ORACLE

Academy

Java Programming

7-1

Introduction to JVM Architecture

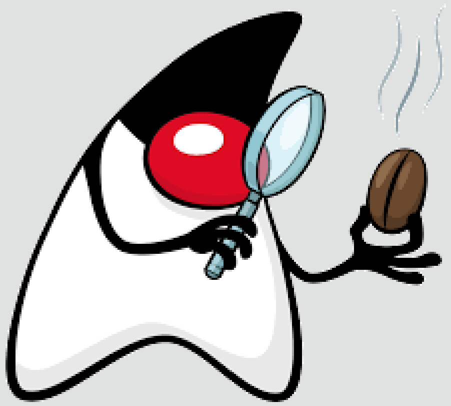
ORACLE
Academy

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?

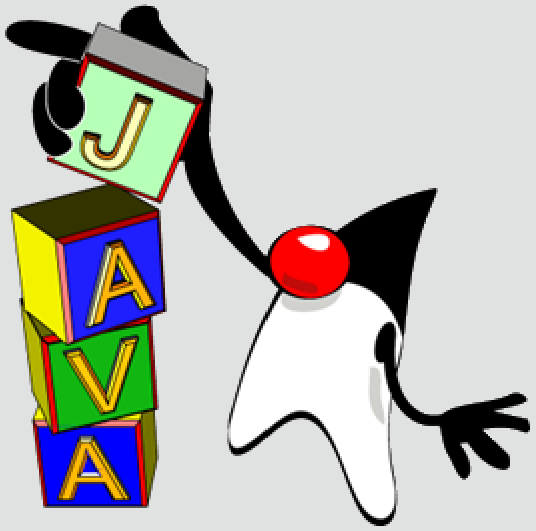
- The Java Programming Language is a high-level language
- Its syntax is similar to C and C++ but it removes many of the complex, confusing features of C and C++
- The Java Programming Language includes automatic storage (memory) management by using a garbage collector
- The Java Programming Language 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 runtime environment
- Load classes dynamically
- Provide multi-thread capability

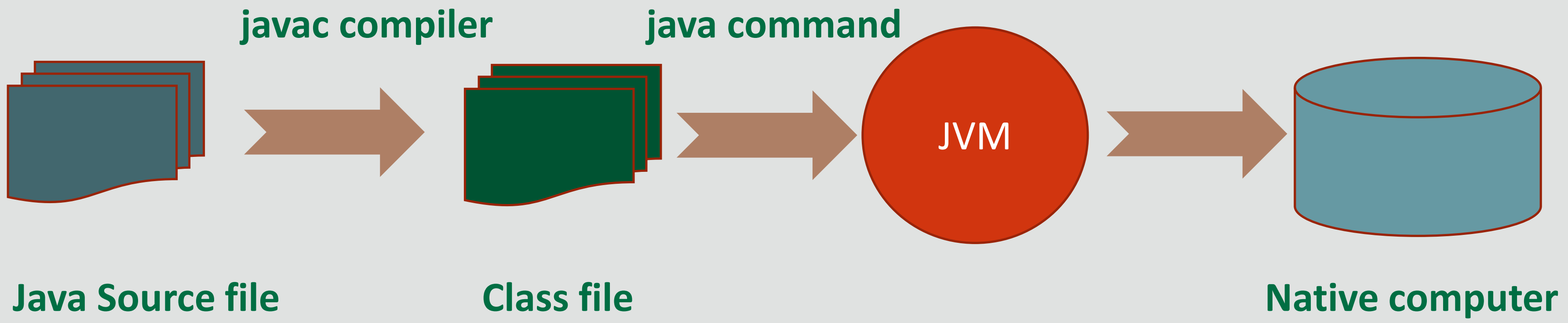
Java Application

- 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



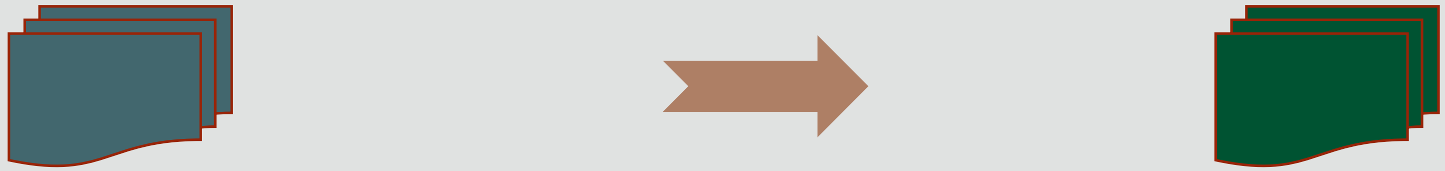
Java Application

- Translation from source code to byte code procedure.



Multi Platform

- Write once:

