Module 1

Getting Started

Objectives

- Describe the key features of Java technology
- Write, compile, and run a simple Java technology application
- Describe the function of the Java Virtual Machine (JVM™)
- Define garbage collection
- List the three tasks performed by the Java platform that handle code security

NOTE: The terms “Java Virtual Machine” and “JVM” mean a Virtual Machine for the Java™ platform.

What Is the Java™ Technology?

- Java technology is:
  - A programming language
  - A development environment
  - An application environment
  - A deployment environment
- It is similar in syntax to C++.
- It is used for developing both *applets* and *applications*.

Primary Goals of the Java Technology

- Provides an easy-to-use language by:
  - Avoiding many pitfalls of other languages
  - Being object-oriented
  - Enabling users to create streamlined and clear code
- Provides an interpreted environment for:
  - Improved speed of development
  - Code portability

Primary Goals of the Java Technology

- Enables users to run more than one thread of activity
- Loads classes dynamically; that is, at the time they are actually needed
- Supports changing programs dynamically during runtime by loading classes from disparate sources
- Furnishes better security

Primary Goals of the Java Technology

The following features fulfill these goals:

- The Java Virtual Machine (JVM™)[1]
- Garbage collection
- The Java Runtime Environment (JRE)
- JVM tool interface

1. The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java platform

The Java Virtual Machine

- Provides hardware platform specifications
- Reads compiled byte codes that are platform-independent
- Is implemented as software or hardware
- Is implemented in a Java technology development tool or a Web browser

The Java Virtual Machine

JVM provides definitions for the:

- Instruction set (central processing unit [CPU])
- Register set
- Class file format
- Stack
- Garbage-collected heap
- Memory area
- Fatal error reporting
- High-precision timing support

The Java Virtual Machine

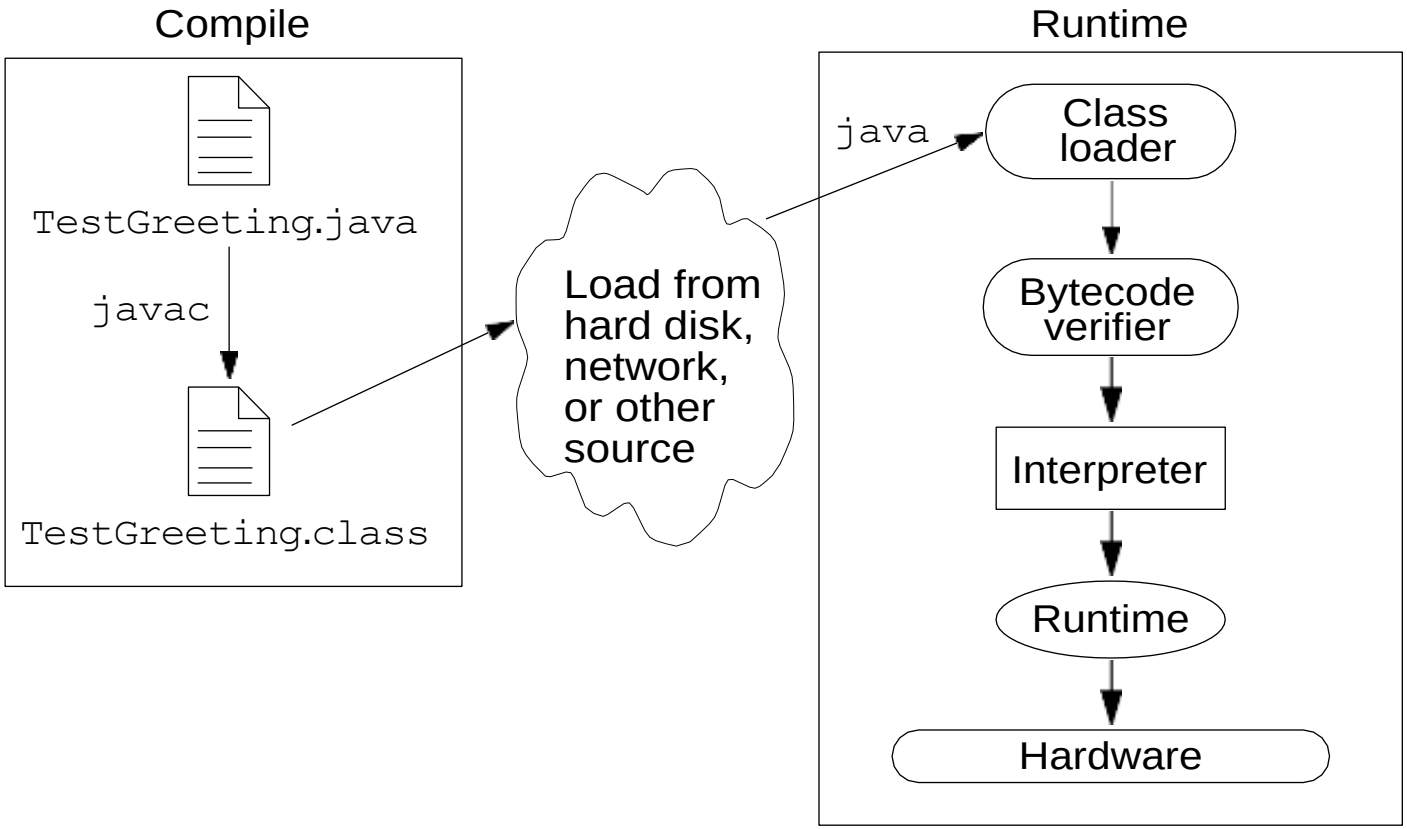
- The majority of type checking is done when the code is compiled.
- Implementation of the JVM approved by Sun Microsystems must be able to run any compliant class file.
- The JVM executes on multiple operating environments.

