

# Introduction to Java

# What is Java?

---

- ❖ Java supports object-oriented programming such as classes, inheritance, and polymorphism
- ❖ Our first Java Program – Hello World !

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

```
Hello, World!
```

# Java Program

---

- ❖ A Java program is really a **class definition** with a method named **main**.
- ❖ When the program is run, the method named **main** is invoked.

```
public class HelloWorld {  
    public static void main(String[] args) {  
        // You actual codes are here  
        System.out.println("Hello, World!") ;  
    }  
}
```

# Another Java Program

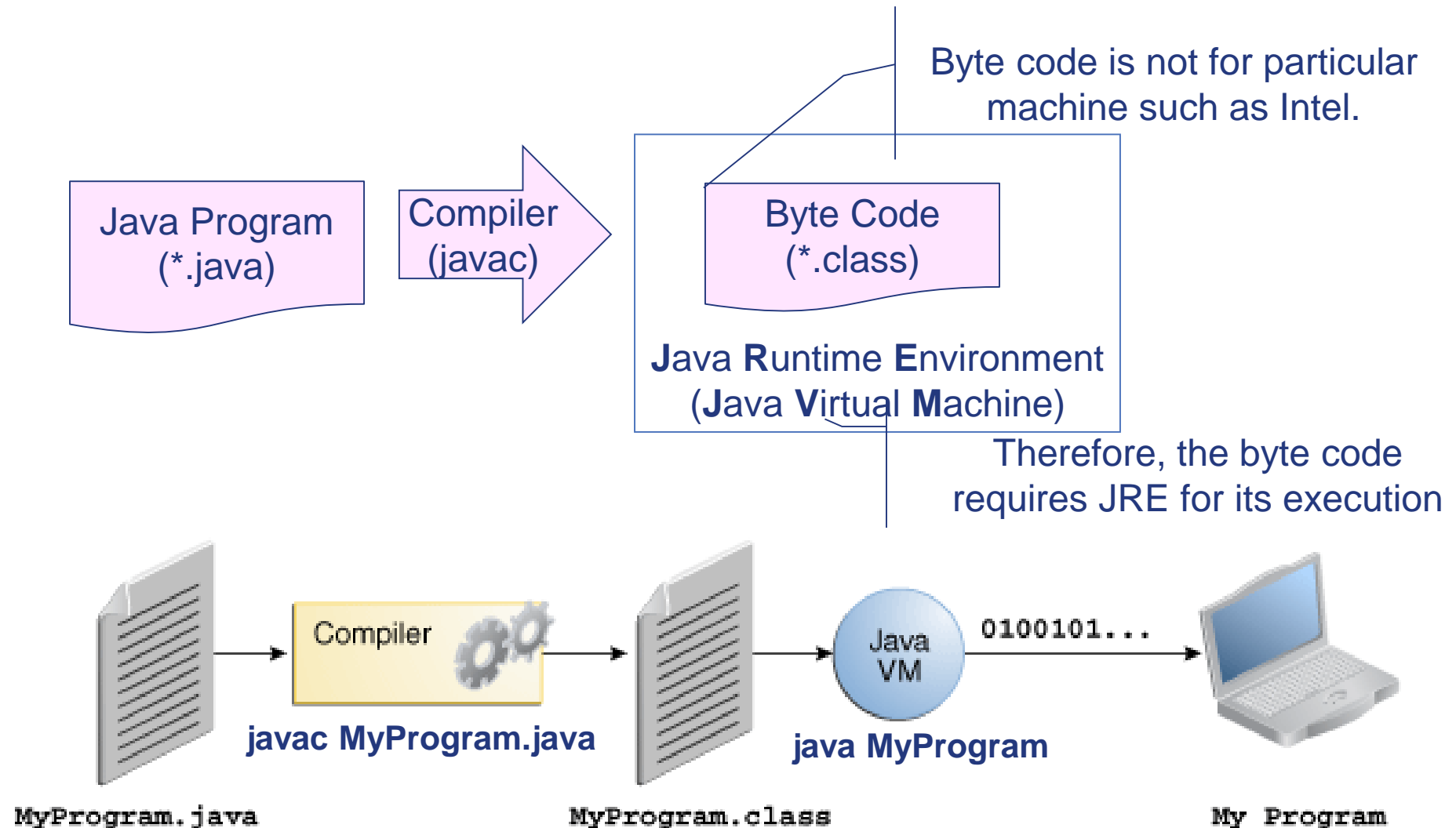
---

❖ Java is similar to C++ in many ways

- Types: int, long, array
- Control structures: for, while, do, return, break, continue
- Function call and parameter passing

```
public class Factorial {  
    public static void main(String[] args) {  
        int values[] = {5, 10, 15} ;  
        for ( int i: values )  
            System.out.println("Factorial of " + i + " is " + factorial(i) ) ;  
    }  
    private static long factorial(final int n) {  
        long result = 1 ;  
        for ( int i = 1 ; i <= n ; i ++ )  
            result *= i ;  
        return result ;  
    }  
}
```

