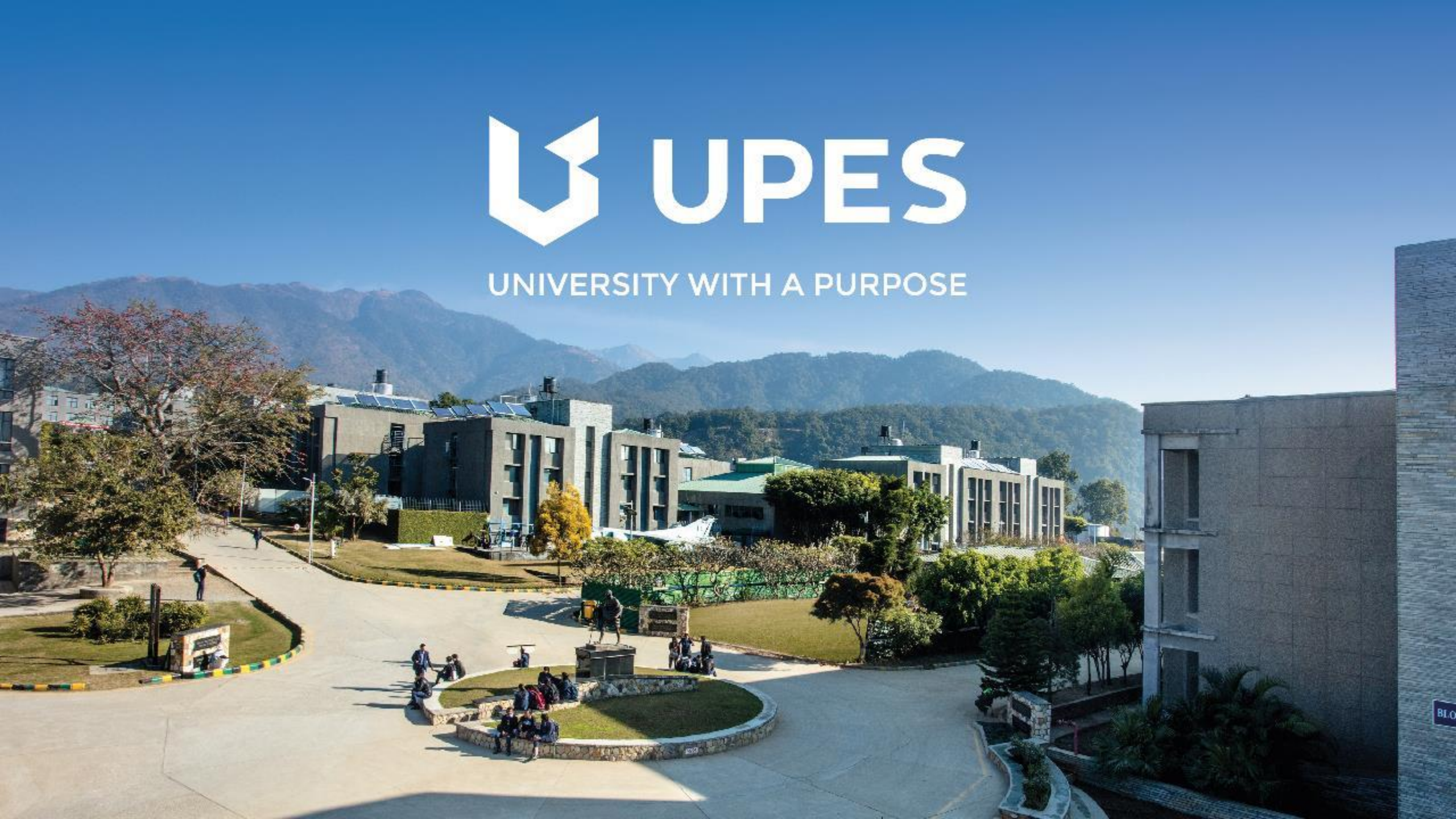

UPES
UNIVERSITY WITH A PURPOSE

Introduction to Java

Introduction to Java

- An always young programming language:

A simple, object oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high performance, multithreaded, dynamic language.

From: *Java: An Overview*
James Gosling, Sun Microsystems, February 1995.

Introduction to Java

- According to Gosling:

- "An environment"
- "A platform"
- "A way of thinking"

History of Java

- 1993 Oak project at Sun
- small, robust, architecture independent, Object-Oriented, language.
- 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
- 2004 Java 1.5 (parameterized types, enum, …) (Java5!)
- 2005 J2EE 1.5

Features of Java

- Java is a general-purpose, high-level programming language.