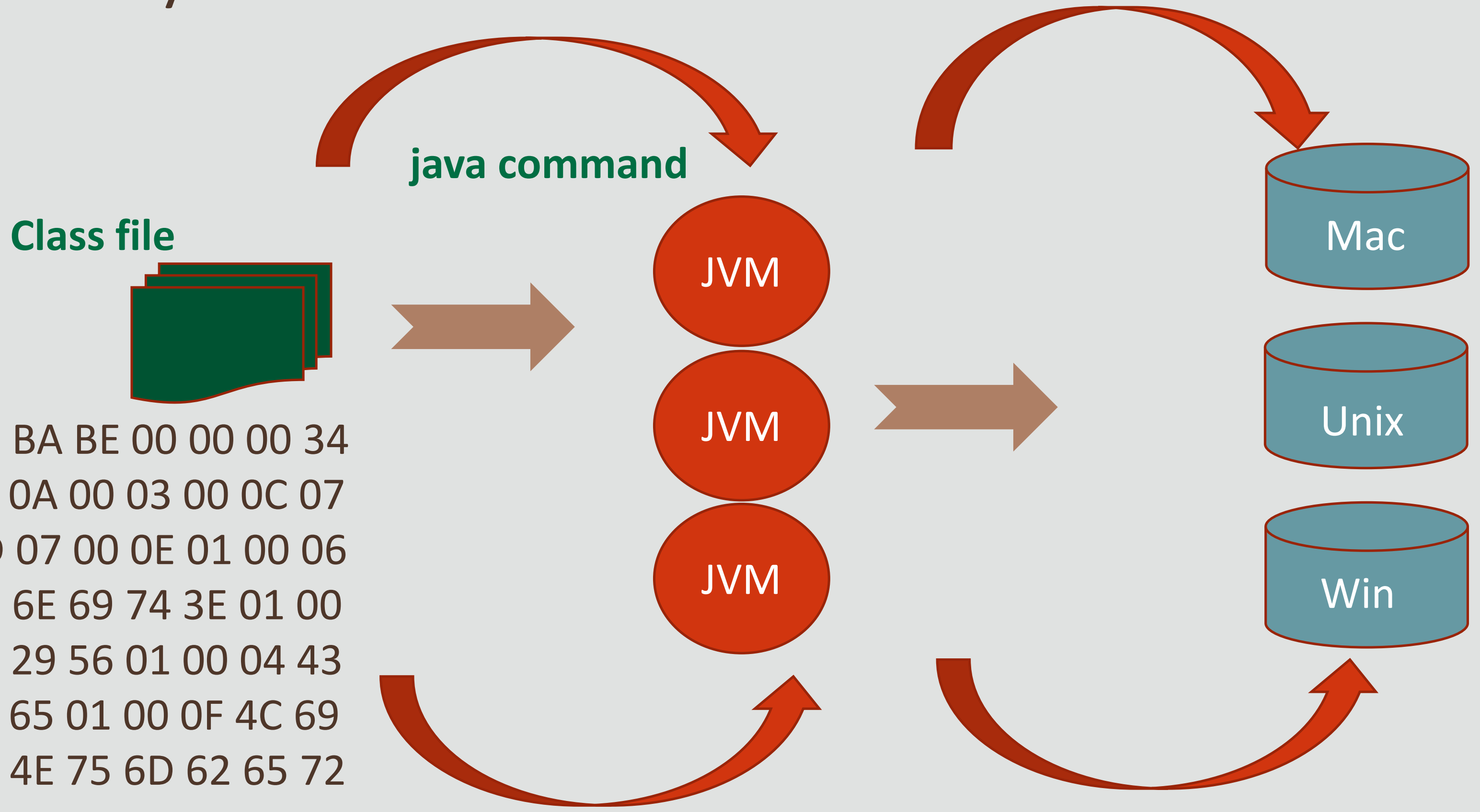


```
class Test{
   public int foo(int i){
      i=i*i;
      return i;
   }//end method foo
}//end class Test
```

```
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

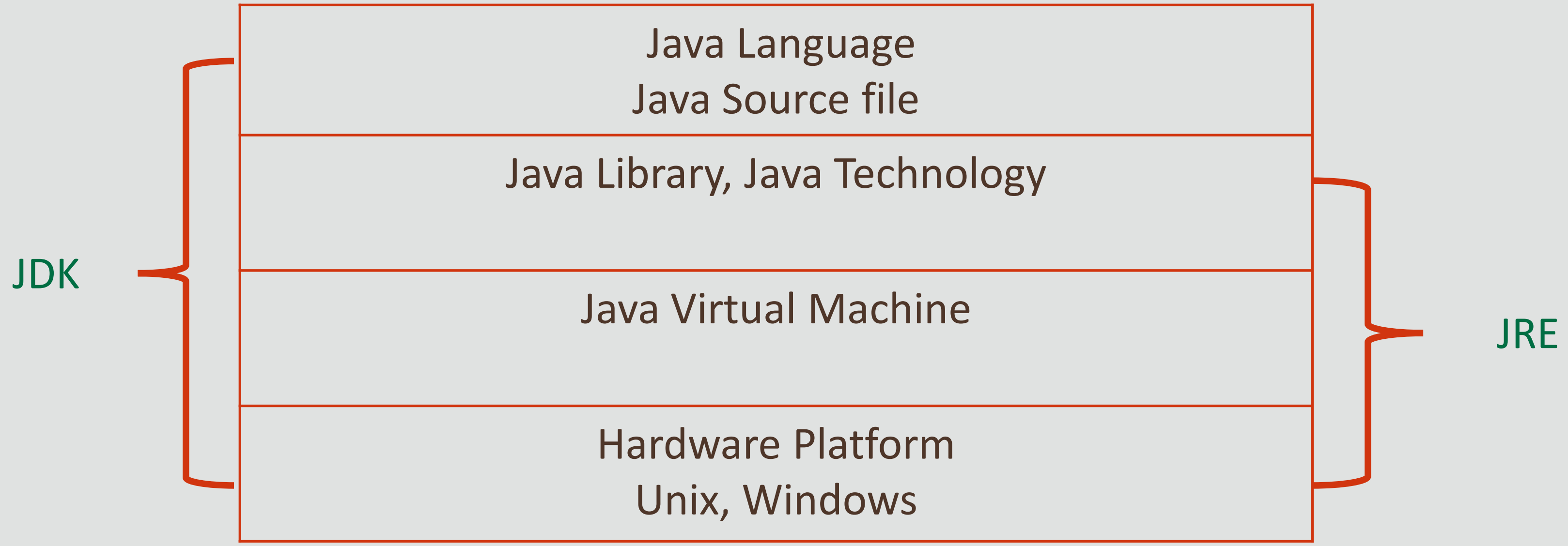
- Run anywhere:



```
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
```

The Java SE Platform

- Oracle has two products that implement the Java Platform Standard Edition, Java SE Development Kit (JDK) and Java SE Runtime Environment (JRE)

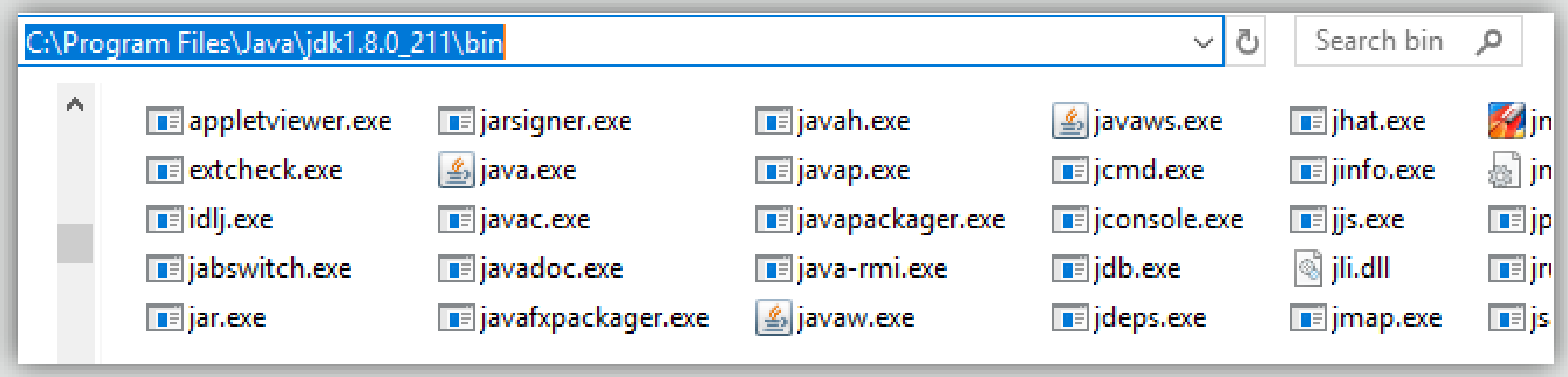


Let's code

In the console

Java Application Example

1. Check your system, as to be able to program in the console you need to make sure that Java has been added to your system properties
2. The first stage in this is to find the path to the javac.exe (compiler) file on your system
3. It's within the bin directory of the JDK, copy the path!



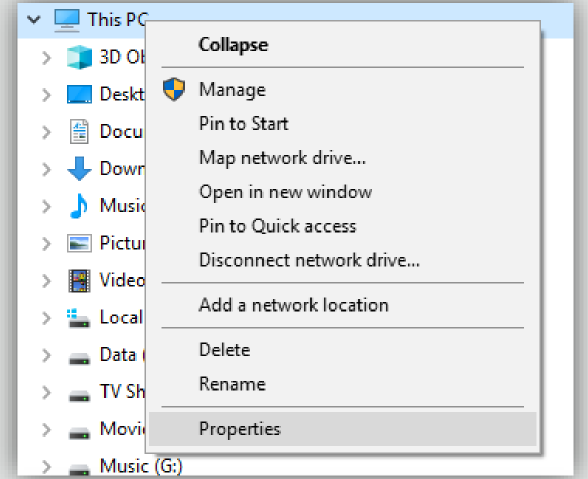
```
C:\Program Files\Java\jdk1.8.0_211\bin                Search bin

