Advanced Object-Oriented Programming

Introduction to OOP and Java

Dr. Kulwadee Somboonviwat

International College, KMITL

kskulwad@kmitl.ac.th

Course Objectives

- Solidify **object-oriented programming** skills
- Study the Java Technology
  - The Java Programming Language
  - The Java Platform, Enterprise Edition (Java EE 7)

Key Topics covered in this course

- Fundamentals of Java Programming
- Object-oriented programming concepts
- GUI Programming
- Concurrency
- Java EE 7

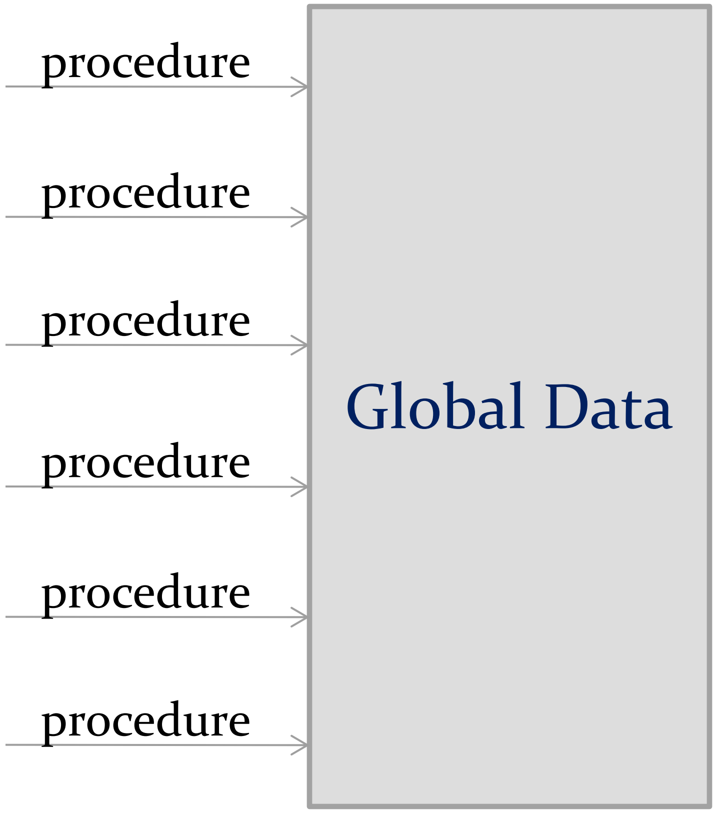
Object-Oriented Programming

- Dominant programming paradigm these days
- A program is made of **objects**.
- Each object
  - exposes specific functionality to the users
  - encapsulates (hides) the implementation of its functionality

Traditional Procedural Programming

- 1970s: "**structured**", procedural programming
  - **Programs = Algorithms + Data** (Niklaus Wirth, 1975)
    - First, we think about a set of procedures (algorithms) needed to solve our problem.
    - Then, we find appropriate ways to store the data
  - Used in C, Pascal, Basic, etc.
  - Structured programming works well for small to medium sized problems

In procedural programming,
- problem is decomposed into **procedures**
- all procedures manipulate a set of **global data**

