



Java Programming

1-1

Introduction to JVM Architecture



Objectives

This lesson covers the following topics:

- What Is the Java Technology?
- Primary Goals of the Java Technology
- The Java Virtual Machine Architecture
- JVM Runtime Area

What Is the Java Technology?

- The Java Technology is:
 - A programming language
 - An application environment
 - A development environment
 - A deployment environment



What Is the Java Technology?

- The Java Programming Language is a high-level language. Its Syntax is similar to C and C++, but it removes many of the complex , confusing features of C and C++.
- The Java Programming Language includes the feature of automatic storage management by using a garbage collector.
- The Java programming language source code is compiled into the bytecode instruction set which can be run inside the Java Virtual Machine (JVM) process.

Primary Goals of the Java Platform

- Provide an Object Oriented programming language
- Provide an interpreted and just-in-time run-time environment
- Load classes dynamically
- Provide Multi-thread capability

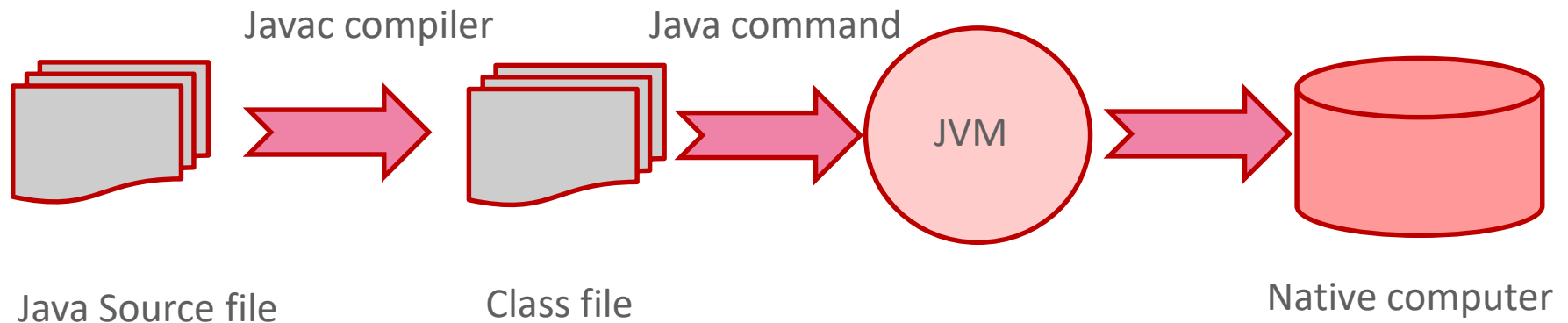


Java Application

- In the Java Language, all of the source code is written in plain text files with the .java extension name
- The java source code files are then compiled into .class extension files by the command javac
- A .class file contains bytecode which is a platform independent instruction set
- The Java command then runs the application

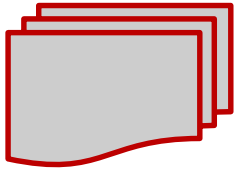
Java application

- Translation from source code to byte code procedure.



Multi Platform

- Write once:

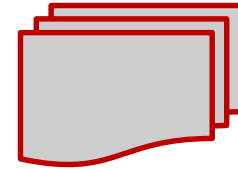


Java Source file

```
Class Test{  
    pubic int foo(int i){  
        i=i*i;  
        return I;  
    }  
}
```