appletviewer.exe   jarsigner.exe       javah.exe          javaws.exe     jhat.exe    jn
extcheck.exe       java.exe            javap.exe          jcmd.exe       jinfo.exe   jn
idlj.exe           javac.exe           javapackager.exe   jconsole.exe   jjs.exe     jp
jabswitch.exe      javadoc.exe         java-rmi.exe       jdb.exe        jli.dll     jr
jar.exe            javafxpackager.exe  javaw.exe          jdeps.exe      jmap.exe    js
```

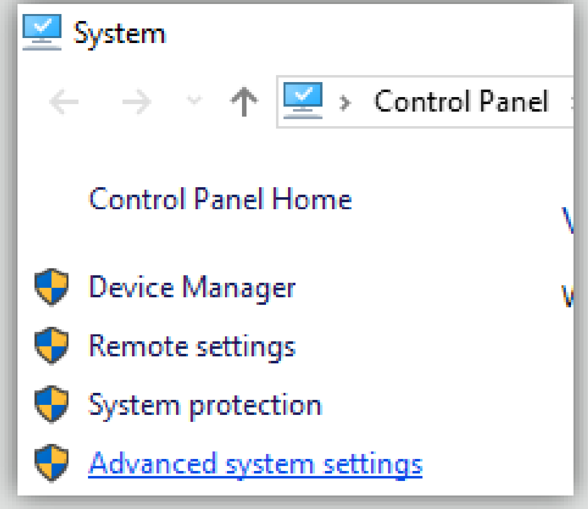
ORACLE Academy

Java Application Example

4. Right-click the PC icon and select properties to access the control panel home page on Windows



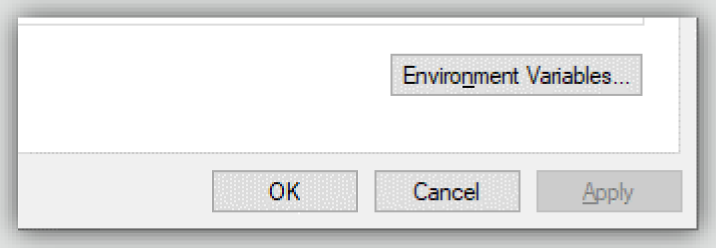
5. From the control panel select Advanced system settings from the menu

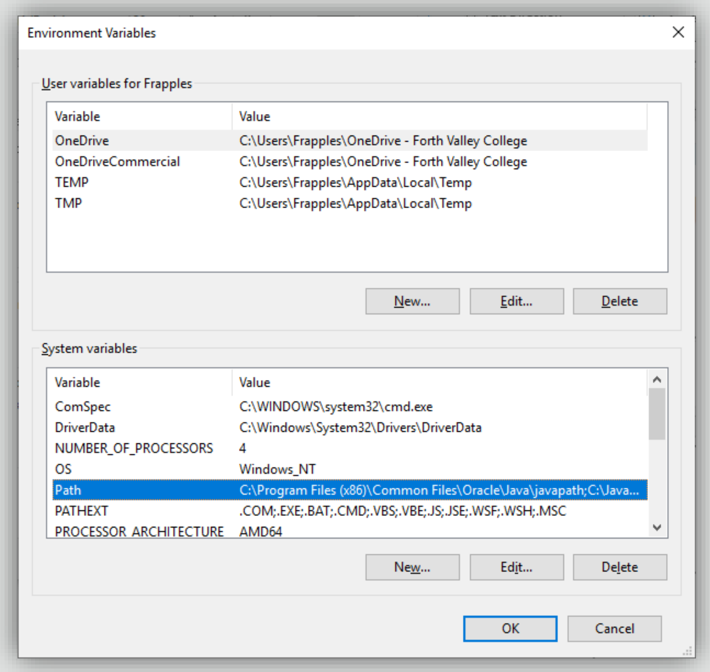


Java Application Example

6. Select the Environment Variables button from the System Properties dialog box

7. Select the Path option within System Variables and then select Edit




Environment Variables

User variables for Frapples

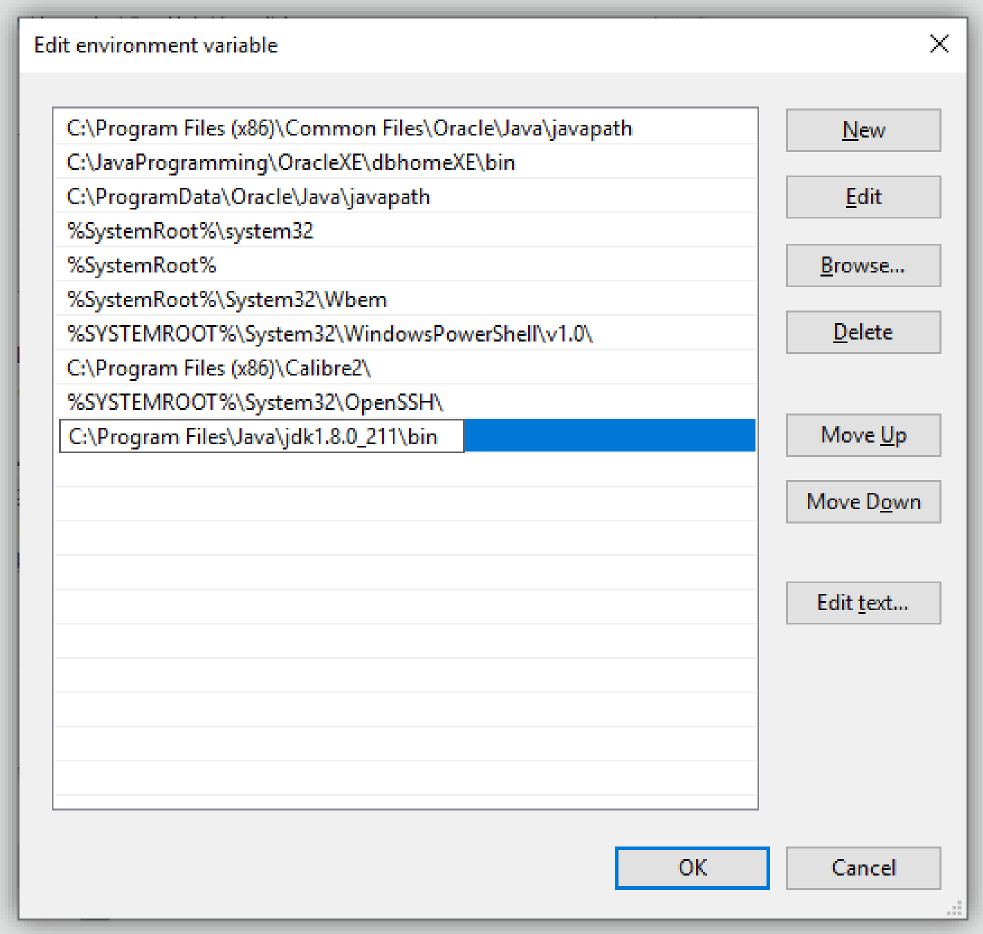
Variable	Value
OneDrive	C:\Users\Frapples\OneDrive - Forth Valley College
OneDriveCommercial	C:\Users\Frapples\OneDrive - Forth Valley College
TEMP	C:\Users\Frapples\AppData\Local\Temp