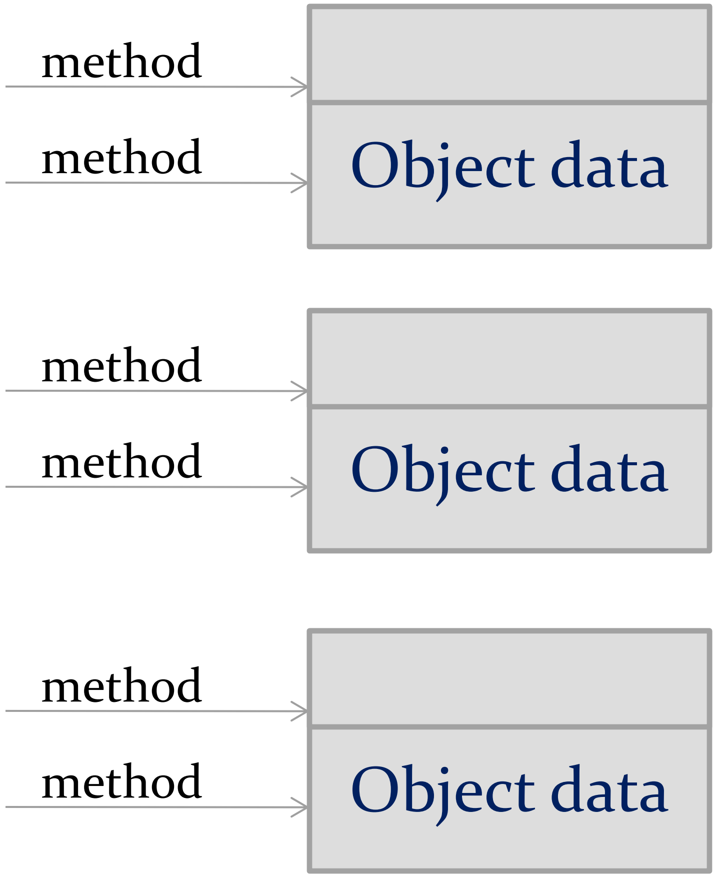


Suppose that ...
- your program has 2,000 procedures
- a piece of data is in an incorrect state

How are you going to find bugs in this situation?

How many procedures you need to search for the culprit?

In object-oriented programming style,
- your program consists of **objects**
- each object has a specific set of **attributes** and **methods**



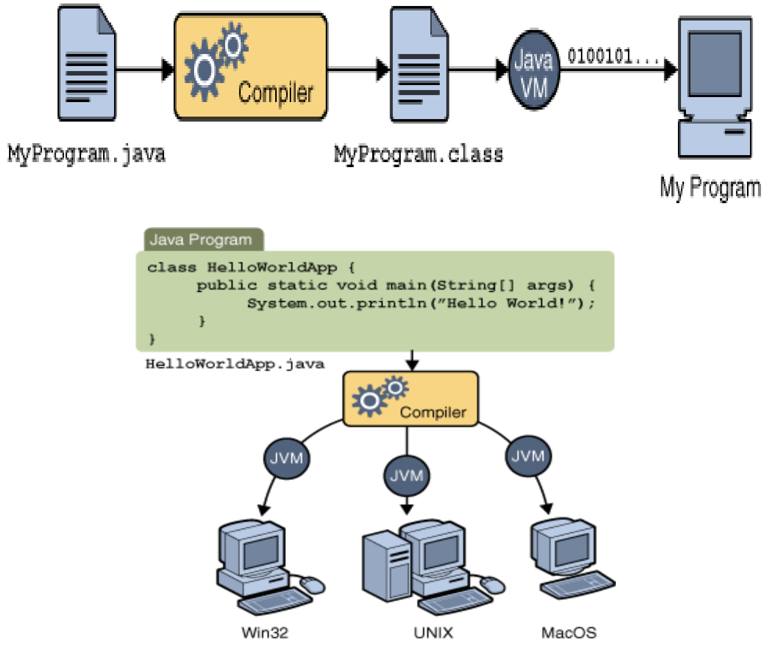
Suppose that …
- your program has 200 objects, and each object has 10 methods.
- a piece of data ***of an object*** is in an incorrect state

How are you going to find bugs in this situation?

How many procedures you need to search for the culprit?