Javac compiler

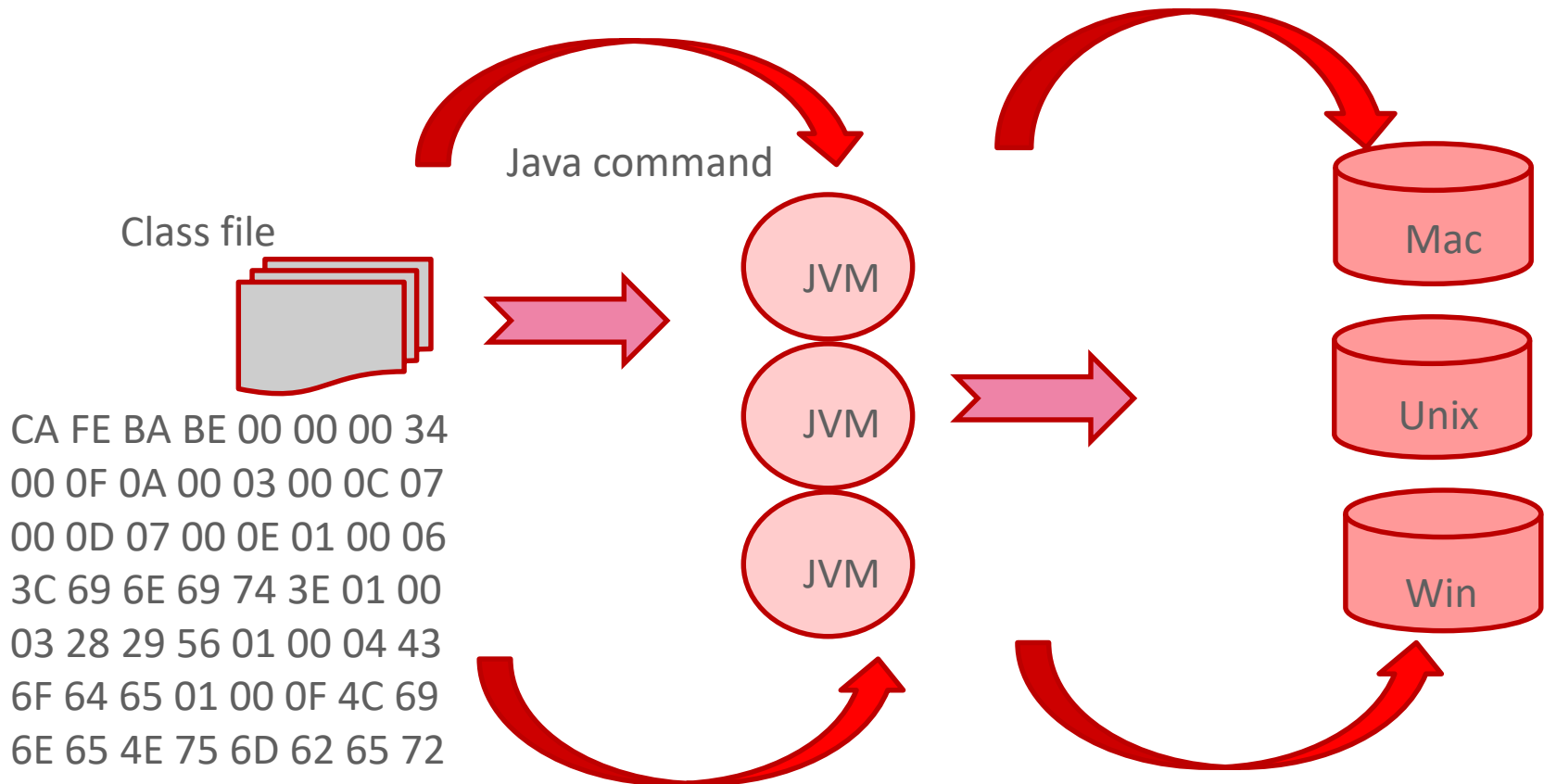


Class file

```
CA FE BA BE 00 00 00 34 00 0F 0A  
00 03 00 0C 07 00 0D 07 00 0E 01  
00 06 3C 69 6E 69 74 3E 01 00 03  
28 29 56 01 00 04 43 6F 64 65 01  
00 0F 4C 69 6E 65 4E 75 6D 62 65  
72 54 61 62 6C 65 01 00 03 66 6F  
6F 01 00 04 28 4900 07 00 00 00  
0A 00 02 00 00 00 03 00 04 00 04  
00 01 00 0A 00 00 00 02 00 0B
```

