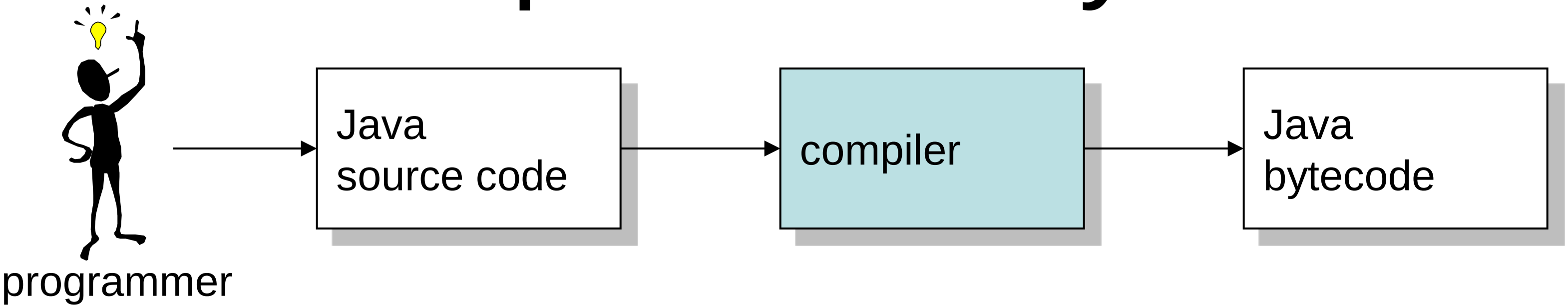
# Java Compilation

# Development lifecycle



Java
source code

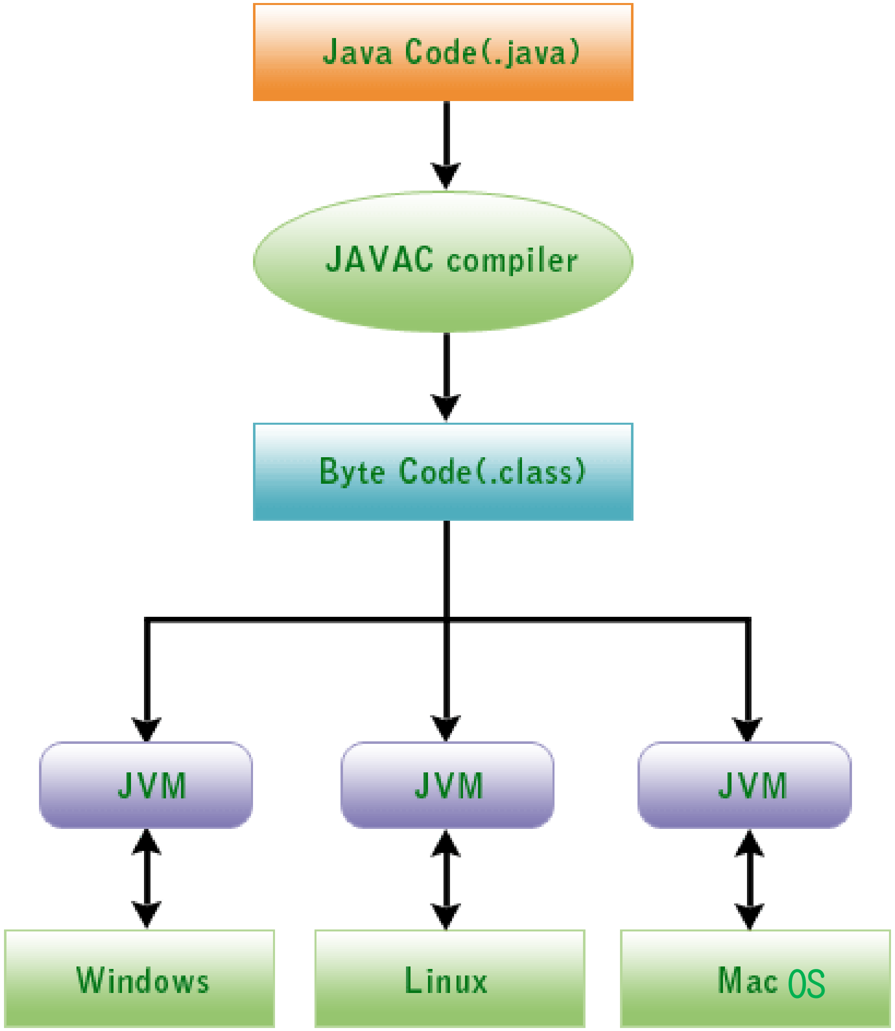
compiler

Java
bytecode

programmer

- Java is a high-level programming language
  - → source code is English-like (syntax is similar to C)
- Java is *compiled* and *interpreted*
  - source code is compiled into bytecode (low-level, platform independent code)
  - bytecode is interpreted (real machine code produced at run time) often
  - → fast and portable (“write once run anywhere”)
- dynamic linking (no link phase at compile time)
  - program consists of class definitions
  - each class is compiled into a separate class file
  - classes may refer to each other, references are resolved at run-time

