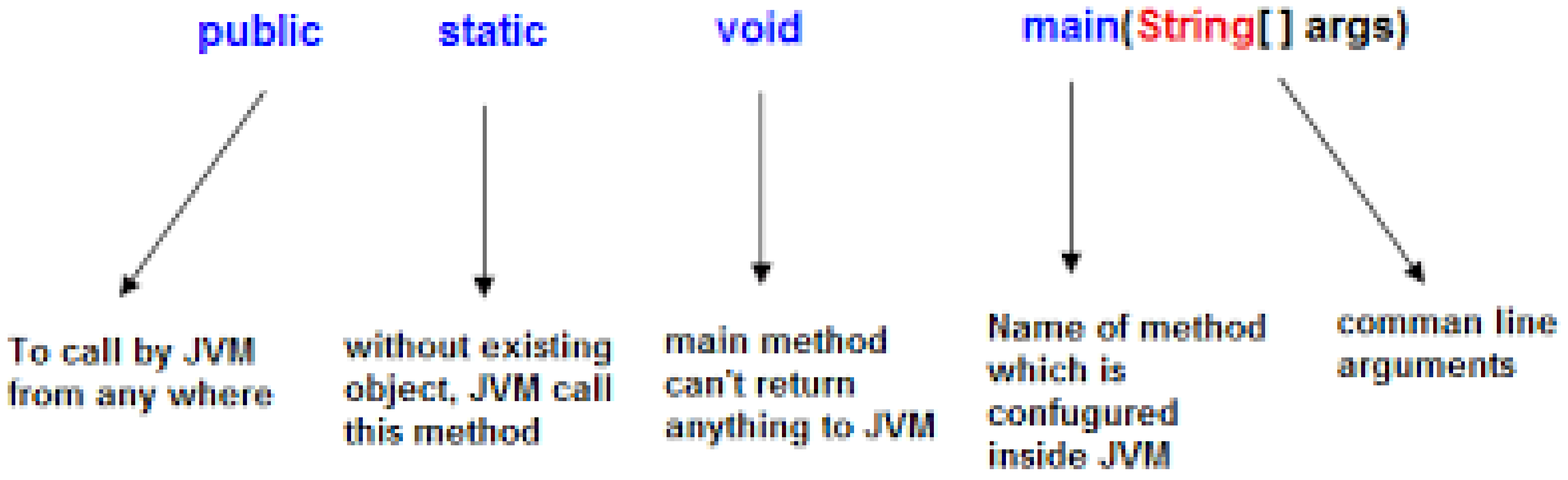
- Java described as WORA (Write once, Run Anywhere)
  - Mostly true
  - Not always true with GUI.
  - Require a lot of testing.

- Because Java source code (.java) compiled to Byte Code (.class) and Byte Code is interpreted…
  - Java code can be executed anywhere that interpreter is available

- The "Interpreter" is called the Java Virtual Machine

HelloWorld.java

```
public class HelloWorld {
        public static void main(String[] args){   // method
                System.out.println("Hello World");
           }
}
```


public
static
void
main(String[] args)
To call by JVM from any where
without existing object, JVM call this method
main method can't return anything to JVM
Name of method which is confugured inside JVM
comman line arguments

Running HelloWorld

- To compile *HelloWorld.java*, use the compiler. If successful, it will produce a file called HelloWorld.class in the same directory.

```
$ javac HelloWorld.java
  [ compiler output ]
```