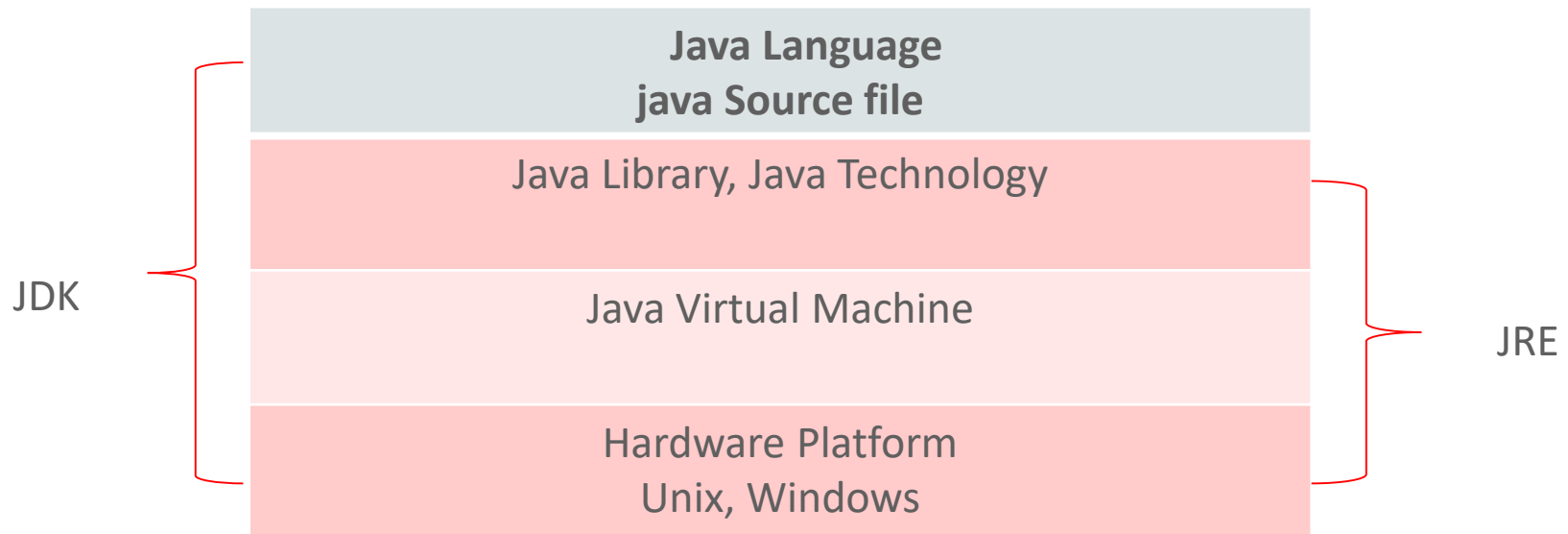
Multi Platform

- Run anywhere:



The Java SE Platform

Oracle has two products that implement Java Platform Standard Edition, Java SE development Kit and Java SE Runtime Environment



Java Application Example

Step 1: create a sample source code file Test.java

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

Java Application Example

Step 2. Compile the Source code to generate the Class file

```
javac HelloWorld.java
```