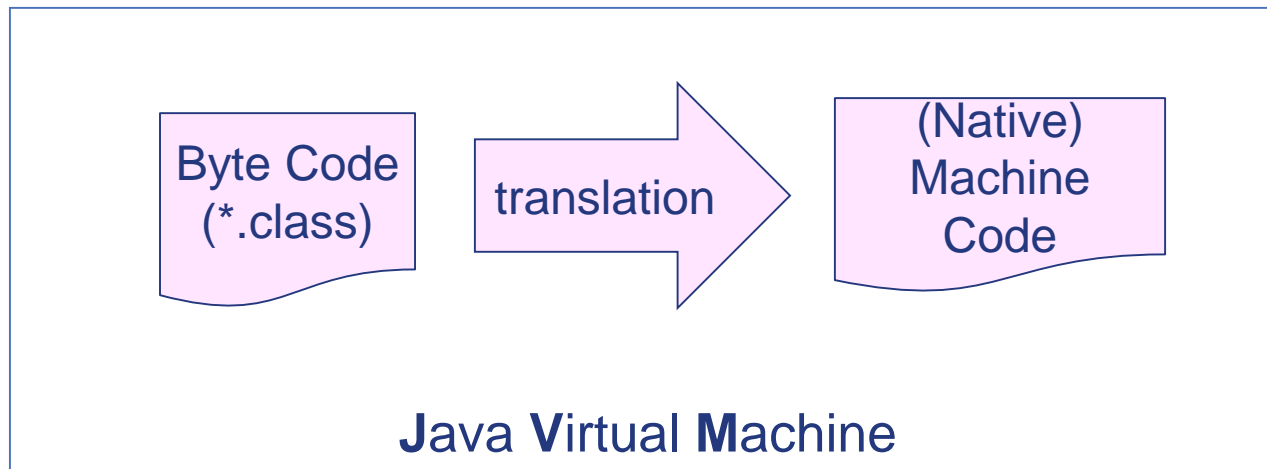
# Java Compilation and Execution



# Java Virtual Machine: Disadvantage

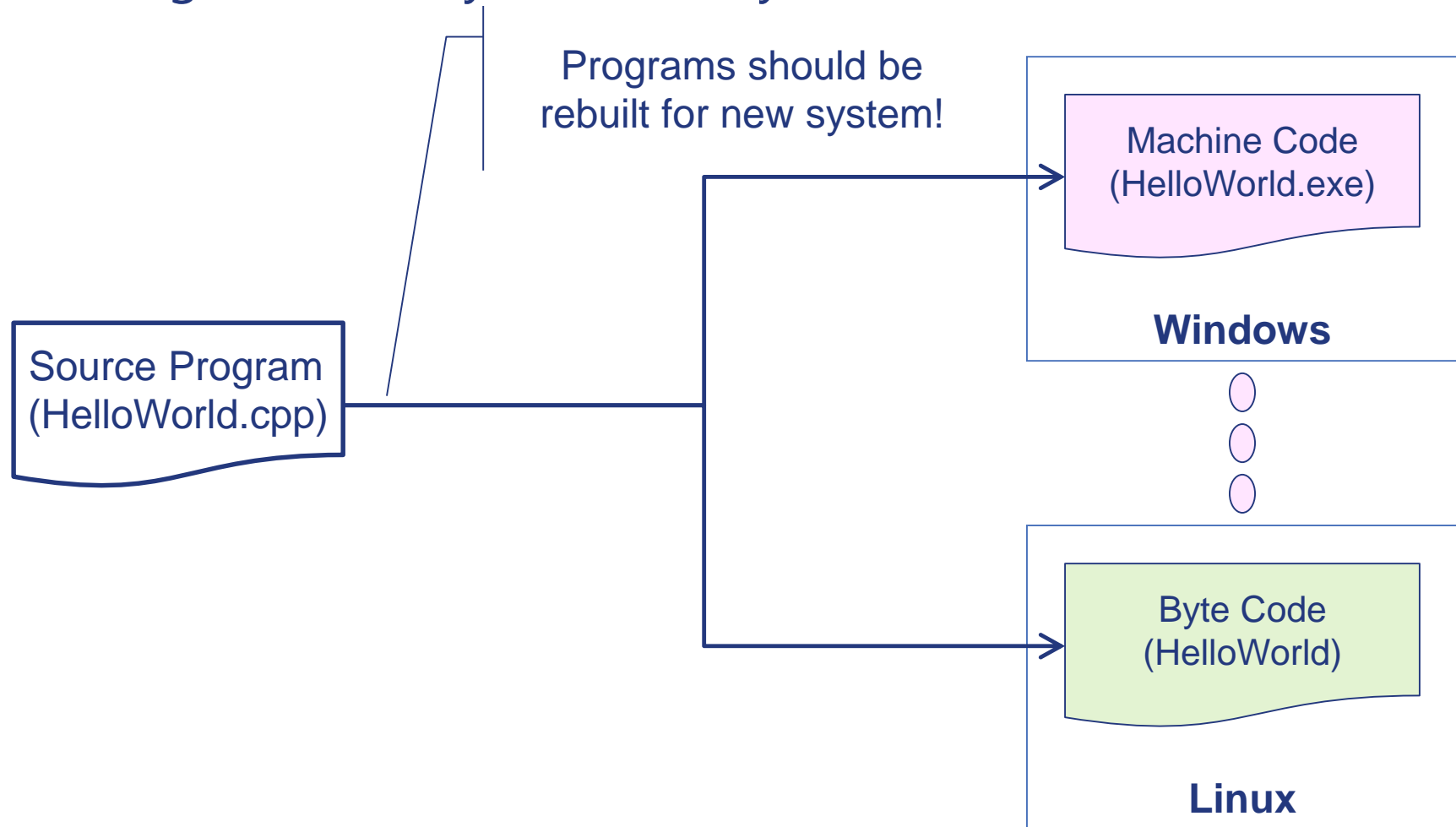
---

- ❖ At run-time, It requires an additional translation from byte code to machine code
- ❖ So, Java programs may show poor performance