TMP	C:\Users\Frapples\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	4
OS	Windows_NT
Path	C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Java...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64

Java Application Example

8. Click the New button
9. Add the path to the bin directory of your JDK installation in the New field
10. Click OK to add the path to your system
11. Close all open system windows



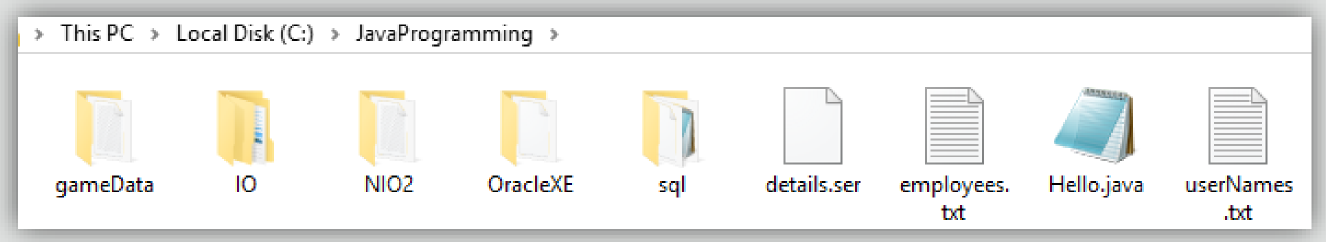
```
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
C:\JavaProgramming\OracleXE\dbhomeXE\bin
C:\ProgramData\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
C:\Program Files (x86)\Calibre2\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Java\jdk1.8.0_211\bin
```

Java Application Example

12. Open a text editor and enter the following Java code:

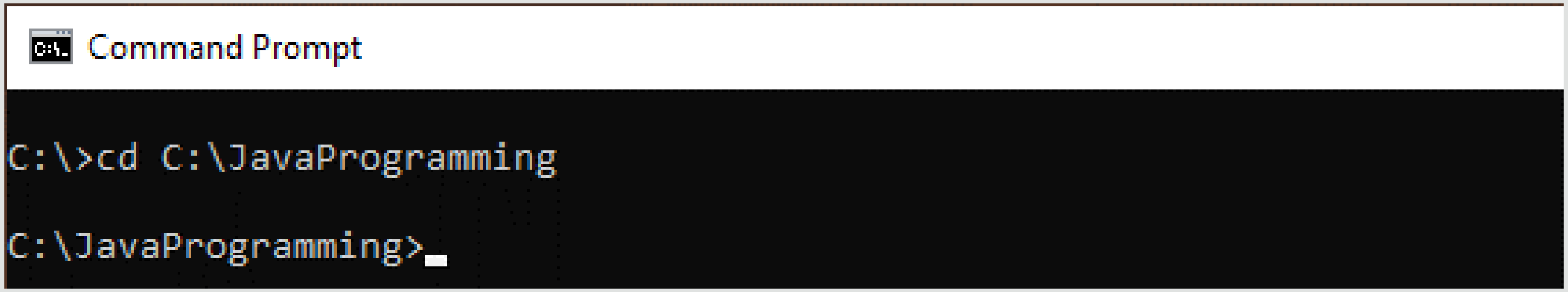
```
public class Hello{
   public static void main(String[] args){
      System.out.println("Hello welcome to console programming!");
   }//end method main
}//end class Hello
```

13. Save the file as **Hello.java** in the JavaProgramming directory on your C Drive



Java Application Example

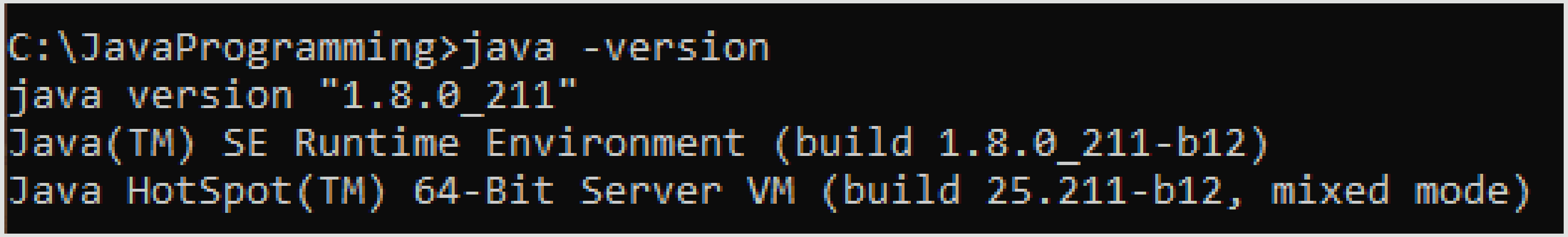
14. Open a command prompt and enter the following command to move to the JavaProgramming directory