Step 3. Run the Application

```
Java HelloWorld
```

Step 4. Print out the result

```
Hello World!
```

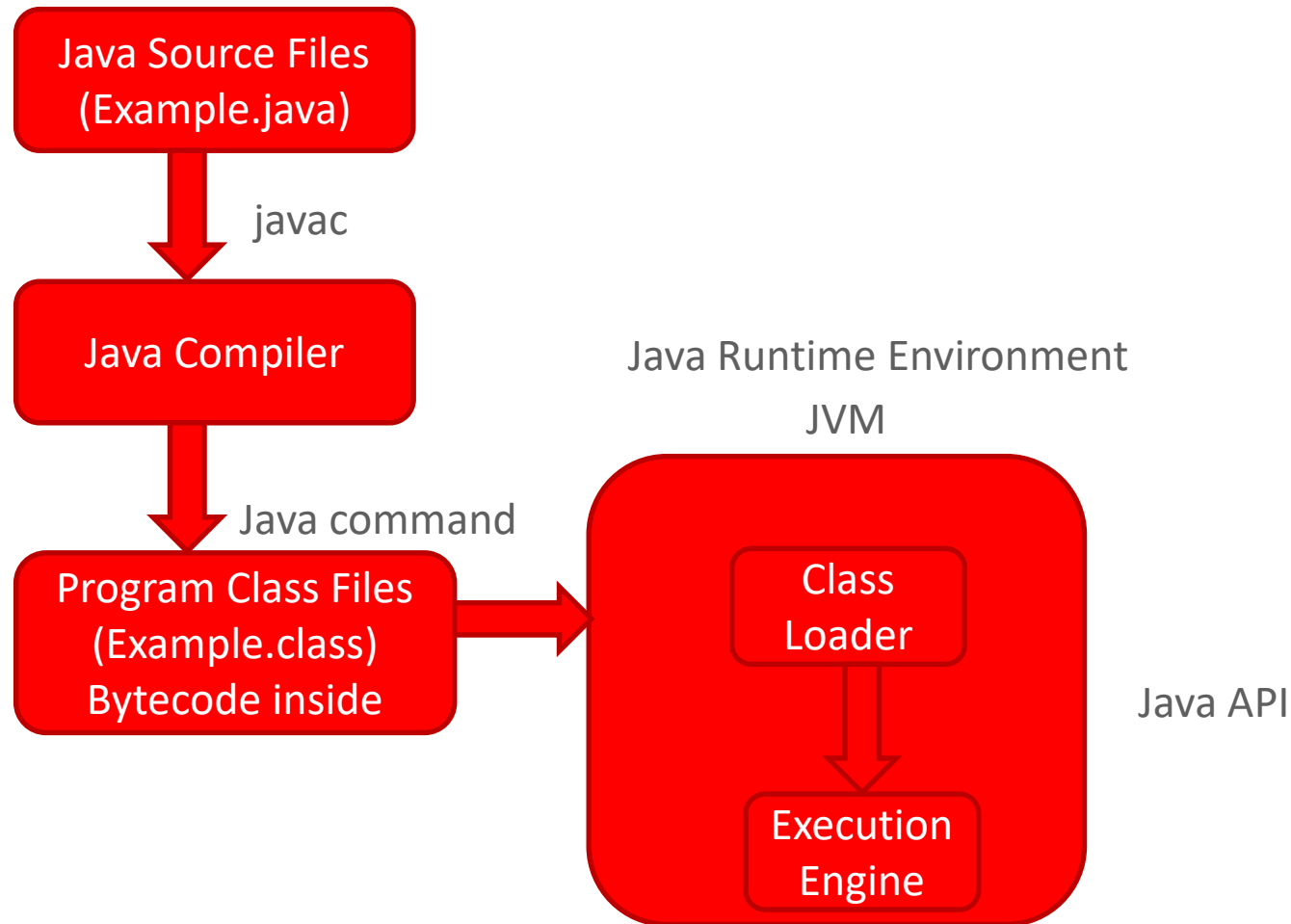
Java Application Example

- The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development.
- The Java Virtual Machine gives runtime support to the application and can make the application independent from different hardware systems.

The Java Virtual Machine

- We have already discussed how to write Java source code following the Java Language Specification and how to use the `javac` command to translate the Java source code into the Java class file.
- When you write your program, you access resources by calling functions in the Java API.
- In this lesson we are going to delve into the concept of the Java Virtual Machine (JVM).