Garbage Collection

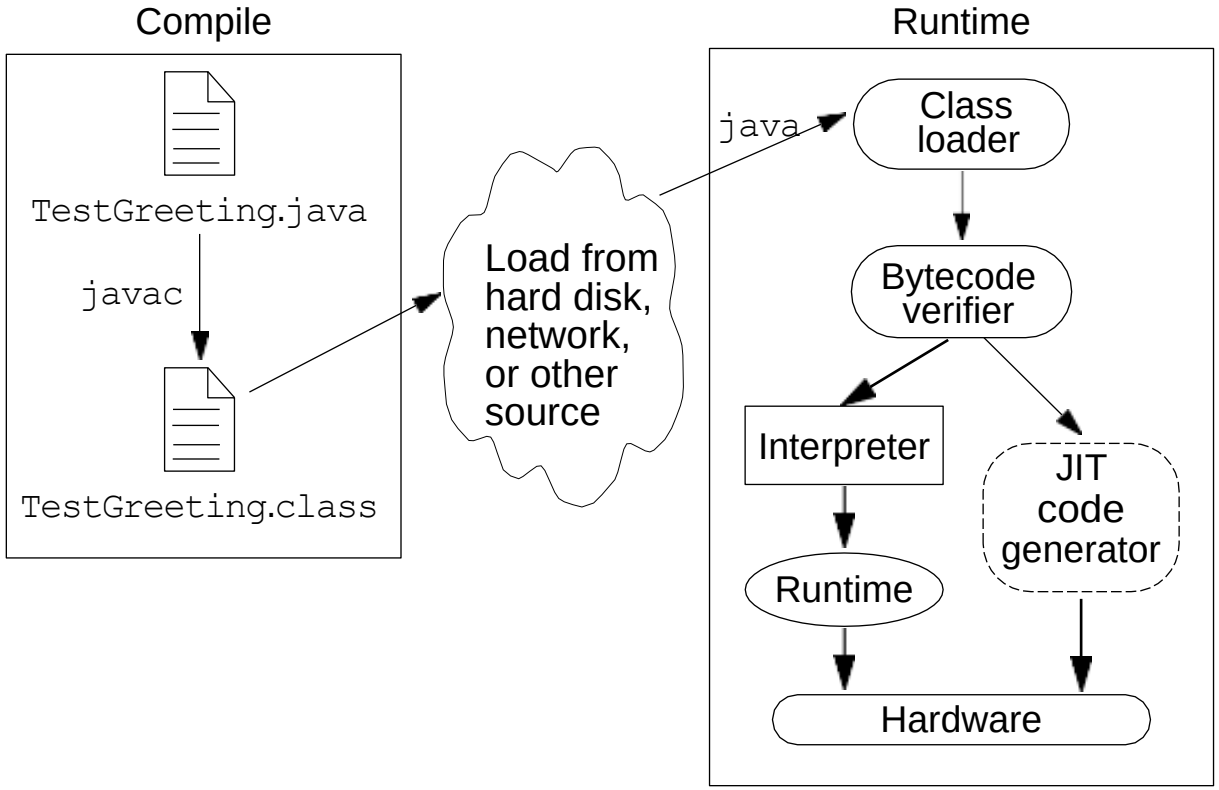
- Allocated memory that is no longer needed should be deallocated.
- In other languages, deallocation is the programmer's responsibility.
- The Java programming language provides a system-level thread to track memory allocation.
- Garbage collection has the following characteristics:
  - Checks for and frees memory no longer needed
  - Is done automatically
  - Can vary dramatically across JVM implementations

The Java Runtime Environment

The Java application environment performs as follows:



Operation of the JRE With a Just-In-Time (JIT) Compiler



JVM™ Tasks

The JVM performs three main tasks:

- Loads code
- Verifies code
- Executes code

The Class Loader

- Loads all classes necessary for the execution of a program
- Maintains classes of the local file system in separate *namespaces*
- Prevents spoofing

The Bytecode Verifier

Ensures that:

- The code adheres to the JVM specification.
- The code does not violate system integrity.
- The code causes no operand stack overflows or underflows.
- The parameter types for all operational code are correct.
- No illegal data conversions (the conversion of integers to pointers) have occurred.

Java Programming language Features

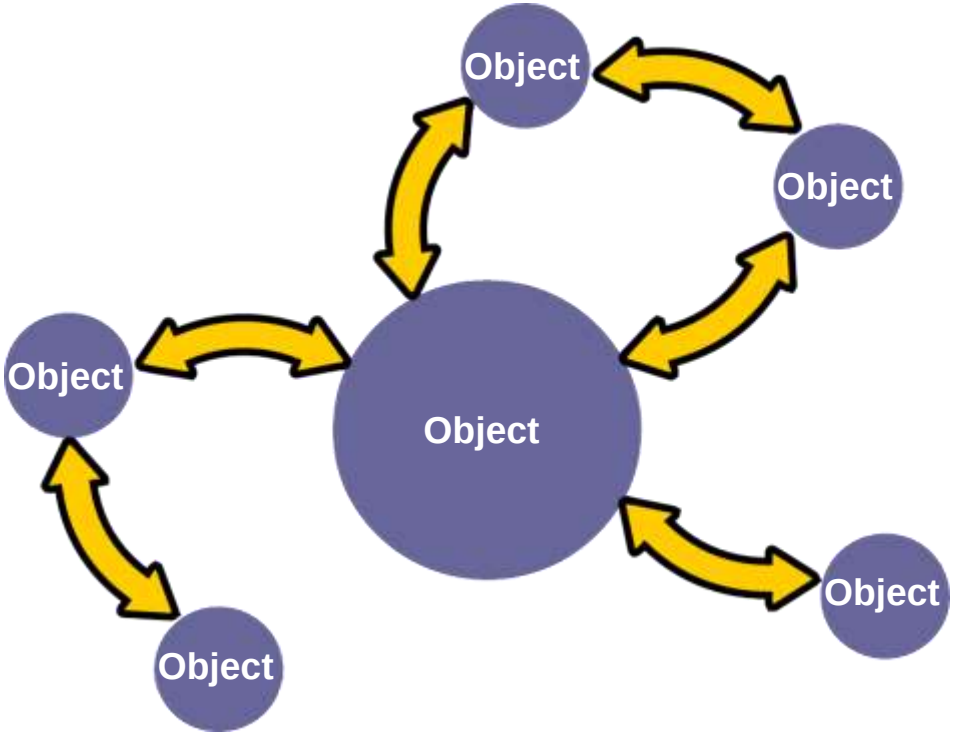
Key Concepts of the Java Programming Language