```
C:\>cd C:\JavaProgramming

C:\JavaProgramming>_
```

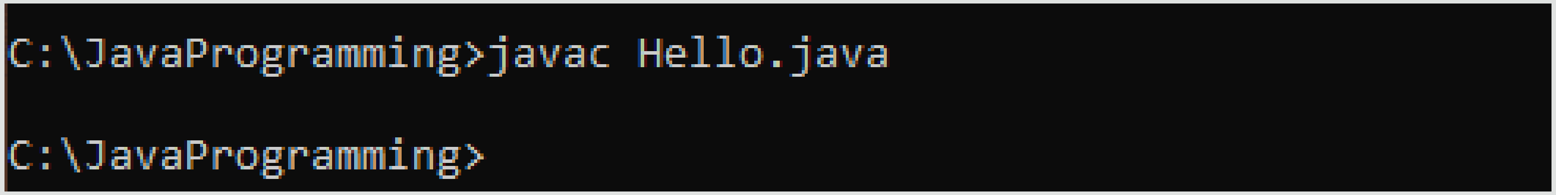
15. Type the following command to check the version of Java that is running on your system



```
C:\JavaProgramming>java -version
java version "1.8.0_211"
Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.211-b12, mixed mode)
```

Java Application Example

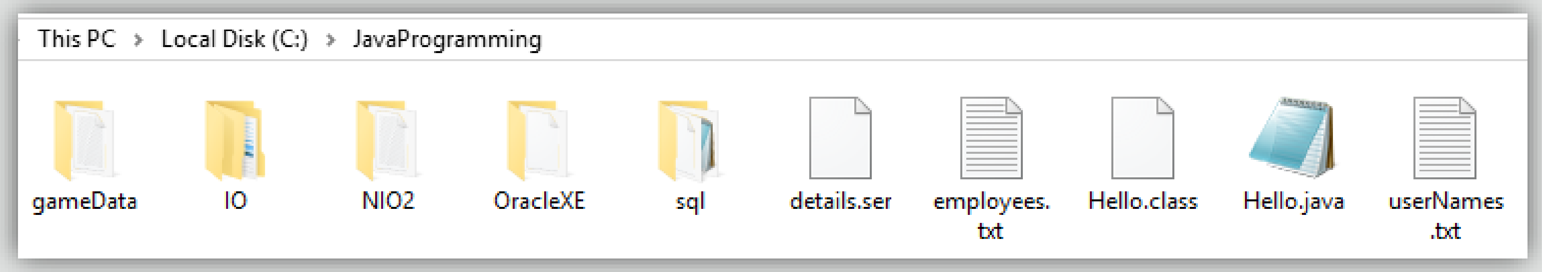
16. Compile the source code to generate the Class file