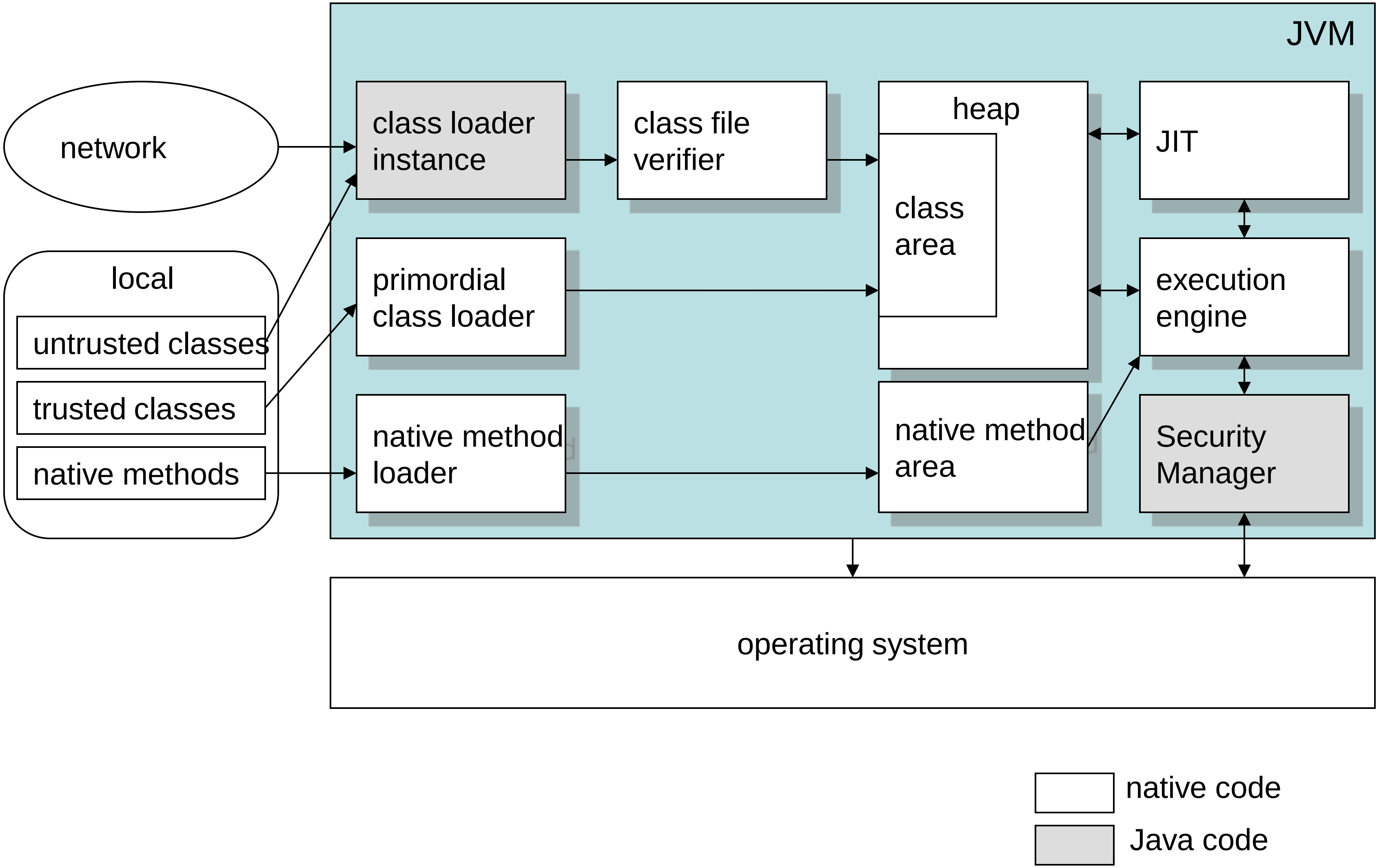
# Write once, run anywhere



# Versions

- JDK 1.0 (January 23, 1996)
- JDK 1.1 (February 19, 1997)
- J2SE 1.2 (December 8, 1998)
- J2SE 1.3 (May 8, 2000)
- J2SE 1.4 (February 6, 2002)
- J2SE 5.0 (September 30, 2004)
- Java SE 6 (December 11, 2006)
- Java SE 7 (July 28, 2011)
- Java SE 8 (March 25, 2014)

# The Java Virtual Machine (JVM)



# What’s a JVM

- A computing machine (just like 8086, but not produced by Intel)
  - Abstract: machine is specified in a book.
  - Concrete: anyone can implement
- Input:
  - A “class” file
  - A search path (to find more class files)
- Output
  - The “execution” of the class file.

# What’s a JVM

- How?
  - It is all in ***The Java Virtual Machine Specification*****, by Tim Lindholm and Frank Yellin**
- Start with?
  - A method named “main” in a given class file.
    - The method must have certain properties
  - Continue execution in other methods as necessary.