The features of Java

- Java program is both compiled and interpreted.
- Write once, run anywhere
- Java is a software-only platform running on top of other, hardware-based platforms.
- Java Virtual Machine (Java VM)
- The Java Application Programming Interface (JAVA API)

Features of Java

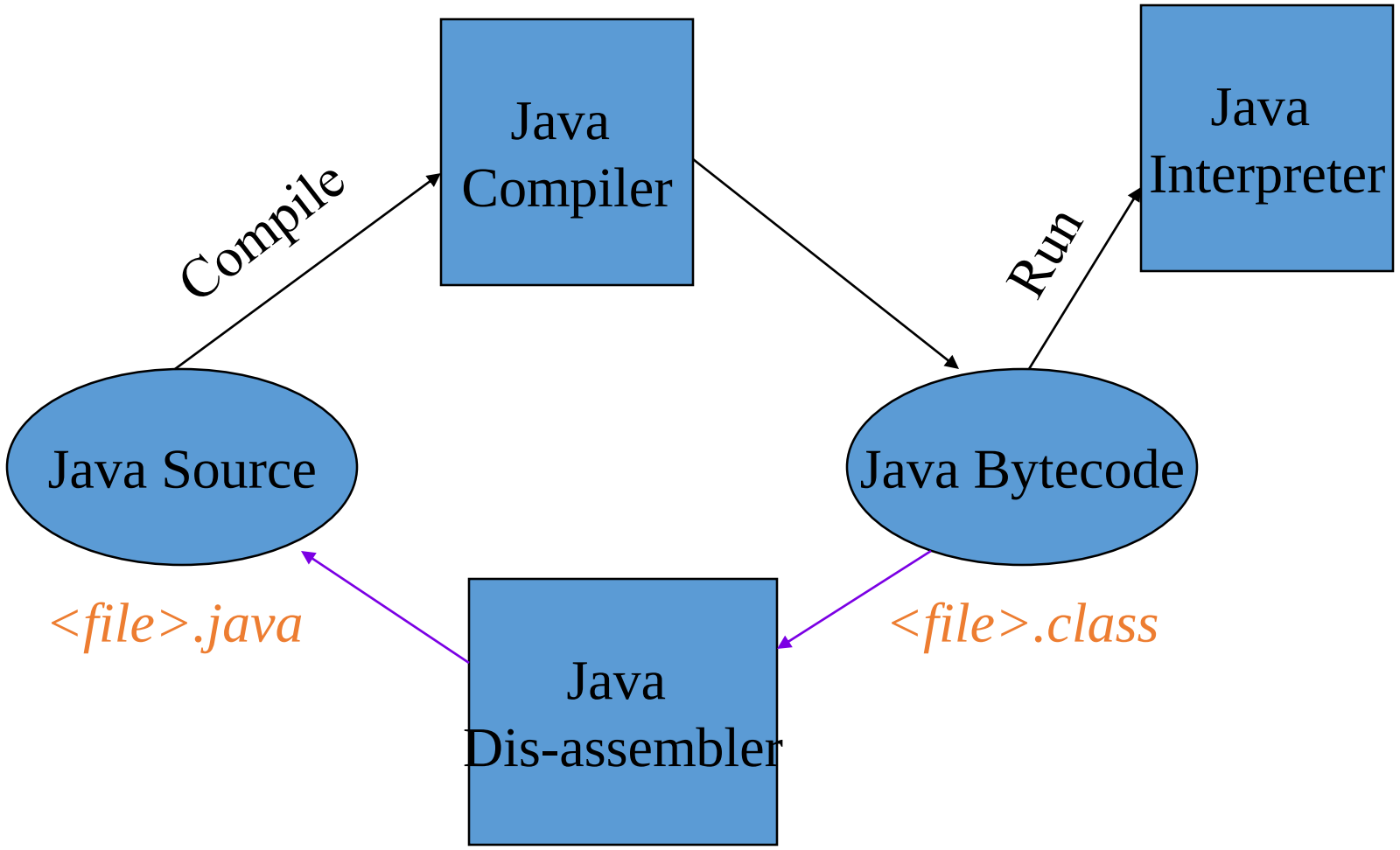
- Simple
- Architecture-neutral
- Object-Oriented
- Distributed
- Compiled
- Interpreted
- Statically Typed
- Multi-Threaded
- Garbage Collected