```
C:\JavaProgramming>javac Hello.java

C:\JavaProgramming>
```

17. There is no successful compilation message although you will get an error if it does not compile!
18. A Java class file (**Hello.class**) will now have been created beside the **Hello.java** file that was compiled

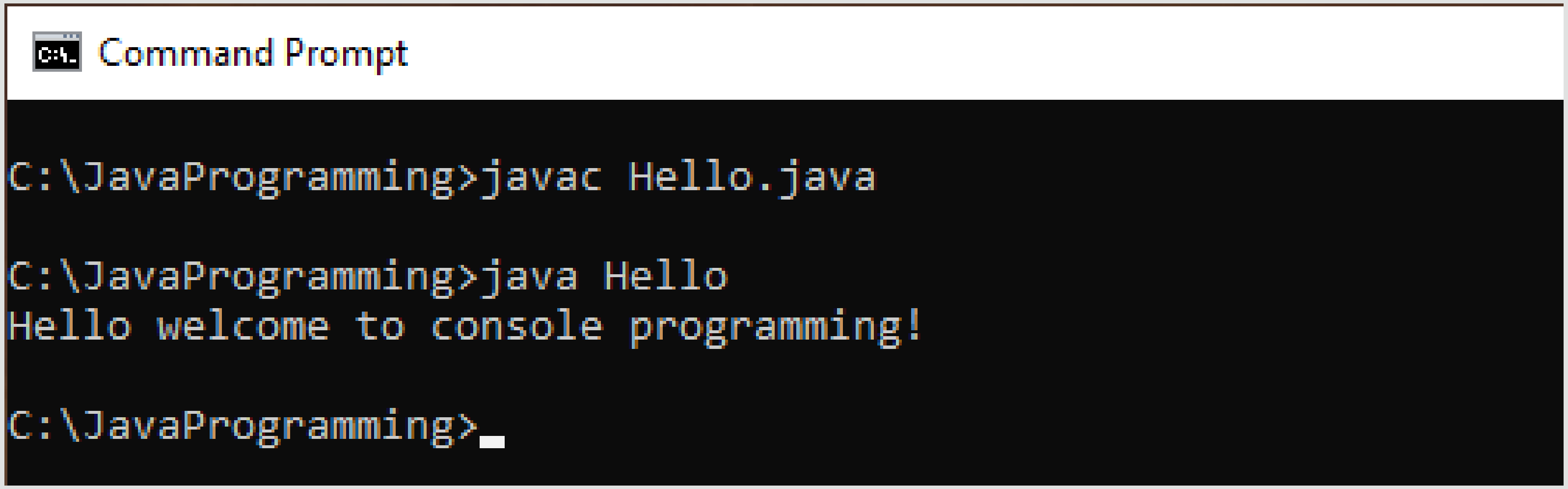


This PC › Local Disk (C:) › JavaProgramming

gameData IO NIO2 OracleXE sql details.ser employees.txt Hello.class Hello.java userNames.txt

Java Application Example

19. Run the Hello application by using the java command followed by the name of the class file (you do not need to add the file extension)



```
C:\JavaProgramming>javac Hello.java

C:\JavaProgramming>java Hello
Hello welcome to console programming!

C:\JavaProgramming>_
```

This is the same process that happens when you click the run button in your IDE!

ORACLE
Academy

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 (JVM) gives runtime support to the application and can make the application independent from different hardware systems

Java Application Example 2

- This code will show you how to pass values into your application at run-time using the (String[] args) parameter

1. Open a text editor and enter the following Java code:

```
public class FruitBasket {
   public static void main(String[] args) {
      displayFruits(args);
   }//end method main

   static void displayFruits(String[] fruits) {
      for(String fruit : fruits)
         System.out.println(fruit);
      //endfor
   }//end method displayFriuts
}//end class FruitBasket
```

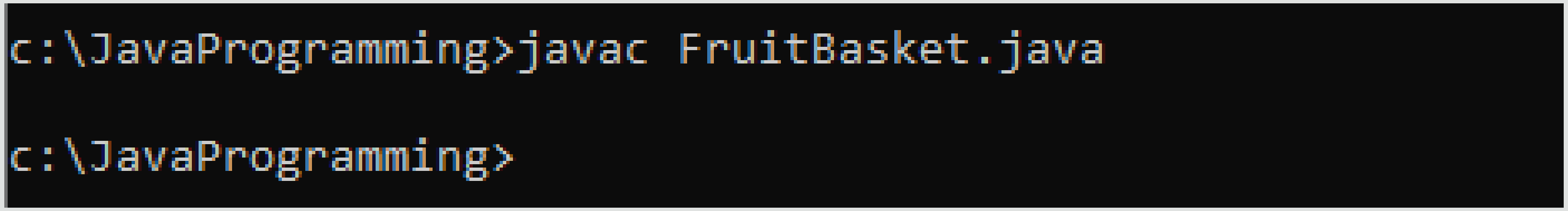
2. Save the file as **FruitBasket.java** in the JavaProgramming directory on your C Drive

Let's code

In the console

Java Application Example 2

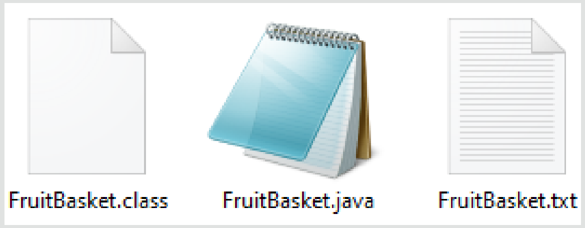
3. Compile the source code to generate the Class file