The Java Virtual Machine



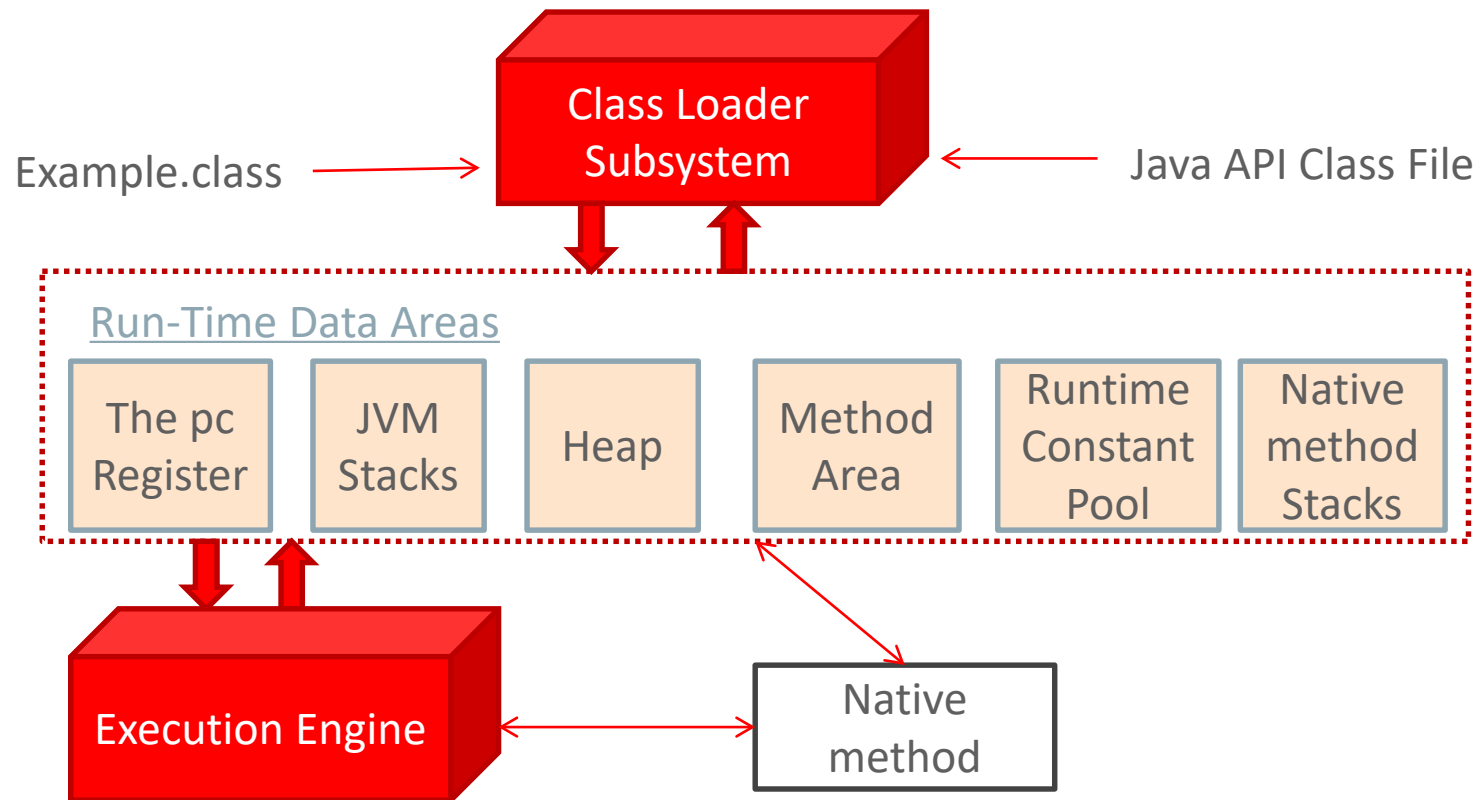
The Java Virtual Machine

- In the Java Virtual Machine Specification, the behavior of a virtual machine is described in terms of subsystem, data type and the Class File Format.
- These components describe an abstract architecture with no detailed internal implementation. The specification defines the required behavior of any Java Virtual Machine instance.
- In this Curriculum, we will use the Oracle hotspot Virtual machine.