- Simple
- Object-oriented
- Distributed
- Multithreaded
- Dynamic
- Platform-independent/Architecture neutral
- Portable
- High performance
- Robust
- Secure

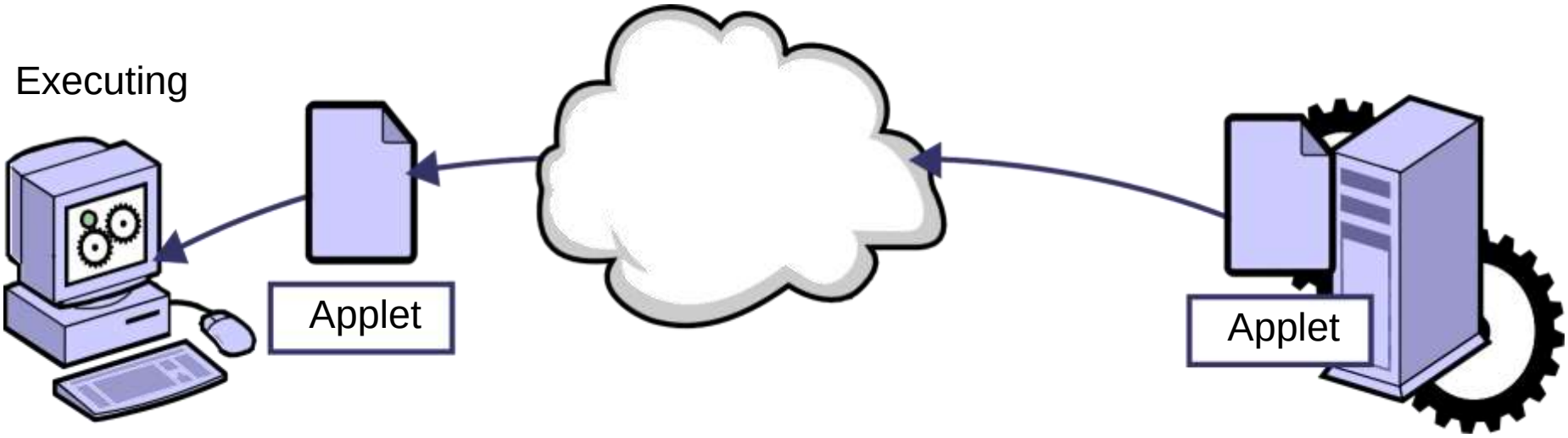
Procedural Programming

Procedural programming focuses on sequence.