# Different JVMs

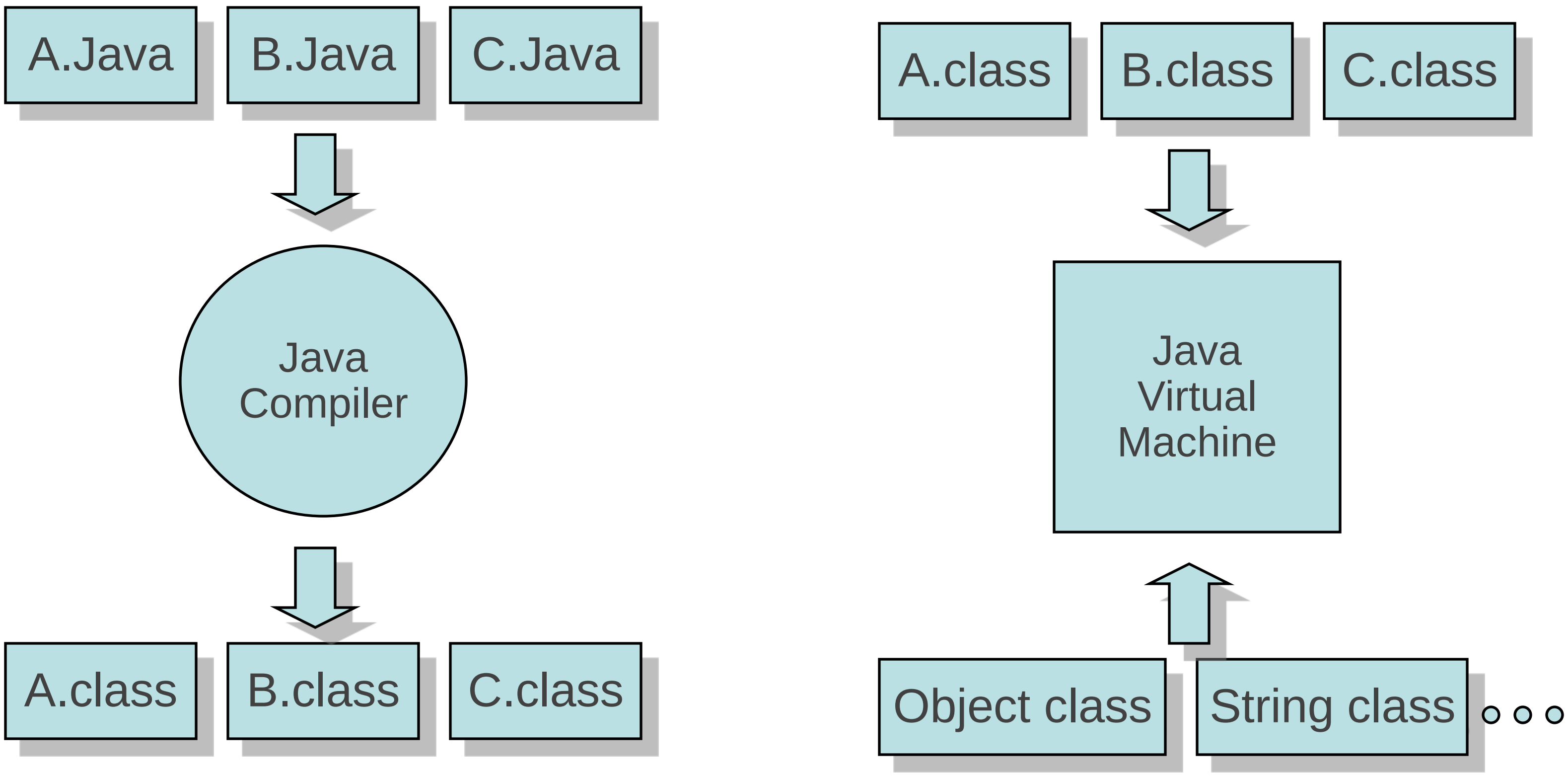
- The primary reference Java VM implementation is HotSpot (Oracle).
- J9 (IBM)
- Unlike Java VMs, which are stack machines, the Dalvik VM (Google) uses a register-based architecture.
- KVM? Nearing death?