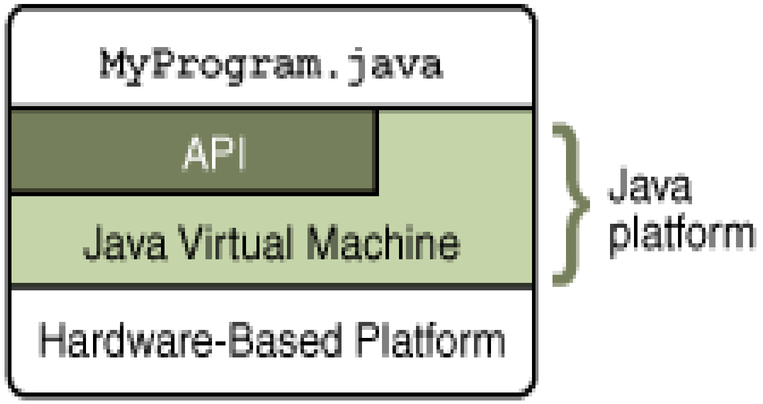
Java Technology

The Java Programming Language



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

The Java Platform



Source: http://download.oracle.com/javase/tutorial/getStarted/intro/definition.html

Characteristics of the Java PL

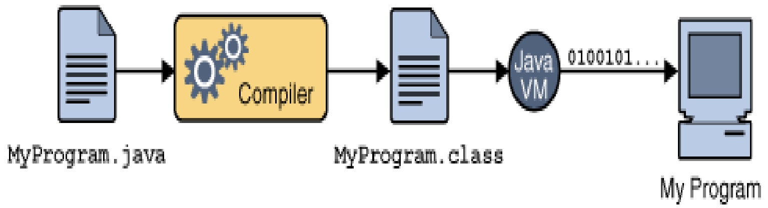
- Simple
- Object oriented
- Distributed
- Multithreaded
- Dynamic
- Architecture neutral
- Portable
- High Performance
- Robust
- Secure

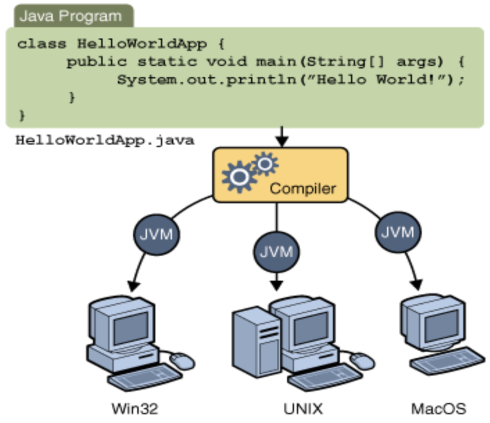
C++ versus Java

Features	Java	C++
Data types	Supports both primitive scalar types and classes	Supports both primitive scalar types and classes
Object allocation	Allocated from **heap**, accessed through reference variables (no pointers)	Allocated from **heap** or **stack**, accessed through reference variables or **pointers**
Object de-allocation	Implicit (garbage collection)	Explicit (delete operator)
Inheritance	Single inheritance only (multiple inheritance is possible with **interfaces**)	Single, Multiple inheritance
Binding	All binding of messages to methods are dynamic except in the case of methods that cannot be overridden	Dynamic binding of messages to methods are **optional (using the virtual keyword)**