Object-Oriented



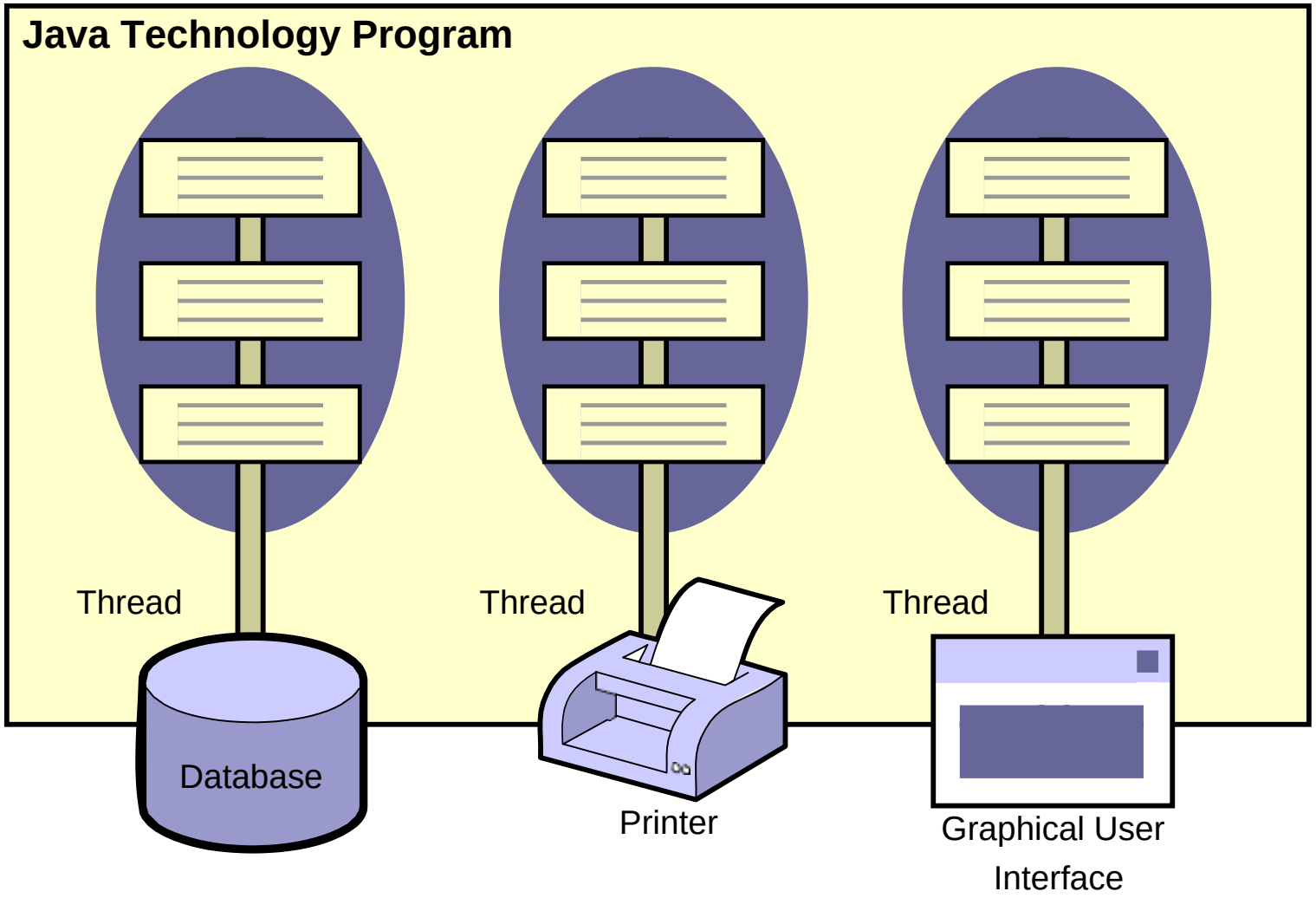
Distributed



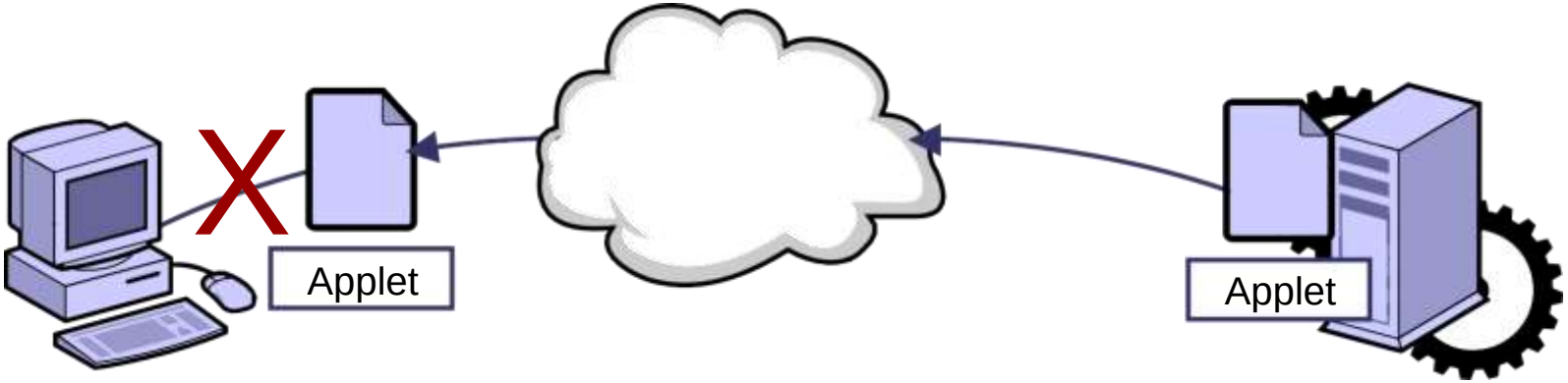
Simple

- References are used instead of memory pointers.
- A `boolean` data type can have a value of either `true` or `false`.
- Memory management is automatic.