# Java Virtual Machine: Advantage

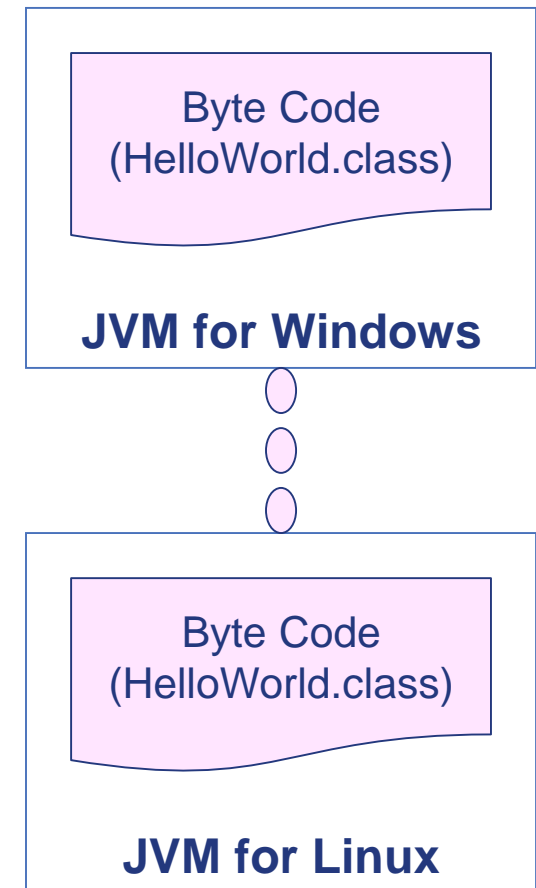
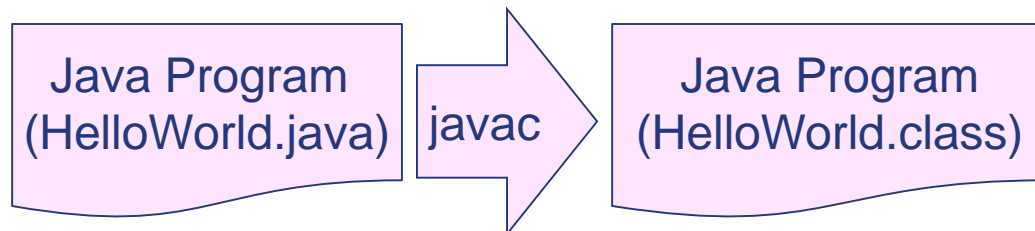
- ❖ Porting to other system is very difficult!



# Java Virtual Machine: Advantage

---

- ❖ Java programs are portable:
- ❖ No compilation is required!
- ❖ This is called Write Once, Run Anywhere

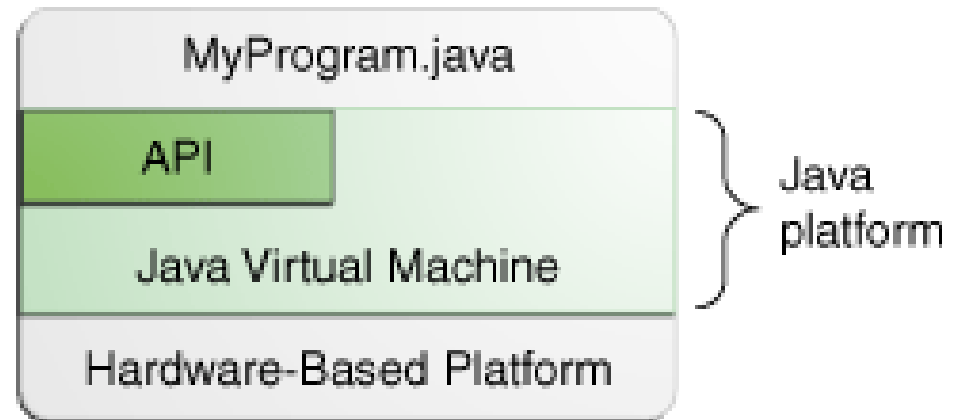




# Where can we get JVM?

---

- ❖ We can get JVM from Oracle free of charge.
  - <https://www.oracle.com/technetwork/java/javase/downloads/index.html>
- ❖ JVMs for the following OSs are available
  - Windows x86, x64
  - Linux x86, x64
  - Solaris SPARC
  - Solaris x64
  - Mac OS X



- ❖ We can also get specific VMs for specific platforms.