Java Technology

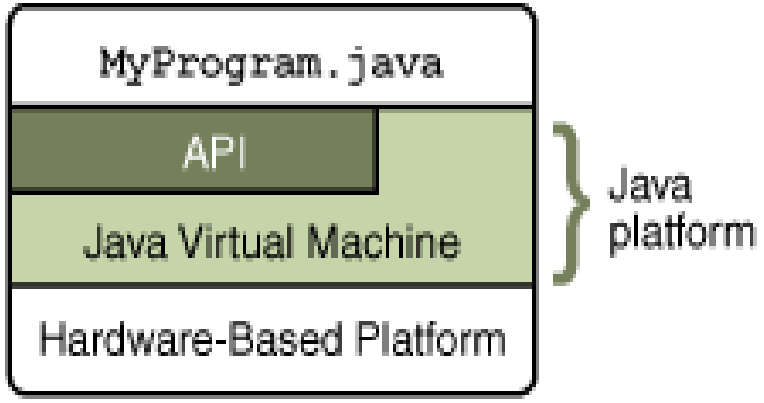
The Java Programming Language





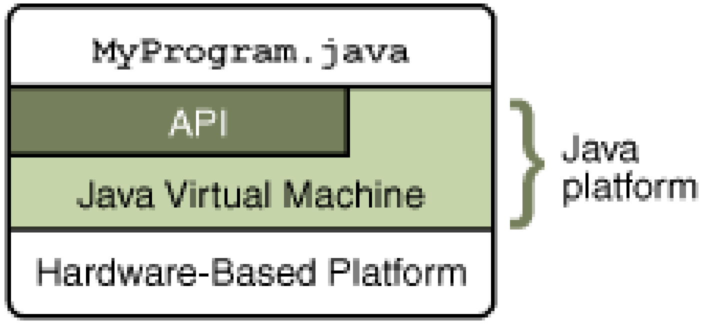
```
class HelloWorldApp {
     public static void main(String[] args) {
          System.out.println("Hello World!");
     }
}
```

The Java Platform



Source: http://download.oracle.com/javase/tutorial/getStarted/intro/definition.html

Java As a Programming Platform



- A *platform* is the hardware or software environment in which a program runs.
  - E.g. Windows, Linux, Solaris OS, and Mac OS
- Java is a software-only platform that runs on top of other hardware-based platforms. It consists of
  - **The Java Virtual Machine**: a software-based processor that presents its own instruction set
  - **The Java Application Programming Interface (API)**

Source: http://download.oracle.com/javase/tutorial/getStarted/intro/definition.html

Different Editions of the Java Platform

- ***Java Platform, Standard Edition (Java SE):***
  - *stand-alone programs that run on* desktops.
  - *applets* (programs that run in the context of a web browser)
- ***Java Platform, Enterprise Edition (Java EE):***
  - built on top of Java SE.
  - enterprise-oriented applications and *servlets* (server programs that conform to Java EE' s Servlet API).
- ***Java Platform, Micro Edition (Java ME):***
  - *MIDlets (programs that run on mobile* information devices)
  - *Xlets (which are programs that run on* embedded devices)