errors and warnings

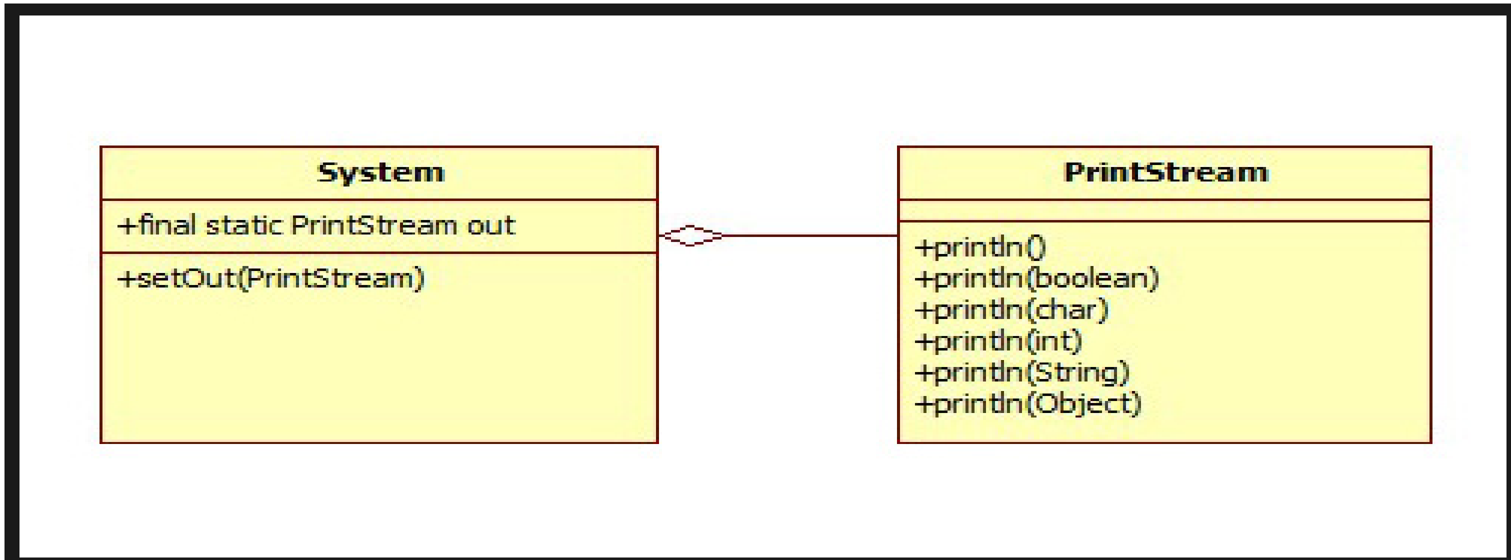
- To execute, run the Java VM and include the name of the class which contains the "main" method as the first command line parameter.

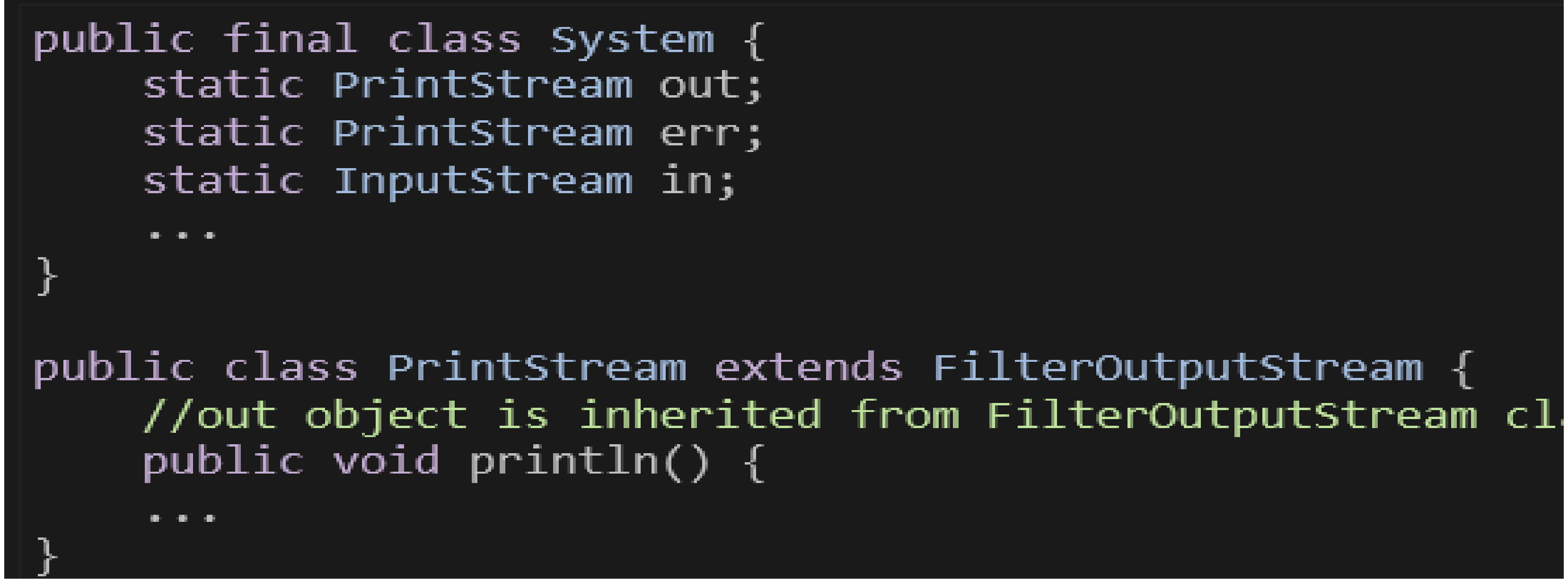
```
$ java HelloWorld
Hello World
```

note: do not include the .class extension

output from program

Additional reading: System.out.println





```
public final class System {
    static PrintStream out;
    static PrintStream err;
    static InputStream in;
    ...
}

public class PrintStream extends FilterOutputStream {
    //out object is inherited from FilterOutputStream cl
    public void println() {
    ...
}
```