# Class File?

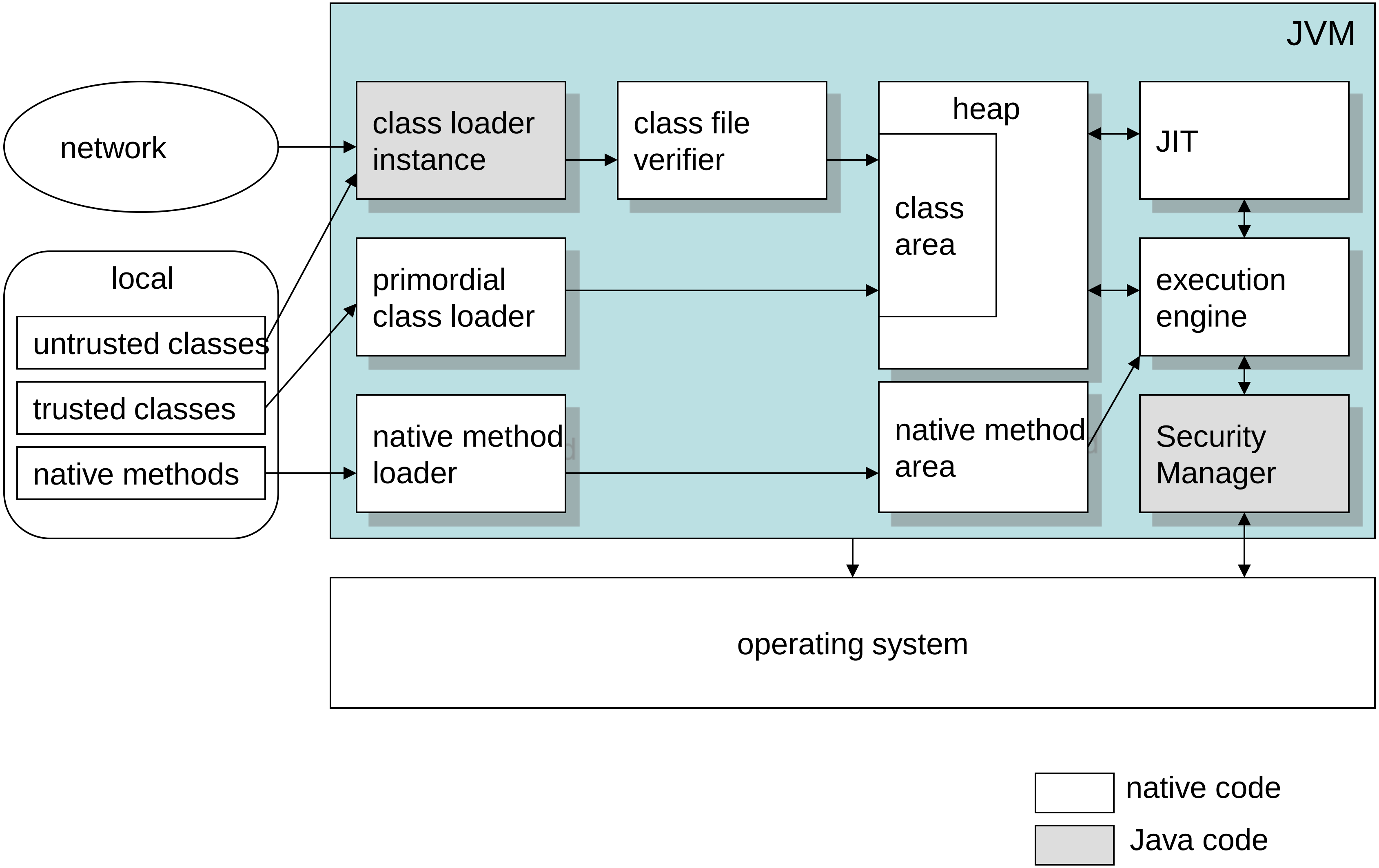
- The binary form for Java programs
- Represents a complete description of one Java class or interface
- Platform independent bytecodes are the machine language of the JVM
- Supports linking code from other languages

# The Java Programming Environment

**Compile time environment** **run time environment**



# The Java Virtual Machine (JVM)



# Programming tools

| Tools | Description |
|---|---|
| **javac** | the Java compiler, which converts source code into Java bytecode |
| **java** | Java interpreter |
| **javadoc** | the documentation generator, which automatically generates documentation from source code comments |
| **javap** | the class file disassembler |
| **jdb** | the debugger |
| **javah** | the C header and stub generator, used to write native methods |

# Compile and Run Java Apps

Your first Java application:

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

# Compile and Run Java Apps

Edit source code:

```
C:\> javac HelloWorld.java
```

Then, a class file is generated:

HelloWorld.class

# Compile and Run Java Apps

Execute the class file:

```
C:\>java HelloWorld
```

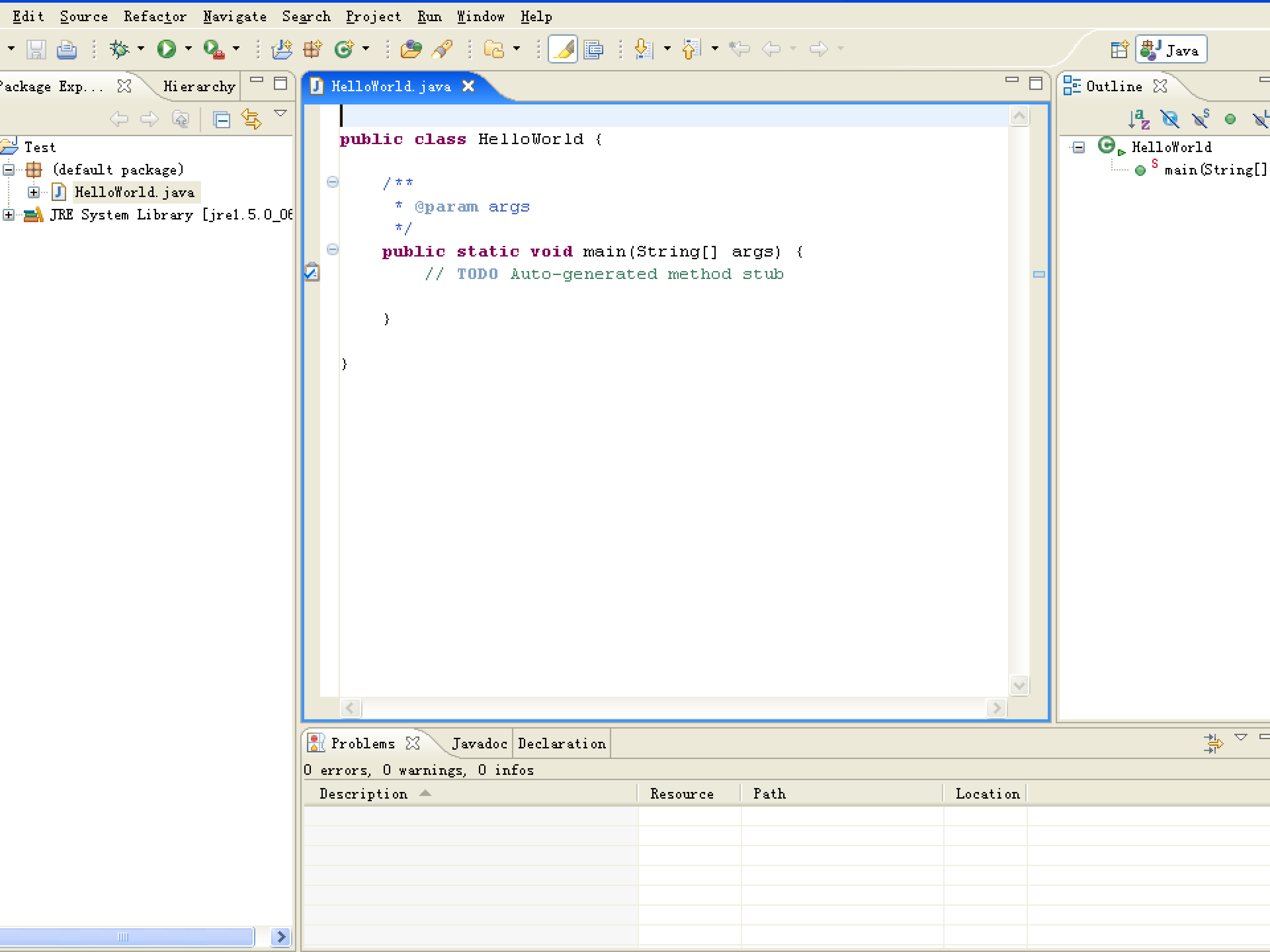
You will see:

```
Hello World!
```

# Eclipse

- Eclipse is a multi-language software development environment comprising of a base workspace and an extensible plug-in system for customizing the environment.
- It provides programmers an Integrated Development Environment (IDE)
- It can be used to develop applications in **Java** and, by means of various plug-ins, other programming languages including C, C++, Ruby , Fortran…

**Everything is a plug-in!**



```
public class HelloWorld {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub

	}

}
```

0 errors, 0 warnings, 0 infos

| Description | Resource | Path | Location |
|---|---|---|---|