Java Jargon

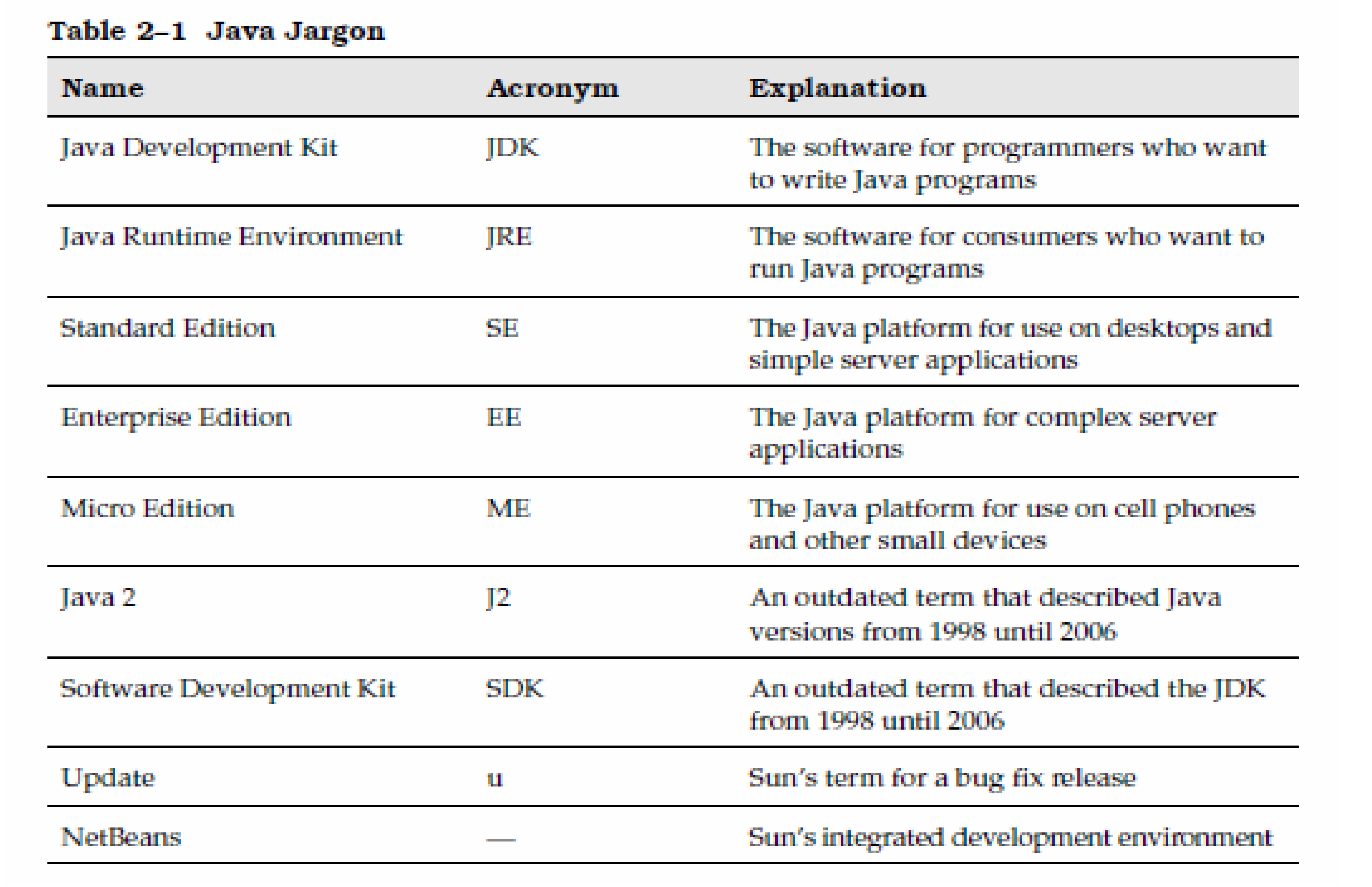
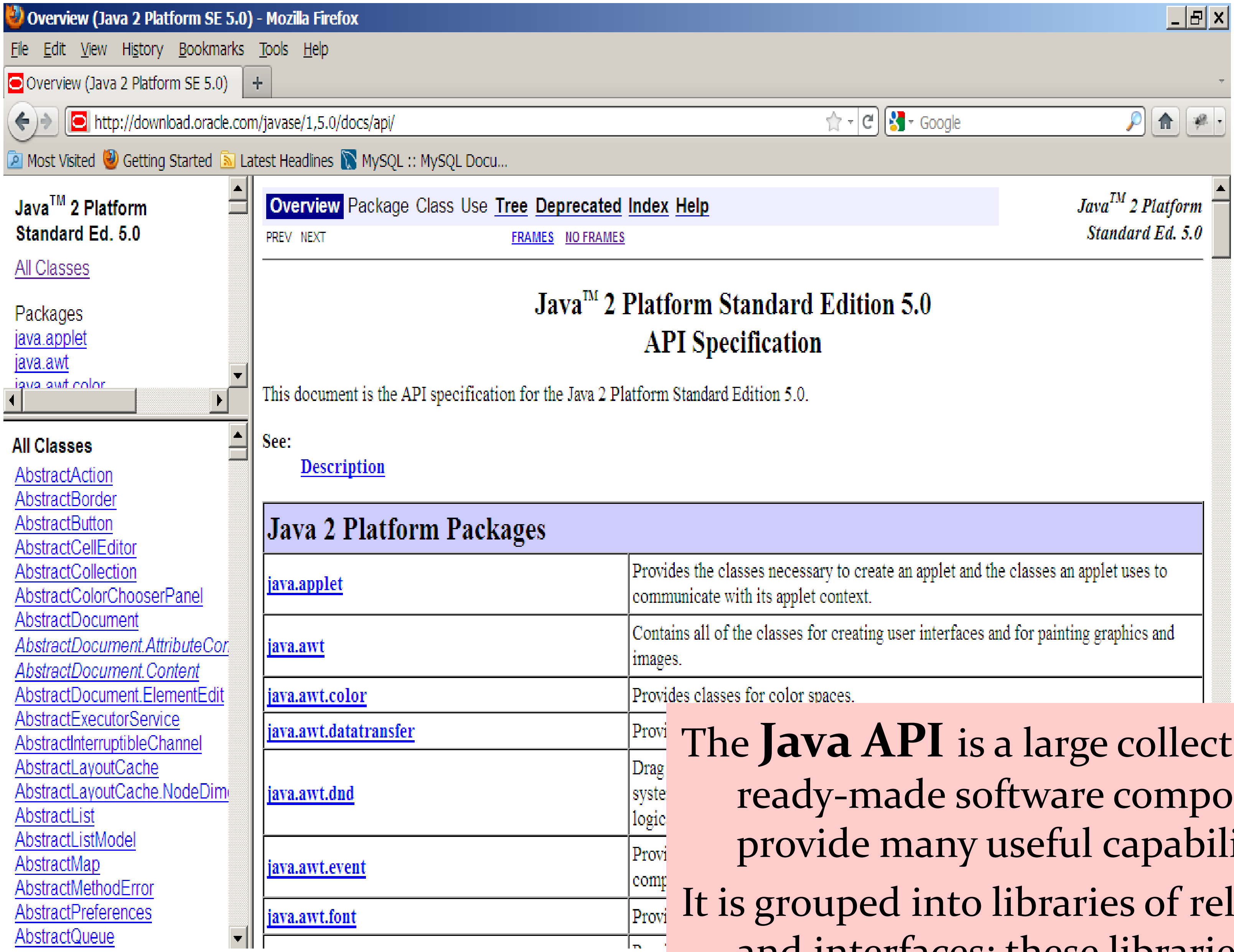