- Portable
- High-Performance
- Robust
- Secure
- Extensible
- Well-Understood

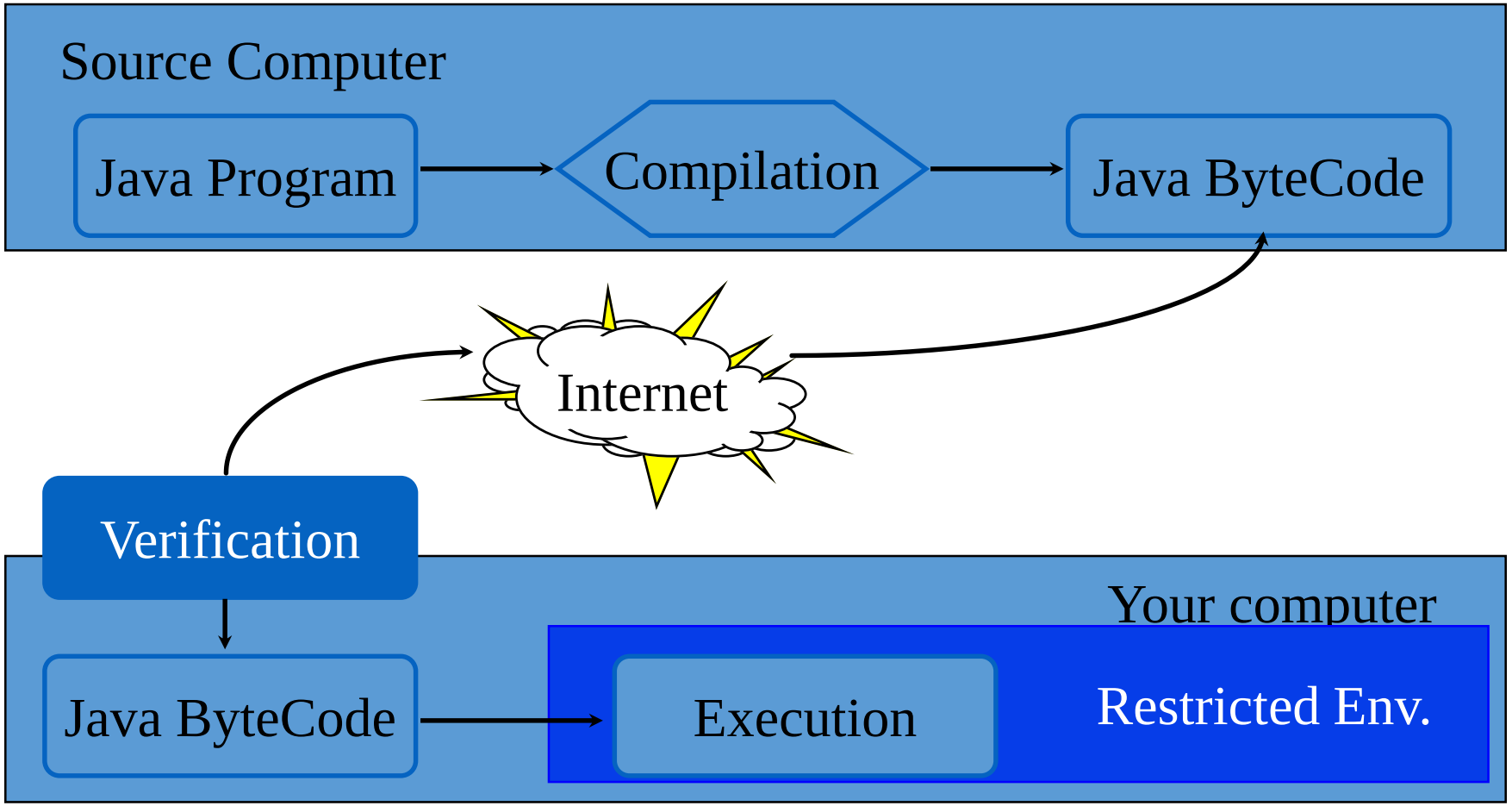
Java Developer's Kit (I)

- Java's programming environment
- Core Java API
- compiler
- interpreter
- debugger
- dis-assembler
- profiler
- more...