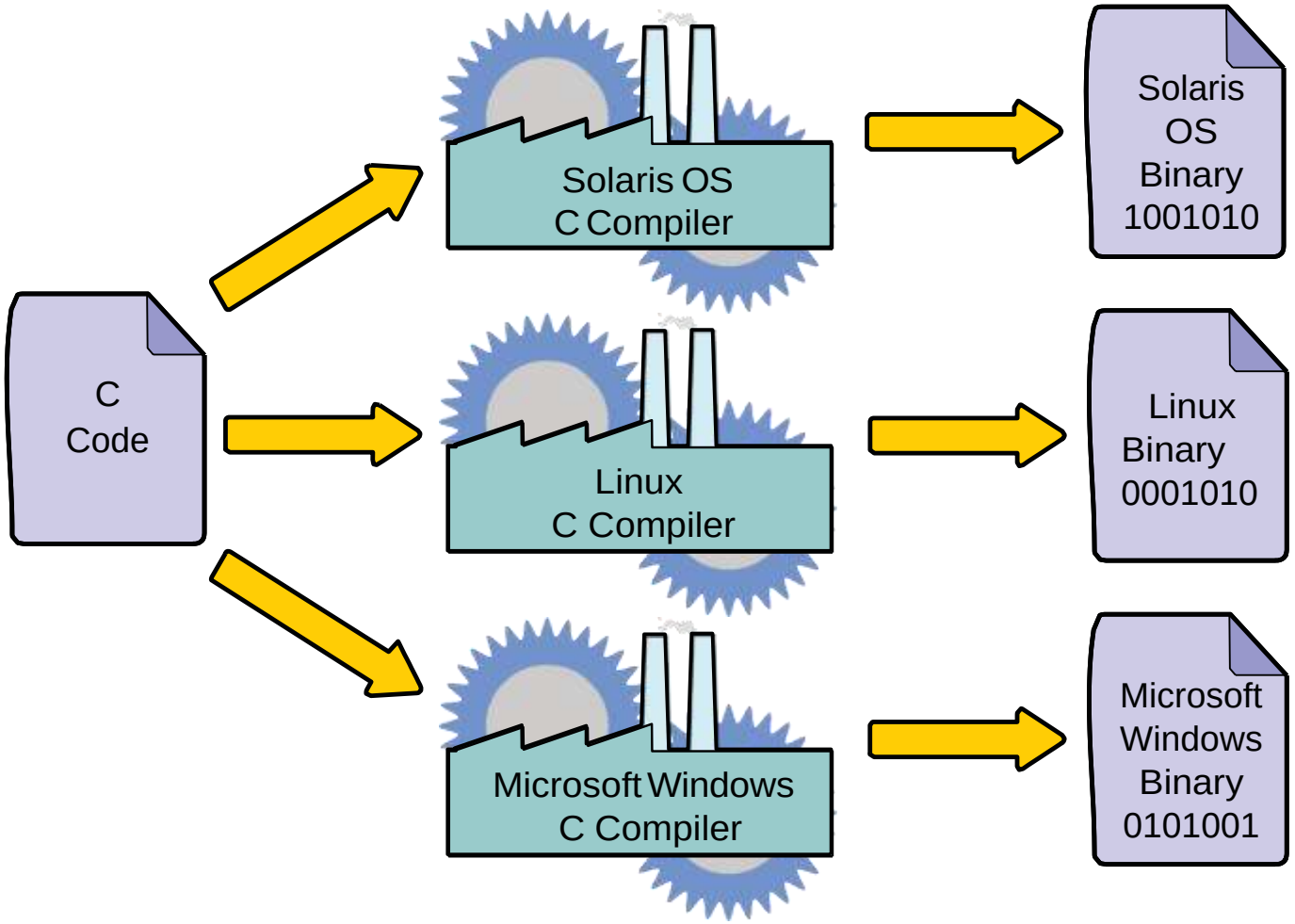
Multithreaded



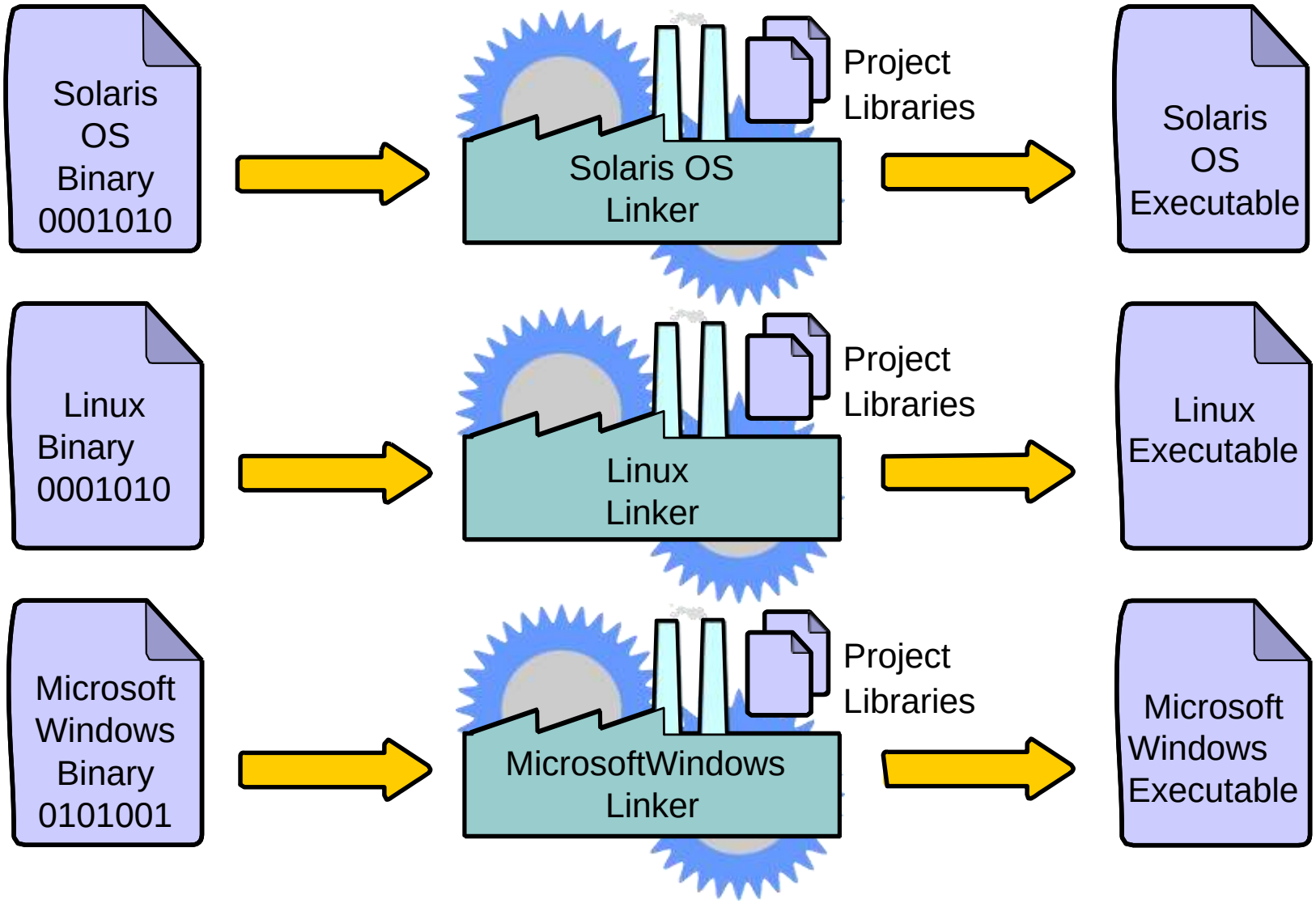
Secure



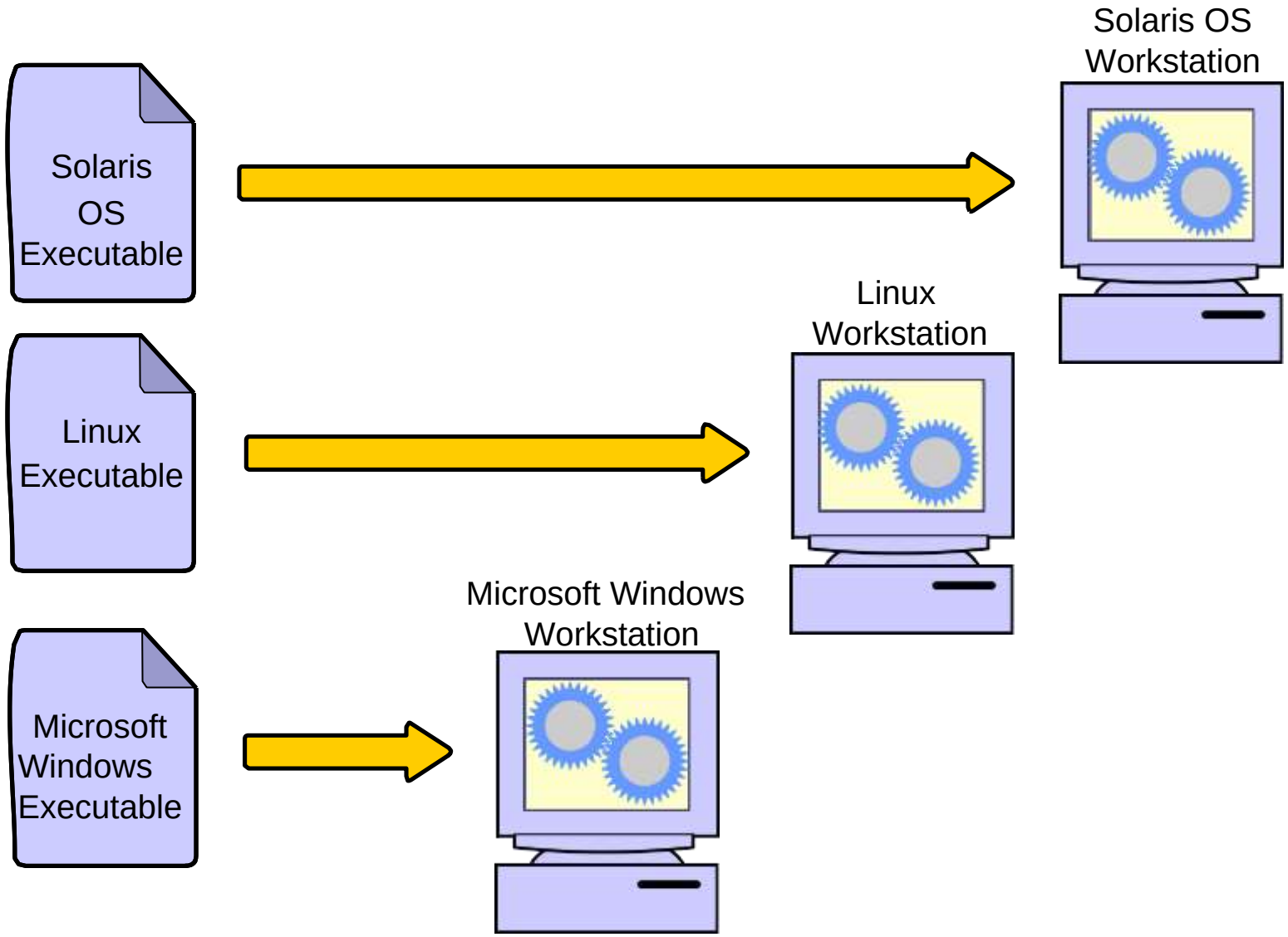
Platform-Dependent Programs



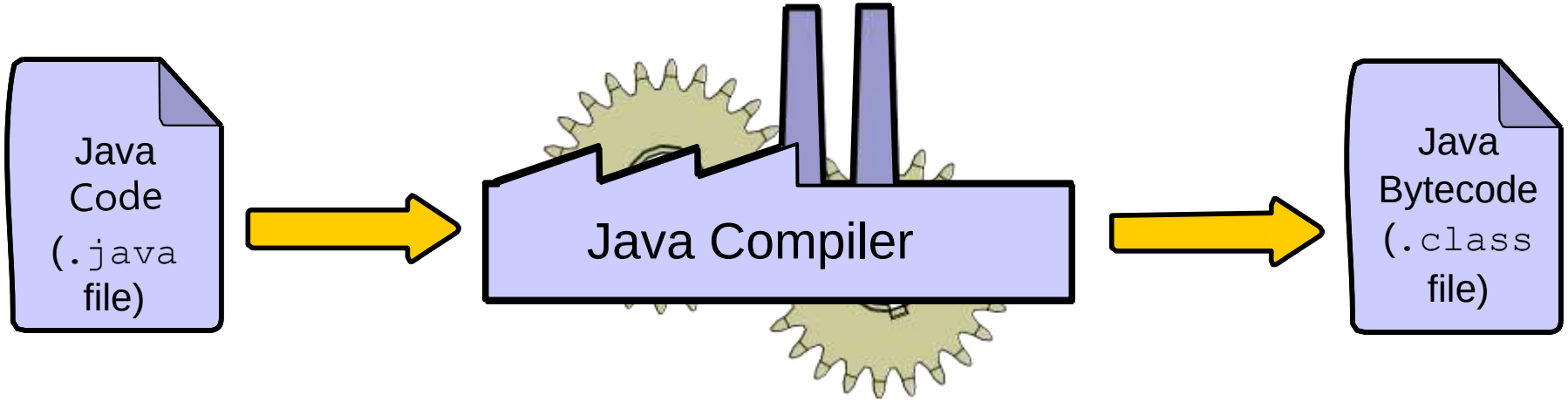
Platform-Dependent Programs



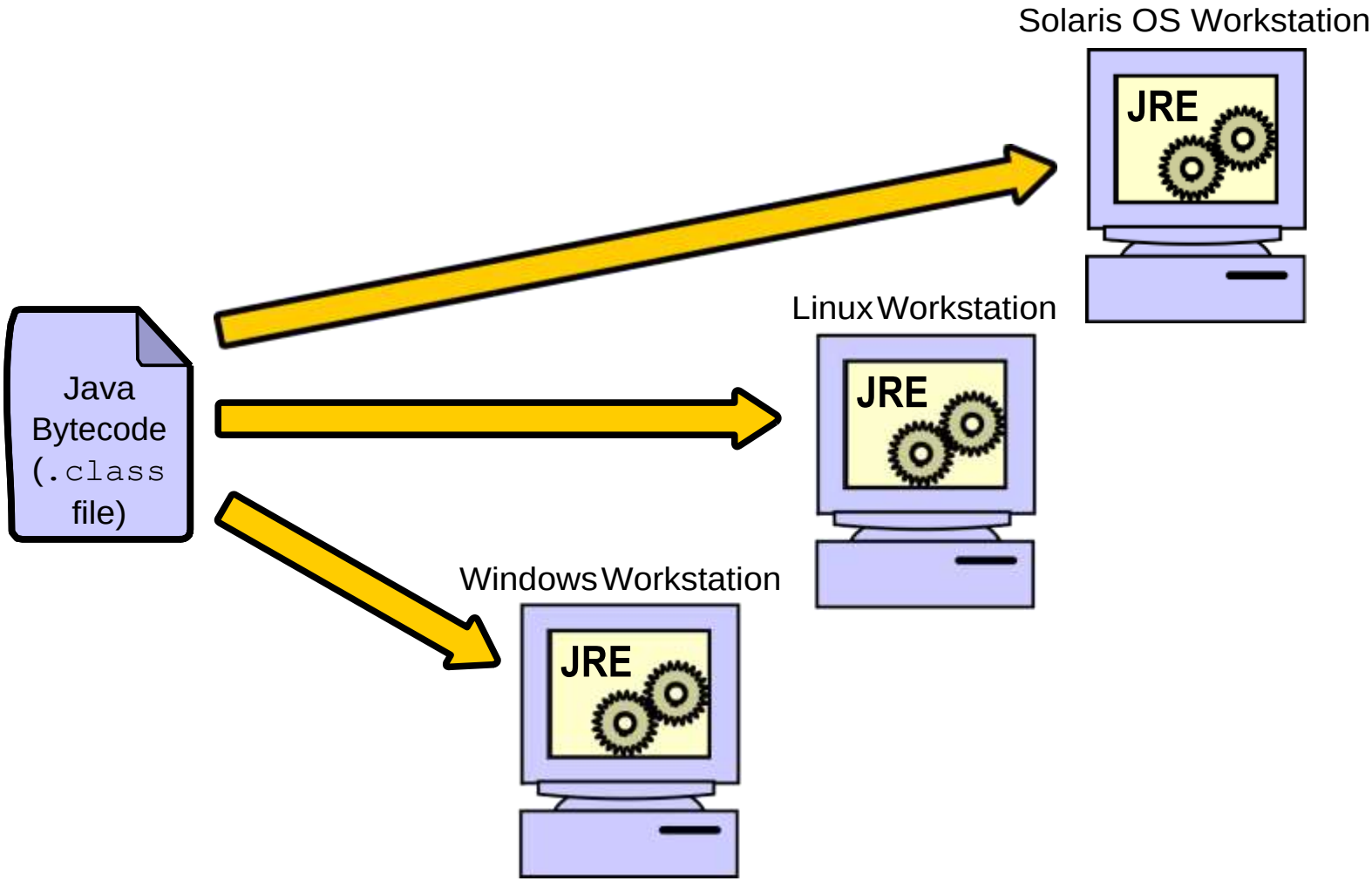
Platform-Dependent Programs



Platform-Independent Programs

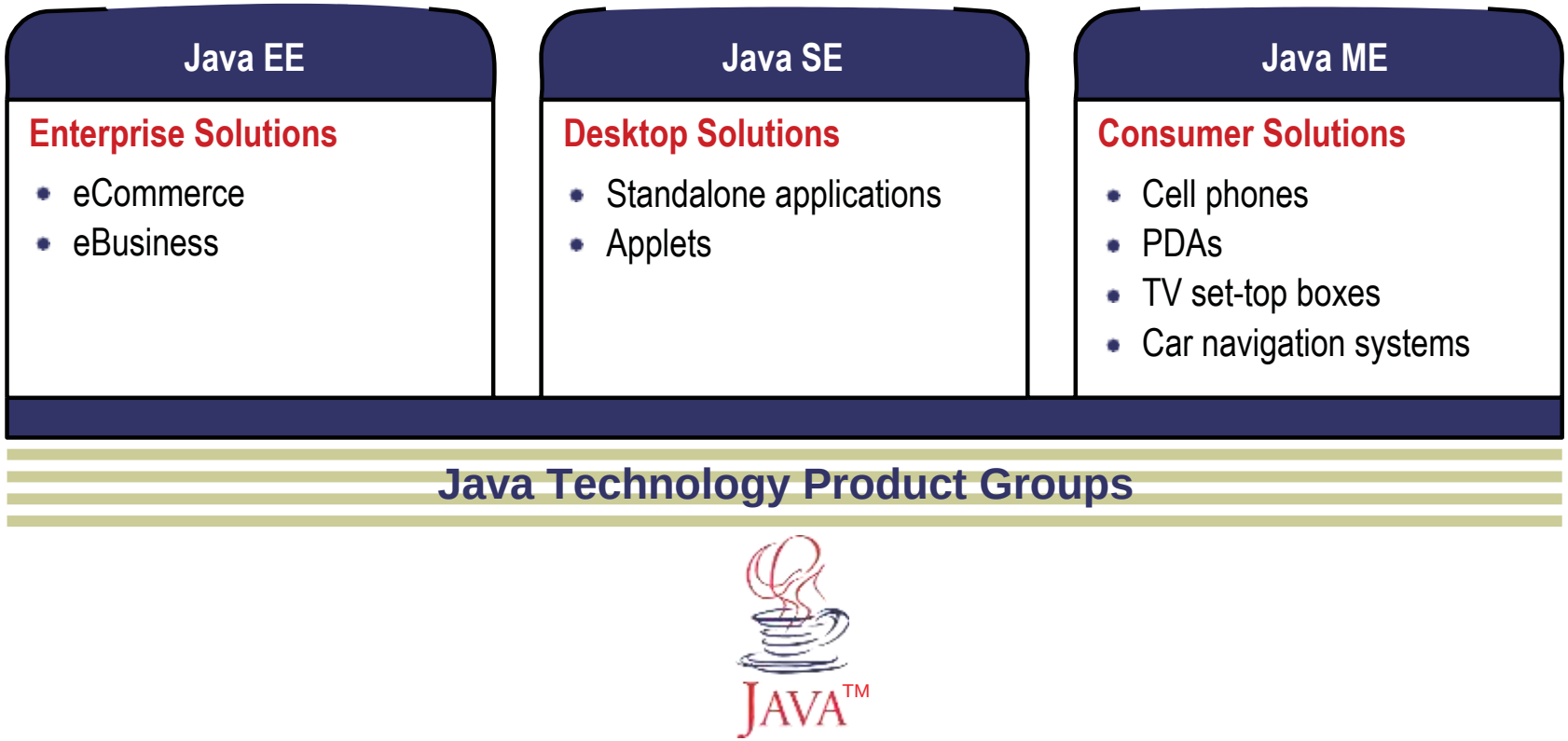


Platform-Independent Programs



Java Technology
3 Product-Groups

Identifying Java Technology Product Groups



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
Server JRE	—	The software for running Java programs on servers
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 small devices
JavaFX	—	An alternate toolkit for graphical user interfaces that is included with certain Java SE distributions prior to Java 11
OpenJDK	—	A free and open source implementation of Java SE
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	Oracle's term for a bug fix release up to Java 8
NetBeans	—	Oracle's integrated development environment

Using the Java Platform, Standard Edition SDK Components

- Java runtime environment (JRE):
  - A Java Virtual Machine (JVM™) for the platform you choose
  - Java class libraries for the platform you choose
- A Java technology compiler
- Java class library (API) documentation (as a separate download)
- Additional utilities, such as utilities for creating Java archive files (JAR files) and for debugging Java technology programs
- Examples of Java technology programs