Table 2–1 Java Jargon

Name	Acronym	Explanation
Java Development Kit	JDK	The software for programmers who want to write Java programs
Java Runtime Environment	JRE	The software for consumers who want to run Java programs
Standard Edition	SE	The Java platform for use on desktops and simple server applications
Enterprise Edition	EE	The Java platform for complex server applications
Micro Edition	ME	The Java platform for use on cell phones and other small devices
Java 2	J2	An outdated term that described Java versions from 1998 until 2006
Software Development Kit	SDK	An outdated term that described the JDK from 1998 until 2006
Update	u	Sun's term for a bug fix release
NetBeans	—	Sun's integrated development environment


Overview (Java 2 Platform SE 5.0) - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://download.oracle.com/javase/1,5.0/docs/api/

Java™ 2 Platform Standard Ed. 5.0

All Classes

Packages
java.applet
java.awt

All Classes
AbstractAction
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractCollection
AbstractColorChooserPanel
AbstractDocument
AbstractDocument.AttributeCor
AbstractDocument.Content
AbstractDocument.ElementEdit
AbstractExecutorService
AbstractInterruptibleChannel
AbstractLayoutCache
AbstractLayoutCache.NodeDim
AbstractList
AbstractListModel
AbstractMap
AbstractMethodError
AbstractPreferences
AbstractQueue

Overview Package Class Use Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

Java™ 2 Platform Standard Ed. 5.0

Java™ 2 Platform Standard Edition 5.0
API Specification

This document is the API specification for the Java 2 Platform Standard Edition 5.0.

See:
Description

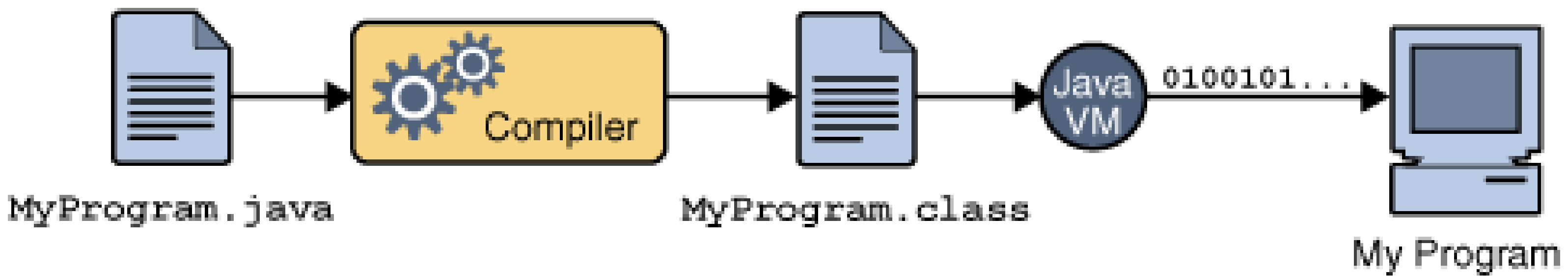
Java 2 Platform Packages	
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provi
java.awt.dnd	Drag syste logic
java.awt.event	Provi comp
java.awt.font	Provi