# Java is very popular!

- ❖ Java has been used for various applications.
- ❖ Java offers three editions for different applications
  - Java **EE** (Enterprise Edition)
  - Java **SE** (Standard Edition)
  - Java **ME** (Micro Edition)



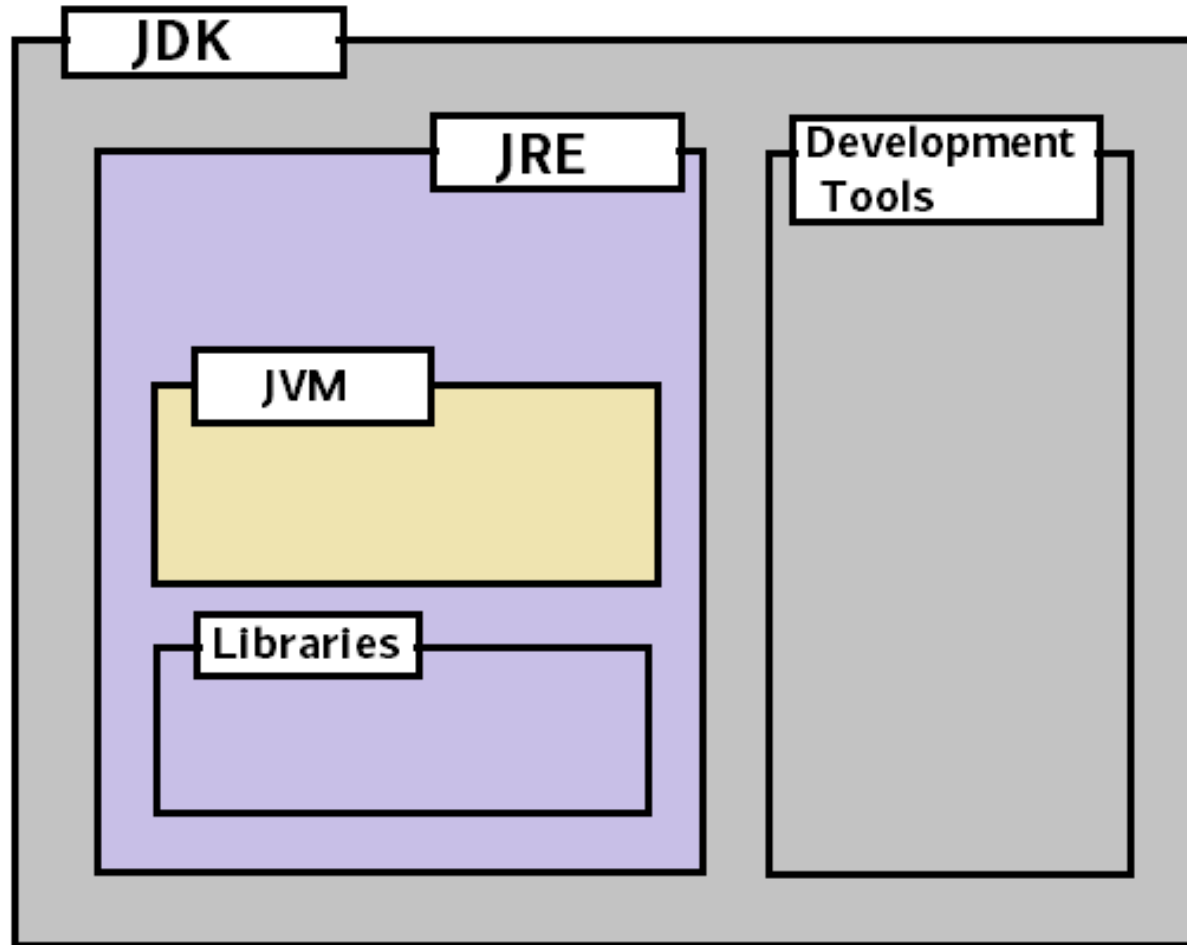
# Java is a system

---

- ❖ Java provides a huge number of useful APIs.
- ❖ Typical APIs are
  - **Swing, JavaFX** for GUI programming
  - **Applet** and **Servlet** for Web programming
  - **Socket** for Network programming
  - **JDBC** (Java Database Connectivity) for Database programming
  - **RMI** (Remote Method Invocation) for Distributed programming
  - **JNI** (Java Native Interface) for Native programming