The Java Virtual Machine

- The Java Virtual Machine knows nothing of the Java programming language, only of a particular binary format, the class file format. A class file contains Java Virtual Machine instructions (or byte codes) and a symbol table, as well as other ancillary information.
- The Java Virtual Machine is an abstract computer. Its specification defines features every JVM must have.
- The main job of a Java Virtual Machine is to load class files and execute the byte codes inside.

Java Virtual Machine Runtime Structure



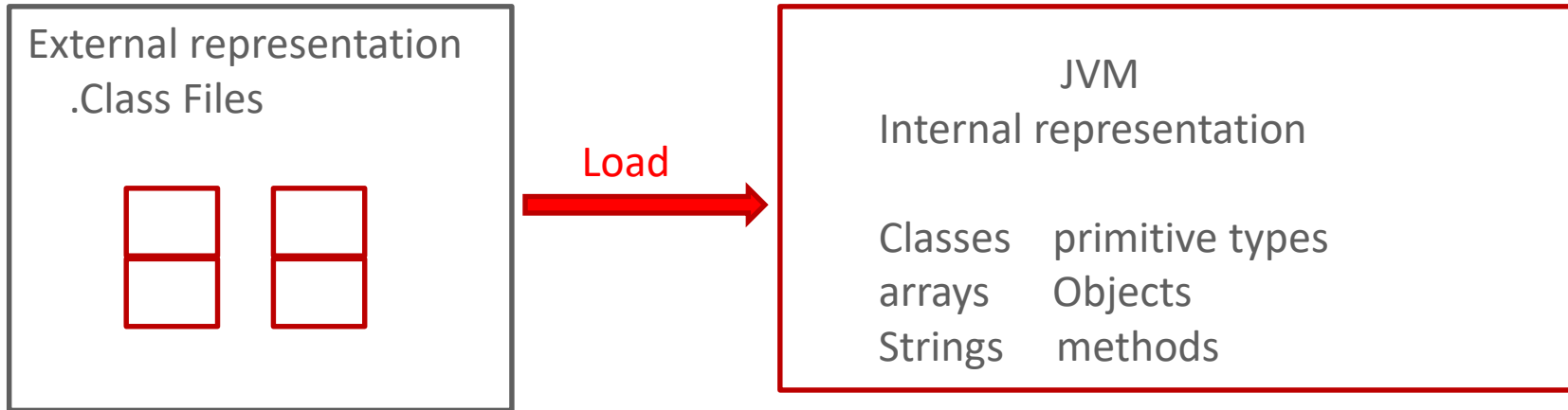
Java Virtual Machine Runtime Structure

- The JVM definition includes the major subsystems and memory areas described in the specification.
- Each Java Virtual Machine has a class loader subsystem: a mechanism for loading types (array, classes and interfaces) given fully qualified names.
- The Java Virtual Machine defines various run-time areas that are used during execution of a Java program.

Java Virtual Machine Runtime Structure

- Some of these data areas are created on the Java Virtual Machine startup procedure and are destroyed only when the Java Virtual Machine exits.
- Other data areas are thread specific.
- The JVM stack data area is created when a thread is created and destroyed when the thread exits.
- Each Java Virtual Machine also has an execution engine which is responsible for executing the instructions contained in the methods of the loaded classes.

Class Files and Class File Format



- The JVM specification defines a machine independent “class file format” that all JVM implementations must support.
- Java provides a dynamic load feature; it loads and links the class when it refers to a class for the first time during runtime, not at compile time.