The **Java API** is a large collection of ready-made software components that provide many useful capabilities.

It is grouped into libraries of related classes and interfaces; these libraries are known as *packages*

Source: http://download.oracle.com/javase/tutorial/getStarted/intro/definition.html

Java Software Development Process



- Write the source code and save in files with **.java** extension
- Compile the source code into **.class** files using the **javac compiler**
- A **.class** file contains **bytecodes**

  (the machine language of the Java Virtual Machine (Java VM)
- Run the application (with an instance of the Java VM)

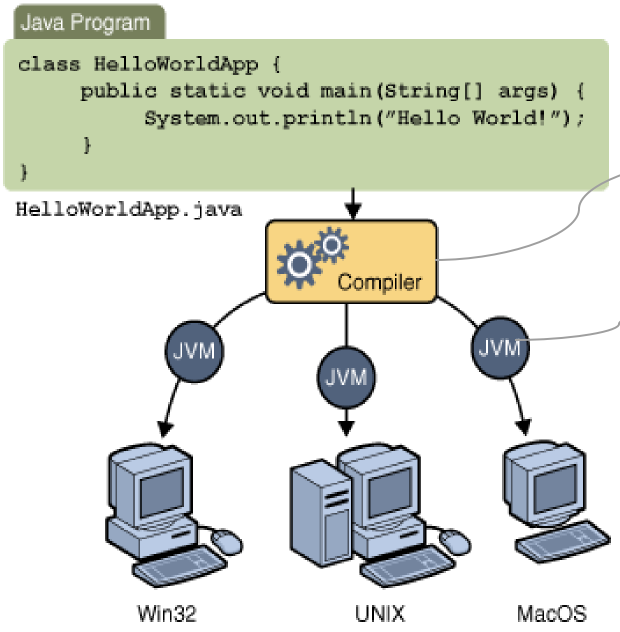
  using the **java** launcher tool.

Source: http://download.oracle.com/javase/tutorial/getStarted/intro/definition.html

Java Program Execution

- The **java** tool loads and starts the VM, and passes the program' s main classfile (.class) to the machine
- The VM uses **classloader** to load the classfile
- The VM' s **bytecode verifier** checks that the classfile' s bytecode is valid and does not compromise security
  - If the bytecode has any problem, the verifier terminates the VM
- If all is well with the bytecode, *the VM 's **interpreter** interprets the bytecode one instruction at a time*

* *Interpretation consists of identifying* bytecode instructions , and executing equivalent native instructions (instructions understood by the physical processor)



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

```
$ javac HelloWorldApp.java
```

$ java HelloWorldApp

(1)Load the JVM

(2)classloader loads HelloWorldApp.class

(3)bytecode verifier check that the classfile is valid and secure

(4)If all is well, the **interpreter** interpret the bytecode

(5)A section of frequently executed bytecode will be compiled to native code by the **JIT (Just In Time) compiler**

- The Java platform provides an abstraction over the underlying hardware/OS platform
  - **Portability**: the same .class files can run unchanged on a variety of hardware platforms and operating systems

What can Java Technology Do?

- Development Tools: javac, java, javadoc
- Rich APIs
- Deployment Technologies: Web Start, Java Plug-In
- User Interface Toolkits
- Integration libraries: JDBC, JNDI, RMI