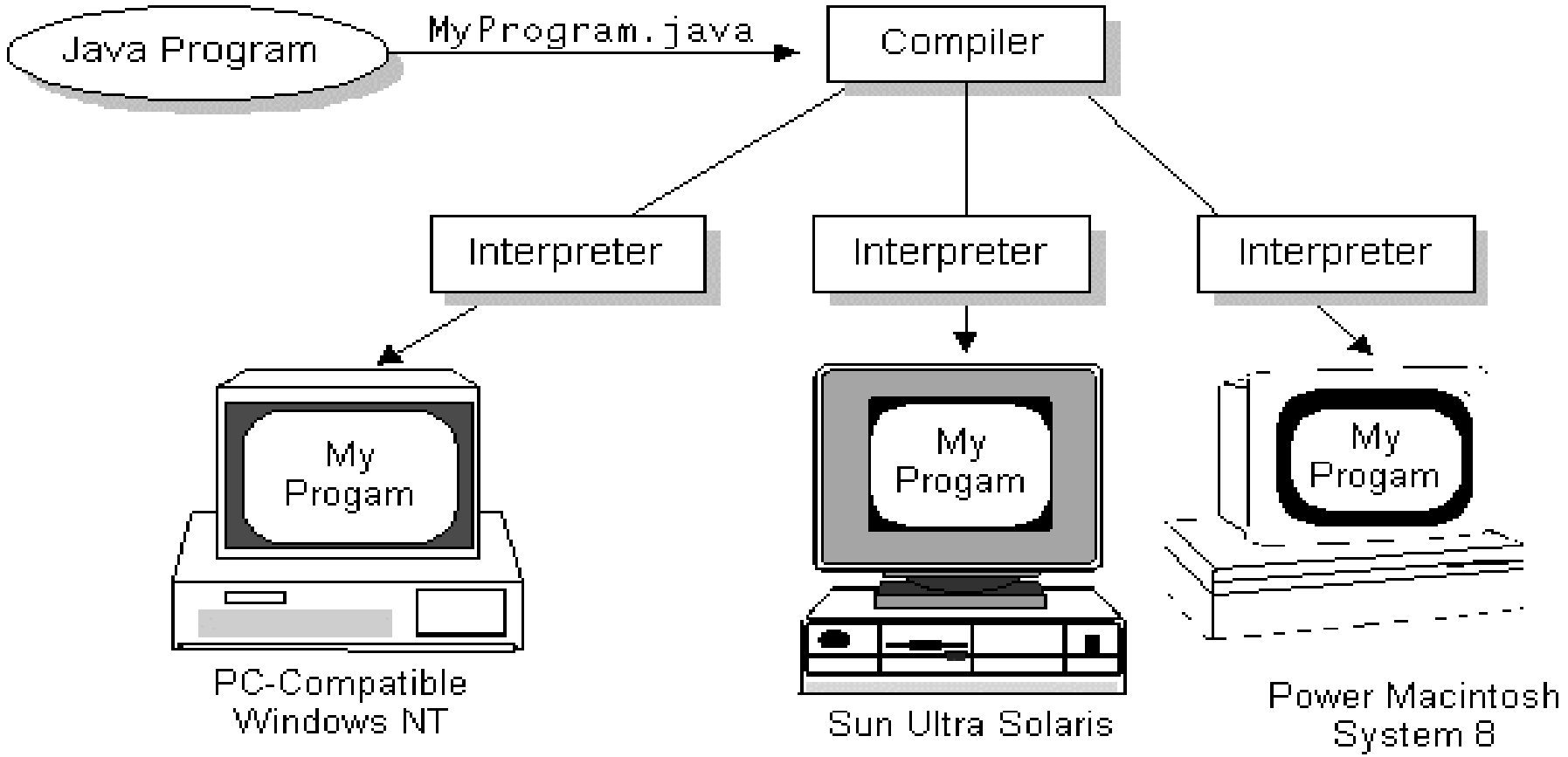
Java Developer's Kit (I)