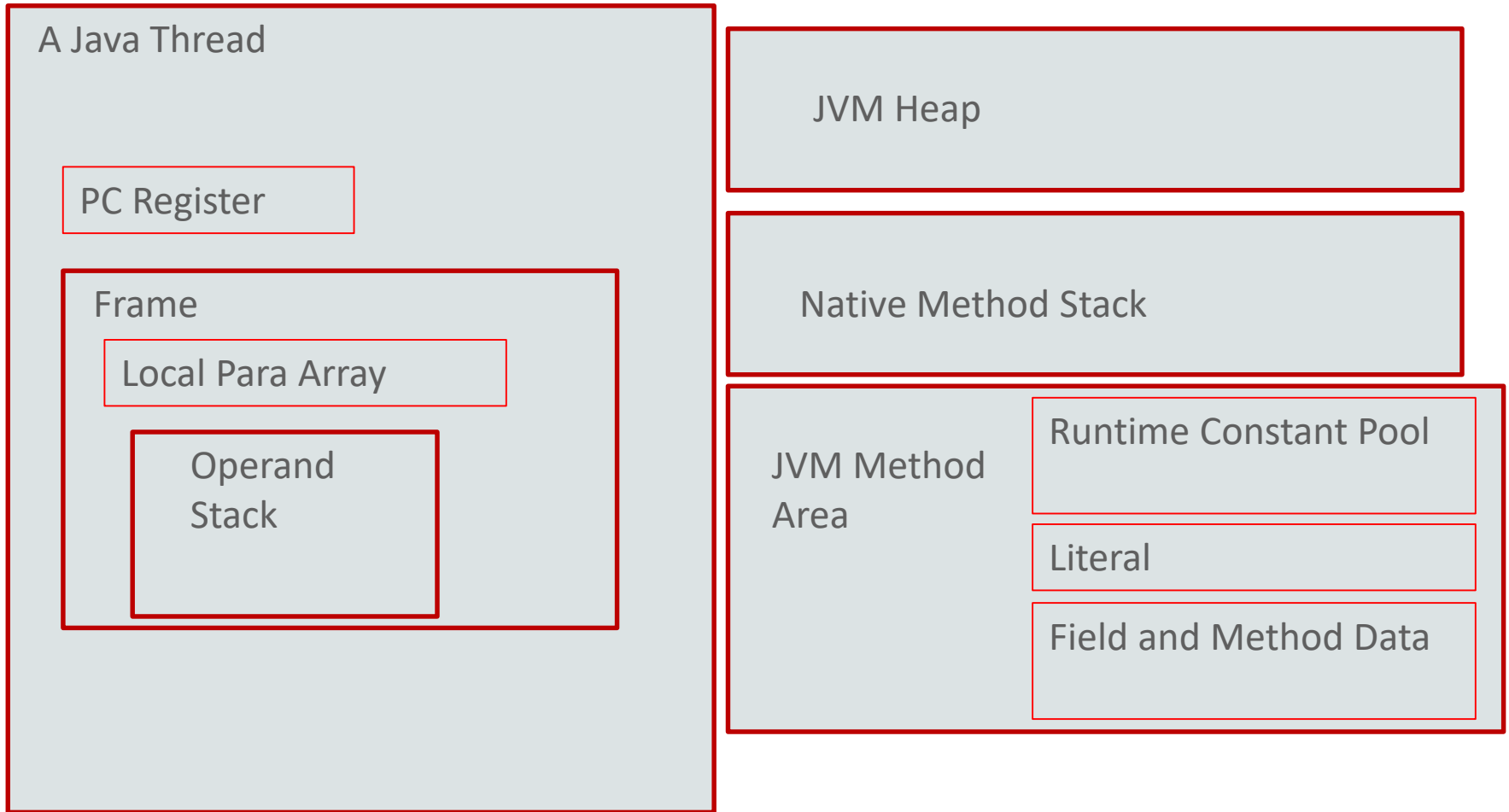
Data Types

- Like the Java programming language, the Java Virtual Machine operates on two kinds of types: primitive types and reference types.
- Primitive types:
 - boolean: boolean
 - numeric integral: byte, short, int, long, char
 - numeric floating point: float, double
 - internal, for exception handling: returnAddress

Data Types

- Reference types:
 - class types
 - array types
 - interface types
- The Java Virtual Machine expects that nearly all type checking is done prior to run time, typically by a compiler.