# JDK vs JRE vs JVM

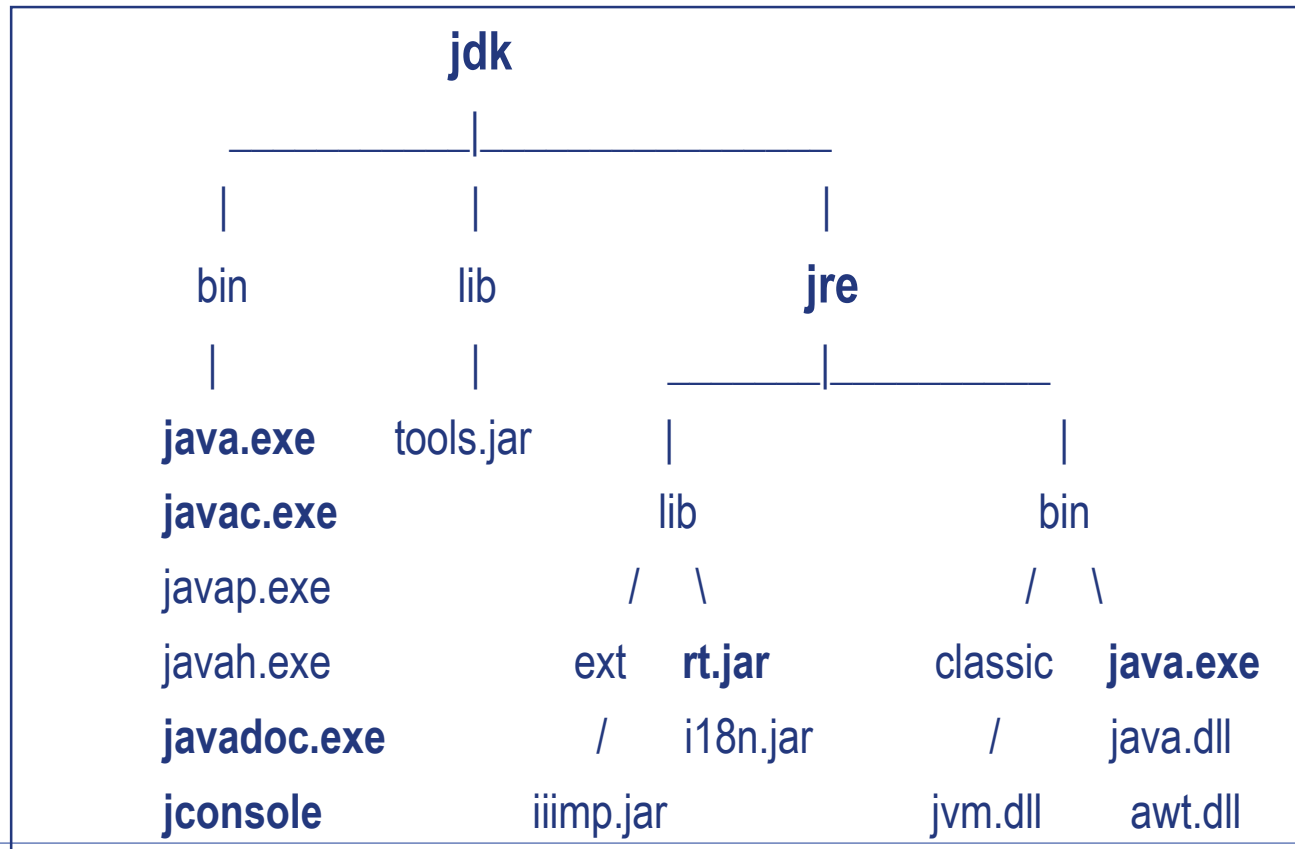
---



# Programming Environment for Java

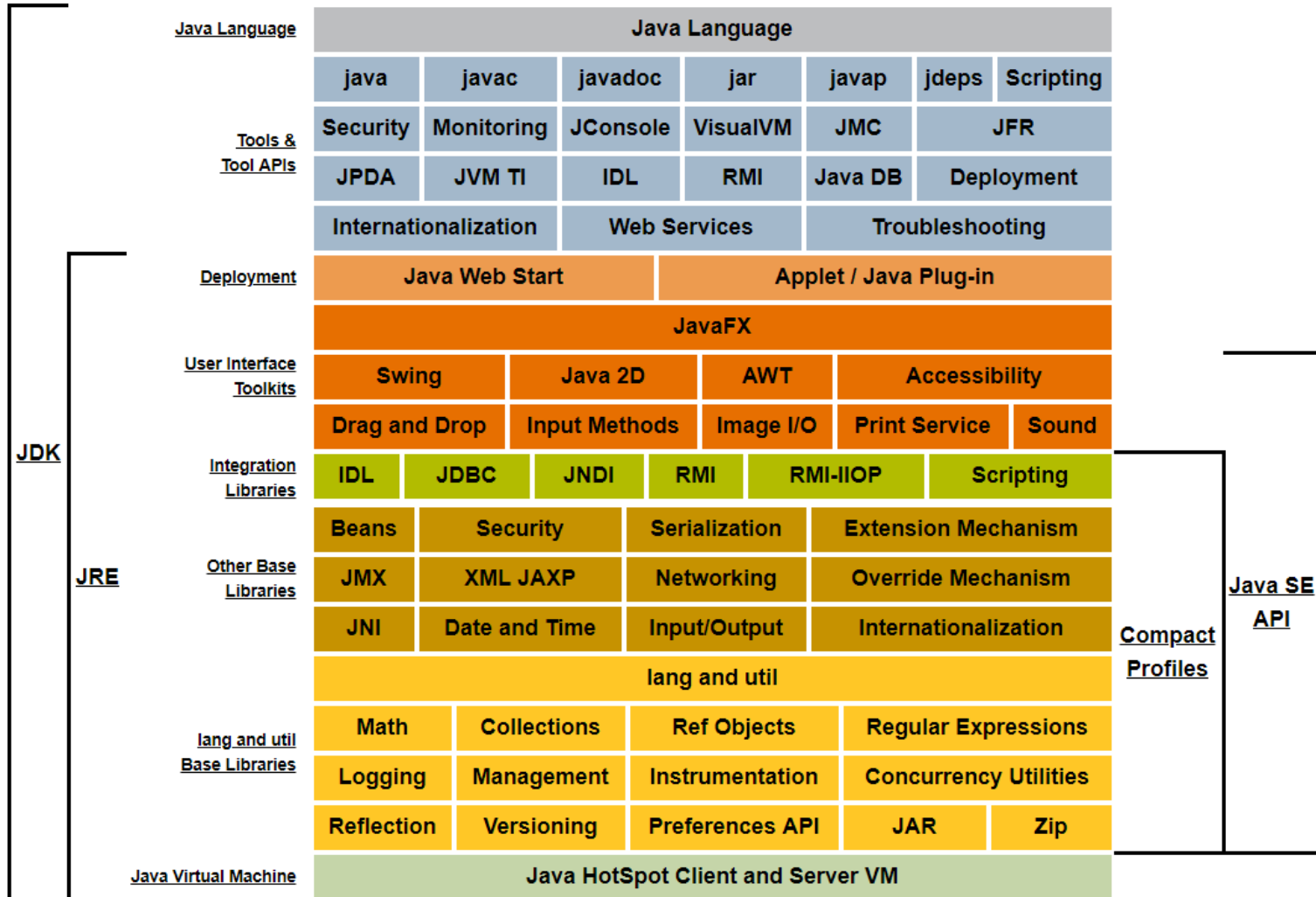
## ❖ JDK (Java Development Kit)