Prepare and Execute Java



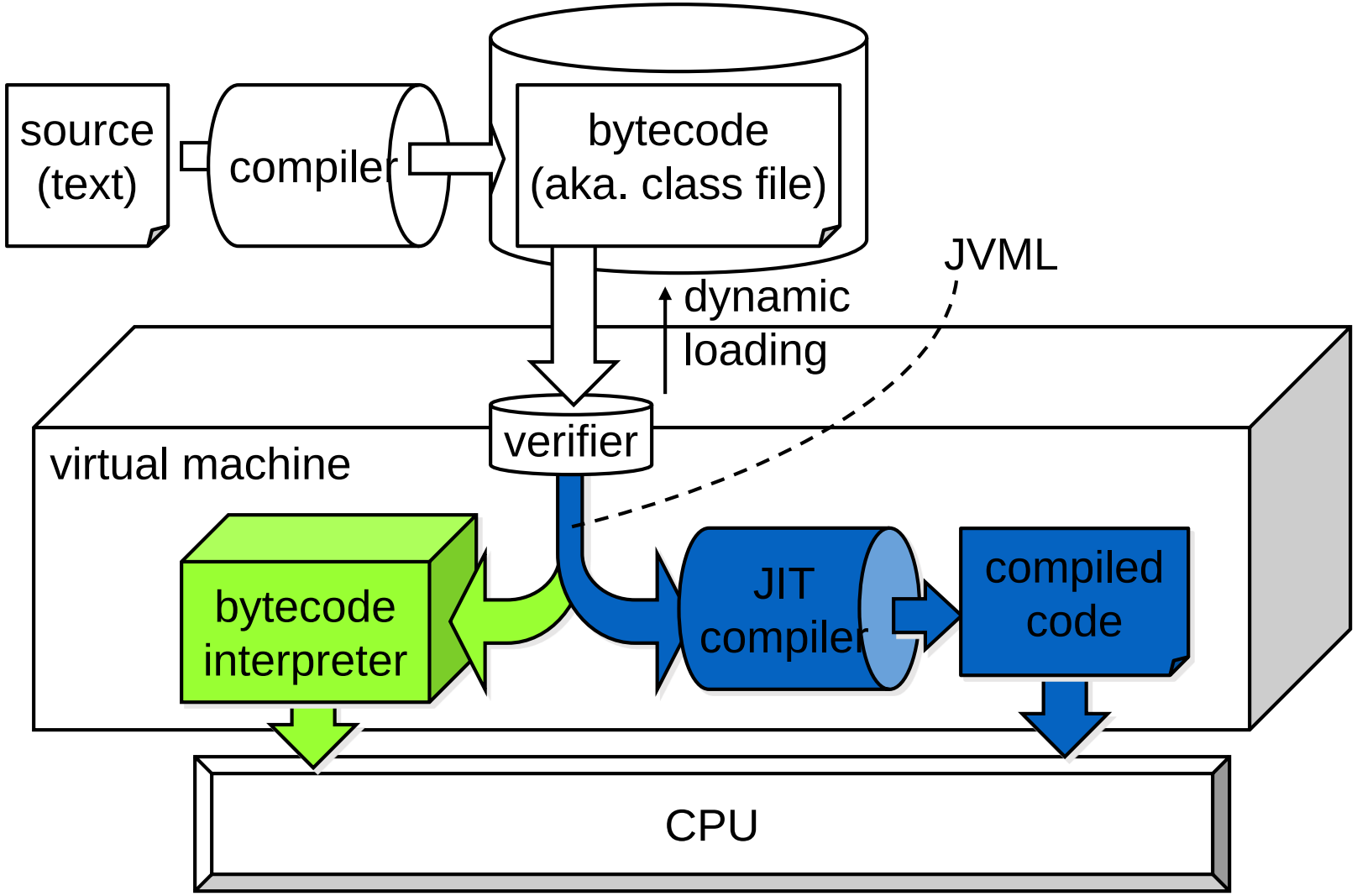
Write Once, Run Anywhere



ByteCode: Food for the VM

- For most languages, compilation produces machine code
- Java compilation produces “bytecode”
  - Intermediate code readable by the VM
  - Transferable across the Internet as *applets*
- VM interprets ByteCode into instructions
  - Partly responsible for performance lag
- ByteCode produced on any platform may be executed on any other platform which supports a VM

execution model of Java



The JIT

- *Just-In-Time* compiler
- Translates bytecode into machine code at runtime
  - 1-time overhead when run initiated
  - Performance increase 10-30 times
- Now the default for most JVM's
  - Can be turned off if desired
  - JIT can apply statistical optimizations based on runtime usage profile

Not just one JVM, but a whole family

- JVM (J2EE & J2SE)
  - Well-known Java Virtual Machine.
- CVM, KVM (J2ME)
  - Small devices.
  - Reduces some VM features to fit resource-constrained devices.
- JCVM (Java Card)
  - Smart cards.
  - It has least VM features.
- And there are also lots of other JVMs

The “Hello World” Application

Create a Java Source File

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

Compile and Run

- Compile
  - javac HelloWorld.java
- One file named HelloWorld.class is created if the compilation is succeeds.
- Run
  - java HelloWorld

The Simplest Java Application: Hello,World!

- Since Java is object-oriented, programs are organized into modules called classes, which may have data in variables and subroutines called methods.

Each program is enclosed in a class definition.

main() is the first method that is run.

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

The notation class.method or package.class.method is how to refer to a public method (with some exceptions).

Syntax is similar to C - braces for blocks, semicolon after each statement. Upper and lower case matter!

References

1. https://docs.oracle.com/javase/8/docs/

2. Schildt, H. (2014). *Java: the complete reference*. McGraw-Hill Education Group.

THANK YOU