Runtime data areas



JVM Runtime data area

- When a Java Virtual Machine runs a program, it needs memory to store many things, including byte codes, the intermediate data generated from computations, objects from new operators, parameters to the methods, return values, local variables and other information it extracts from loaded class files(Strings, integer and metadata and other literals).
- The decisions about the structural details of the runtime data areas are left to the designers of the implementations.

JVM Runtime data area

- The Pc Register:
 - Each JVM thread has its own pc (Program counter) register.
- Native Method Stack:
 - JVM may use conventional stack to support native methods.
- Java Virtual Machine Stack:
 - Describes how to execute the method.

JVM Runtime data area

- JVM Stack will store a stack frame which holds local variables and partial results.
- Plays a part in method invocation and return.
- Each JVM thread has a private Java machine Stack.
- The Java Virtual machine can support many threads of execution at once. Each Java Virtual Machine thread has its own pc (Program counter) register.
- As each new thread comes into existence, it gets its own pc register and Java Stack.

JVM Runtime data area

- The pc register contains the address of the Java Virtual Machine instruction currently being executed.
- JVM implicitly takes arguments from the top of the stack, and places the result on the top of the stack.

JVM Method Area

- Method Area:
 - Stores run-time constant pool, field and method data, method bytecode and constructor.
 - The memory is shared by all Java Threads.
 - The Heap is created by JVM and it's size can be specified and tuned.
 - Stays in MetaSpace for HotSpot JVM .

JVM Method Area

- Inside a Java Virtual Machine instance, information about loaded types is stored in a logical area of memory called the method area.
- This area is shared among all Java Virtual Machine threads.
- When the JVM loads a type, it uses a class loader to locate the appropriate class file.
- The class loader reads and passes the class definition to the virtual machine. The virtual machine stores the information in the method area.

JVM Method Area

It typically contains:

- Type information : The fully qualified name of the type and direct superclass
- The constant pool
- Field information
- Method information
- All class or static variables
- A reference to class Classloader
- A reference to class Class

JVM Heap

The Heap

- Stores the class instances and arrays
- Is The most common place to store run-time data
- Can be managed by Garbage collector
- Is created by JVM and the size can be specified and tuned
- Is the memory shared by all Java Threads
- Is the run-time data area from which memory for all class instances and array is allocated

JVM Heap

- The heap is created on virtual machine start-up
 - Heap storage for objects is reclaimed by an automatic storage management system (known as a garbage collector).
 - The heap may be of a fixed size or may be expanded as required.
 - The heap is shared among all java Virtual Machine threads.
 - There is a maximum size that the heap can not exceed otherwise JVM throws a OutofMemoryError exception.

JVM Run-time data area Example

```
static Object sobj;  
  
public void method(){  
    Object obj=new Object();  
    sobj=new Object();  
}
```

Obj local variable locates in the JVM Stack

Instances of Object class stays in the Heap

Sobj static variable locates in the method area

JVM Run-time data area Example

- In this example, two instances are created
 - After the return of the method(), the obj variable will be removed from the Stack
 - Instance of object referenced by obj local variable needs to be collect by the Garbage Collector.
 - Instance of Object referenced by sobj static variable is still active in the program.
- The example demonstrates the data will occupy different areas of the Java Virtual Machine.

JVM Run-time data area Example

- The obj and sobj instance variables will refer to an object in the heap.
- The obj and sobj reference variable will be stored in the stack area.

Summary

In this lesson, you should have learned:

- What Is the Java Technology?
- Primary Goals of the Java Technology
- The Java Virtual Machine Architecture
- JVM Runtime Area