- the basic tools and libraries for developing Java applications
- <http://www.oracle.com/technetwork/java/javase/downloads/index.html>



Before Java 9

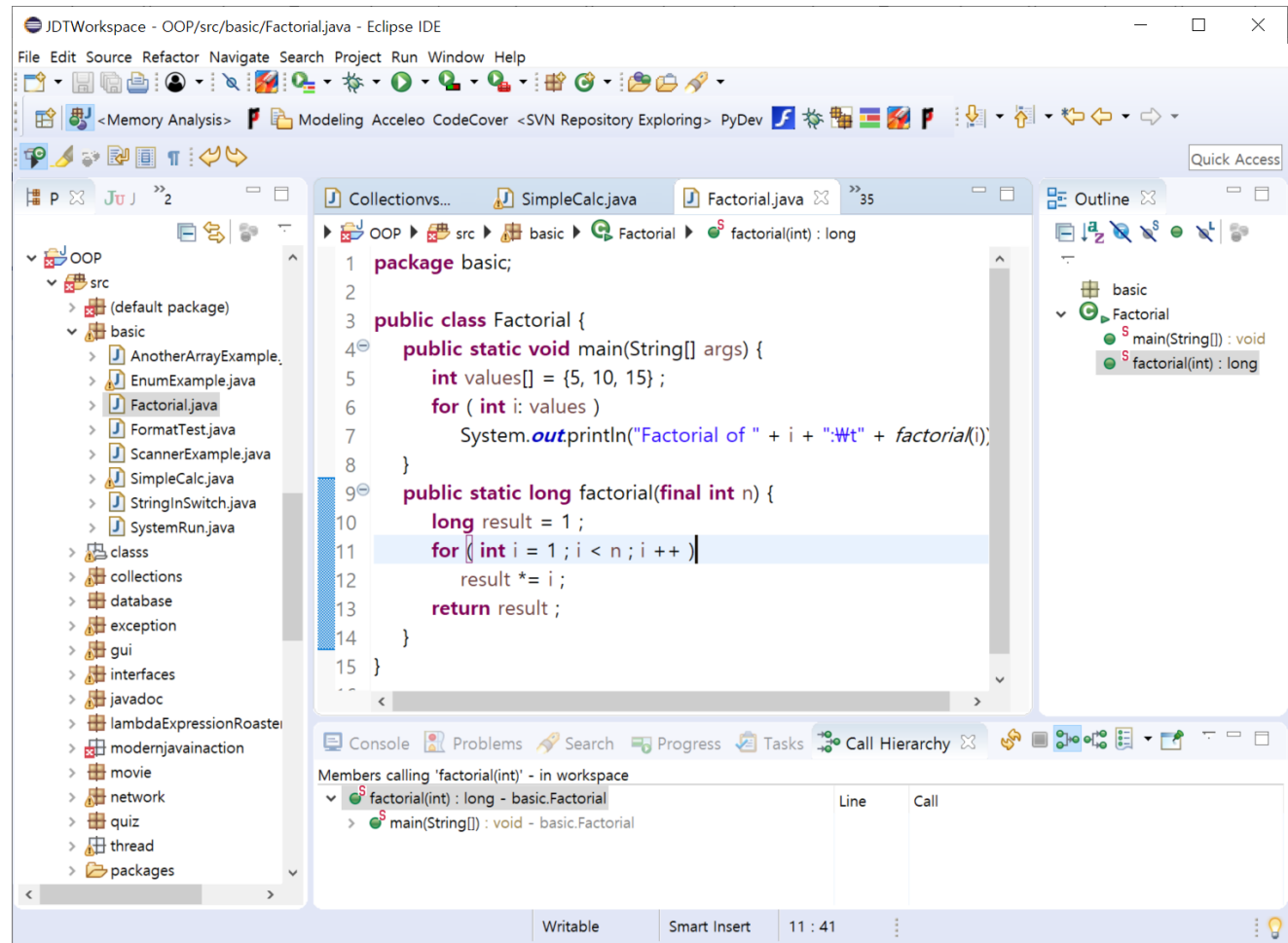
# Java SE 8 Platform



<https://docs.oracle.com/javase/8/docs/>

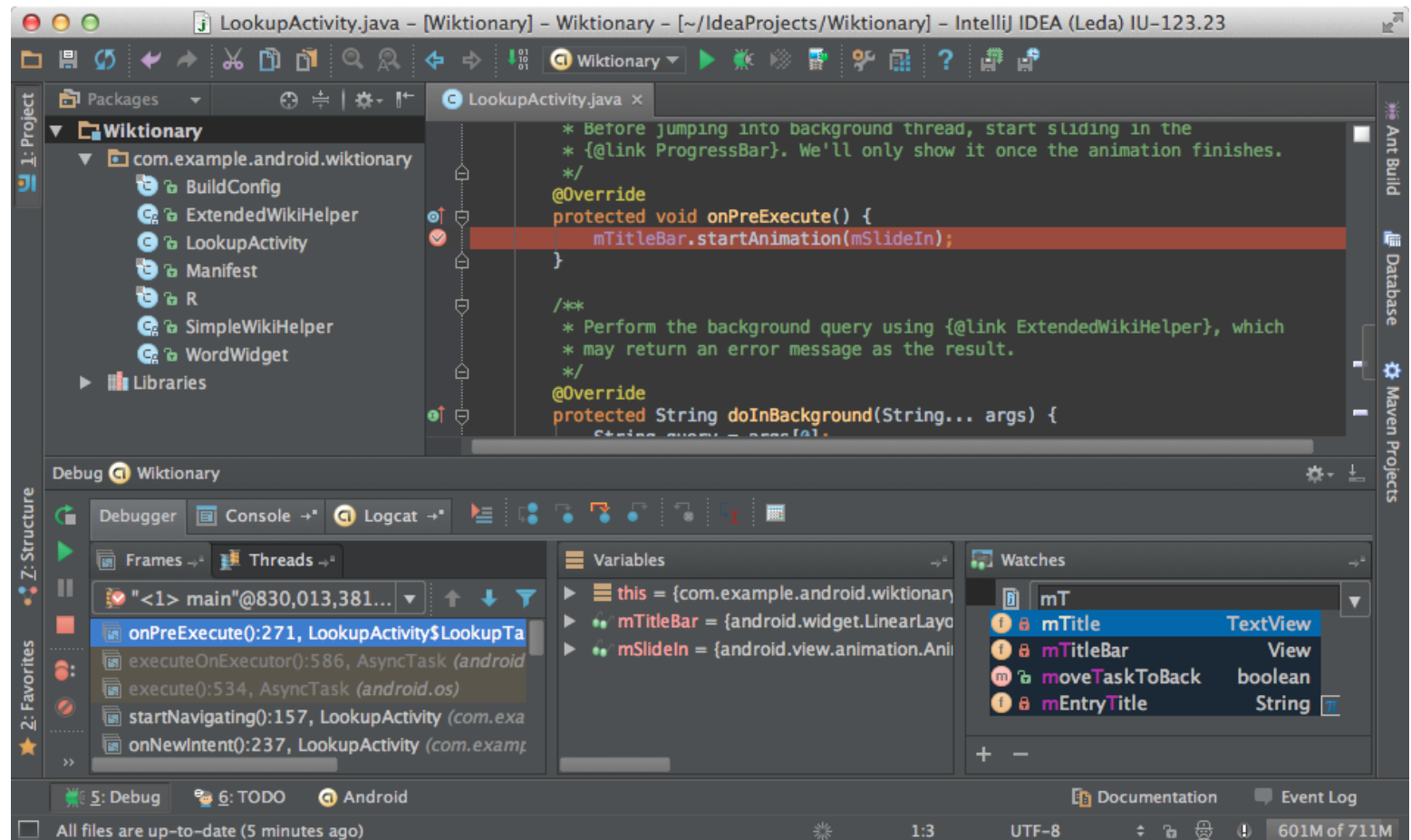
# Programming Environment for Java

❖ Eclipse: <http://www.eclipse.org/>



# Programming Environment for Java

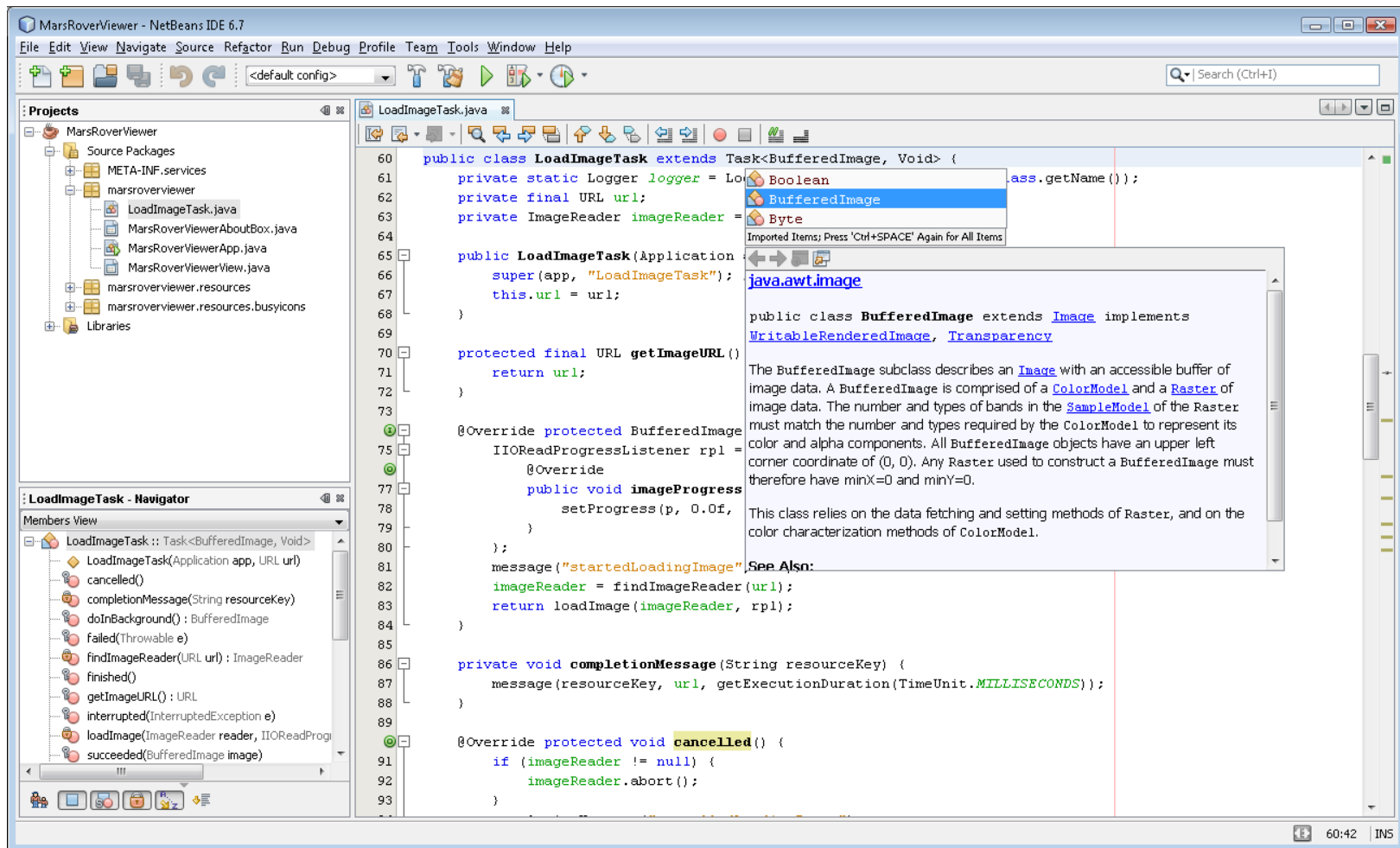
❖ IntelliJ IDEA: <https://www.jetbrains.com/idea/>





# Programming Environment for Java

❖ NetBeans: <http://netbeans.org/>



# Information Sources on Java

---

- ❖ The entry to Java: <http://java.oracle.com/>
- ❖ Tutorials: <http://docs.oracle.com/javase/tutorial/>
- ❖ Java SE 8 documentation:
  - <https://docs.oracle.com/javase/8/docs/>
- ❖ Java SE 8 API documentation:  
<http://docs.oracle.com/javase/8/docs/api/>