Java Programming Environment

Installing the JDK

- Download the appropriate JDK installation exe or gzip tarball file for your Windows, Solaris, or Linux platform
  - https://www.oracle.com/technetwork/java/javase/downloads/index.html
- run the Windows executable or unarchive the Solaris/ Linux gzip tarball
- modify environment variable
  - **PATH** to include the JDK tools **bin**
  - **JAVA_HOME** environment variable to point to JDK 's home directory

Directory Structure of JDK

Environment Variable

- Windows
  - system properties dialog->Advanced tab->Environment button->System Variables “Path”, Add the *jdk\bin directory to the beginning of the path*
  - c:*jdk\bin;other stuff*
- In UNIX (including Linux)
  - C shell ~/.cshrc file:
    - set path=(/usr/local/*jdk/bin $path)*
  - Bourne Again shell ~/.bashrc or ~/.bash_profile file:
    - export PATH=/usr/local/*jdk/bin:$PATH*

Command-Line Tools

- Java
  - a tool for running applications
- Javac
  - a tool that launches the Java compiler to compile one or more source files
- javadoc
  - a tool that generates special HTML-based documentation from Javadoc comments
- jar
  - a tool for packaging classfiles and resource files into special ZIP files with ".jar" file extensions

Documentation

- **Documentation**
  - You can download the documentation from https://www.oracle.com/technetwork/java/javase/downloads/index.html
  - jar xvf jdk-*version-doc.zip*

Integrated Development Environment

- Eclipse
  - eclpse.org
  - the most commonly used, origin from IBM
- Net**b**eans
  - netbeans.org
  - Apache free IDE
- Other
  - IntelliJ IDEA, Visual Studio Code …

First Java Program

A Simple Java Application

The `TestGreeting.java` Application

```
1   //
2   // Sample "Hello World" application
3   //
4   public class TestGreeting{
      5     public static void main (String[] args) {
6       Greeting hello = new Greeting();

7       hello.greet();
    }
8     }
```

The `Greeting.java` Class

```
9
1   public class Greeting
2     { public void
3     greet() {
4       System.out.println(“hi”);
      }
5
```

The `TestGreeting` Application

- Comment lines
- Class declaration
- The `main` method
- Method body

The `Greeting` Class

- Class declaration
- The `greet` method

Compiling and Running the `TestGreeting` Program

- Compile `TestGreeting.java`:

  `javac TestGreeting.java`
- The `Greeting.java` is compiled automatically.
- Run the application by using the following command:

  `java TestGreeting`
- Locate common compile and runtime errors.

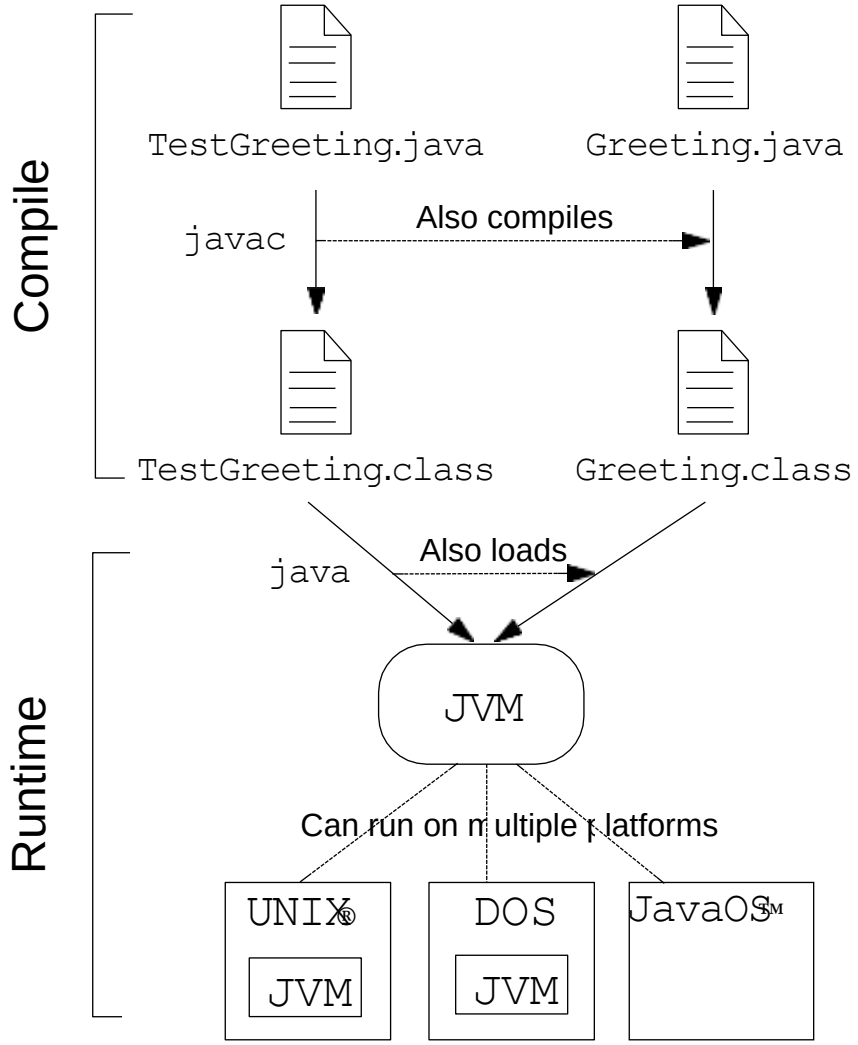
Compile-Time Errors

- `javac: Command not found`
- `Greeting.java:4: cannot resolve symbol`
  `symbol  : method printl  (java.lang.String)`
  `location: class java.io.PrintStream Sys-`
  `tem.out.printl("hi");`
  `^`
- `TestGreet.java:4: Public class TestGreeting must be defined in a file called "TestGreeting.java".`

Runtime Errors

- `Can't find class TestGreeting`
- `Exception in thread "main" java.lang.NoSuchMethodError: main`

Java Technology Runtime Environment



Summary

- Java™ Technology
- Java Virtual Machine
- Garbage Collection
- Java Runtime Environment
- Java Programming language
- Java Product-Groups
- Java Programming Environment
- First Java Program