# Java SE 8 API Documentation

The screenshot shows a web browser displaying the Java SE 8 API documentation for the `String` class. The browser's address bar shows the URL `docs.oracle.com/javase/8/docs/api/`. The left sidebar, titled "Java™ Platform Standard Ed. 8", contains a "Packages" section with a list of packages including `java.applet`, `java.awt`, and various `String` related classes like `StreamSource`, `StreamSupport`, `StreamTokenizer`, `StrictMath`, `String`, `StringBuffer`, `StringBufferInputStream`, `StringBuilder`, `StringCharacterIterator`, `StringContent`, `StringHolder`, `StringIndexOutOfBoundsException`, `StringJoiner`, `StringMonitor`, `StringMonitorMBean`, `StringNameHelper`, `StringReader`, `StringRefAddr`, `StringSelection`, `StringSeqHelper`, and `StringSeqHolder`. The main content area displays the `java.lang` package and the `Class String`. It shows the inheritance hierarchy: `java.lang.Object` and `java.lang.String`. Under "All Implemented Interfaces:", it lists `Serializable`, `CharSequence`, and `Comparable<String>`. The class declaration is shown as `public final class String`, which `extends Object` and `implements Serializable, Comparable<String>, CharSequence`. A description follows: "The `String` class represents character strings. All string literals in Java programs, such as `"abc"`, are implemented as instances of this class." Another paragraph states: "Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because `String` objects are immutable they can be shared. For example:" This is followed by a code snippet: 

```
String str = "abc";
```

 and the text "is equivalent to:" followed by another code snippet: 

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

# Q & A

---