```
c:\JavaProgramming>javac FruitBasket.java

c:\JavaProgramming>
```

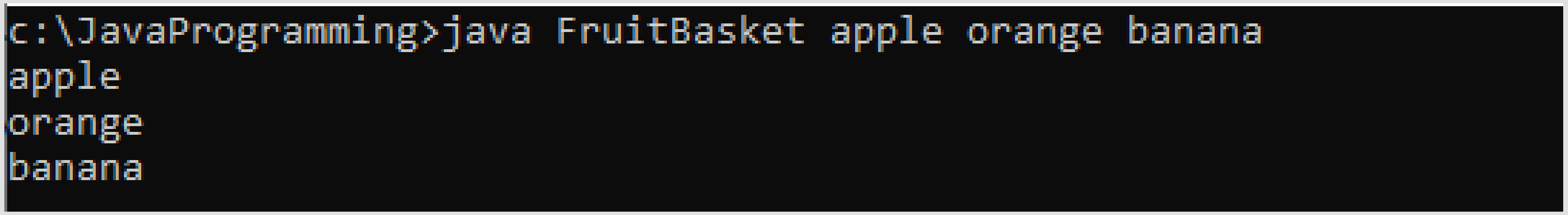
4. A Java class file (**FruitBasket.class**) will now have been created beside the **FruitBasket.java** file that was compiled



ORACLE
Academy

Java Application Example 2

5. Run the FruitBasket application by using the java command and the name of the class file followed by a list of text values separated by a space character



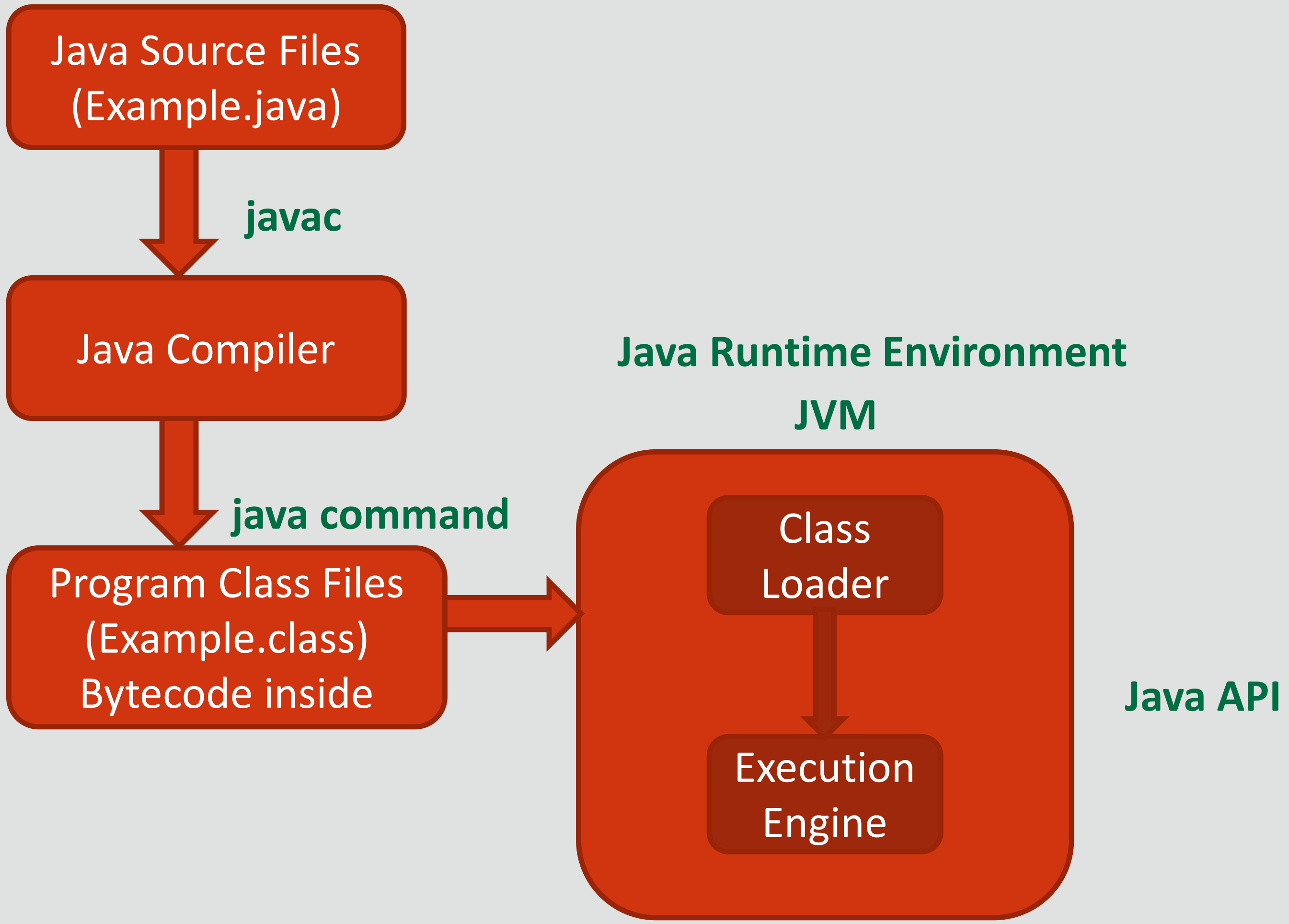
```
c:\JavaProgramming>java FruitBasket apple orange banana
apple
orange
banana
```

6. Test the application by providing different argument lists of fruit:
   – cherry apple orange avocado banana
   – clementine grapefruit
   – grape

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 methods 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



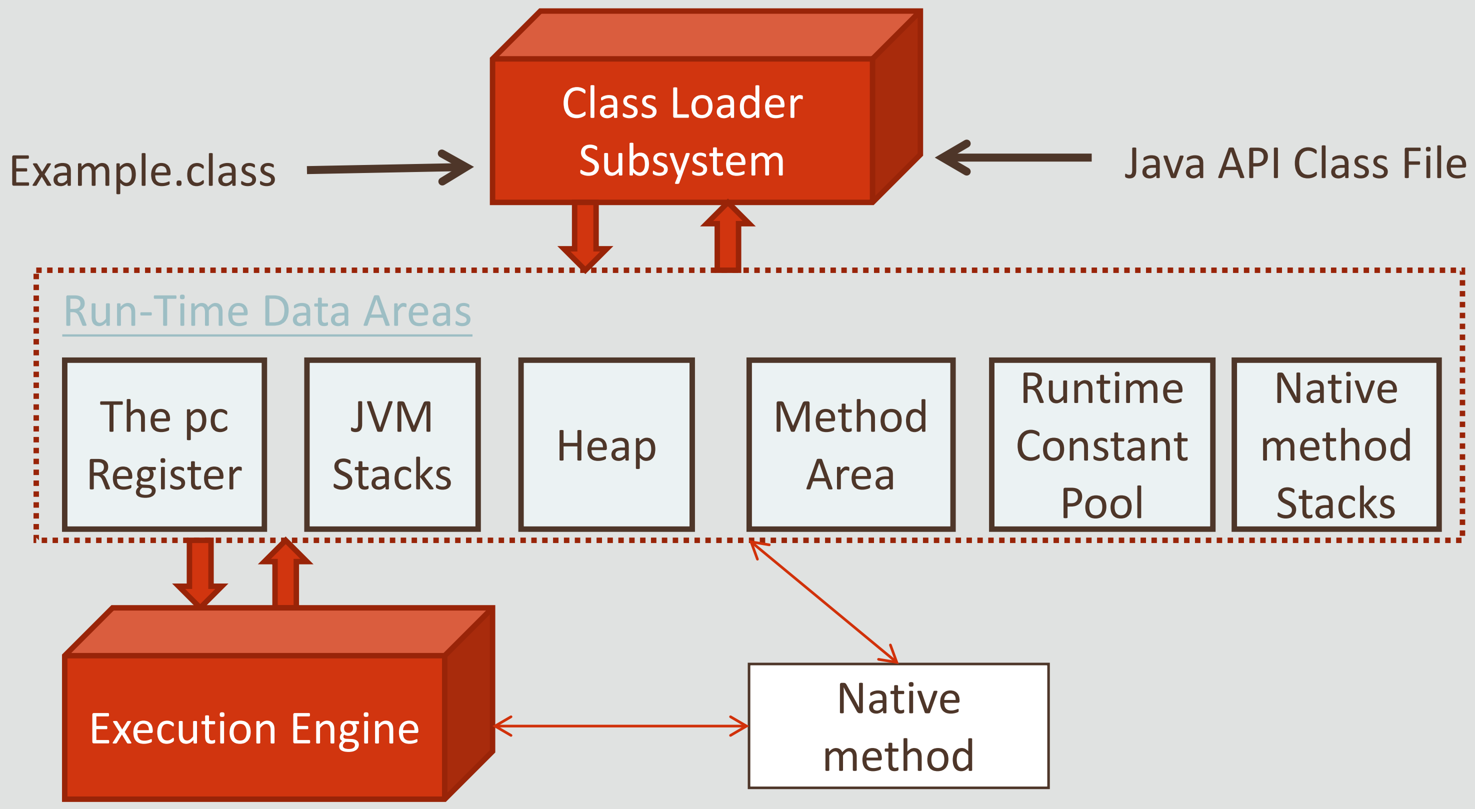
The Java Virtual Machine

- 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 course 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, i’ts 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



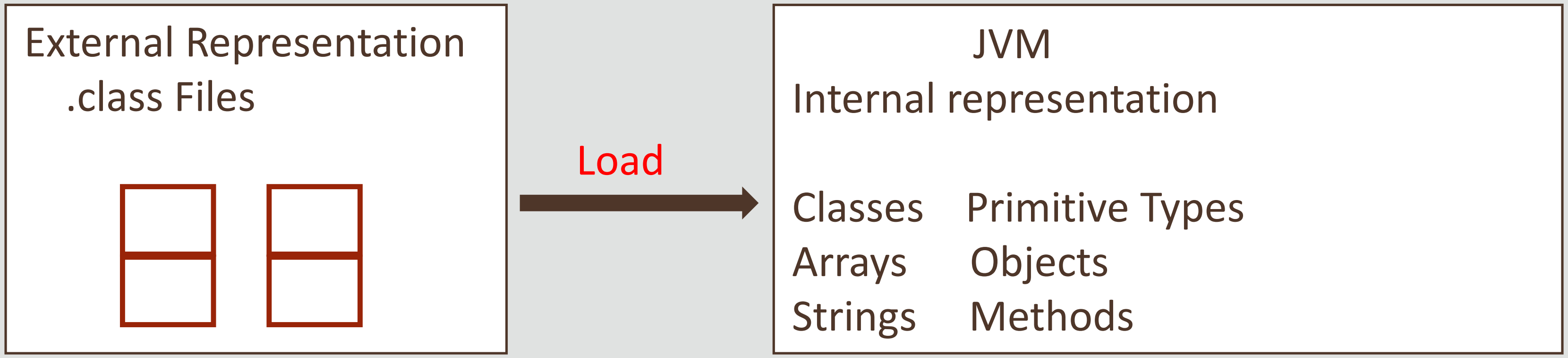
Java Virtual Machine Runtime Structure

- 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 by 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
  - reference types

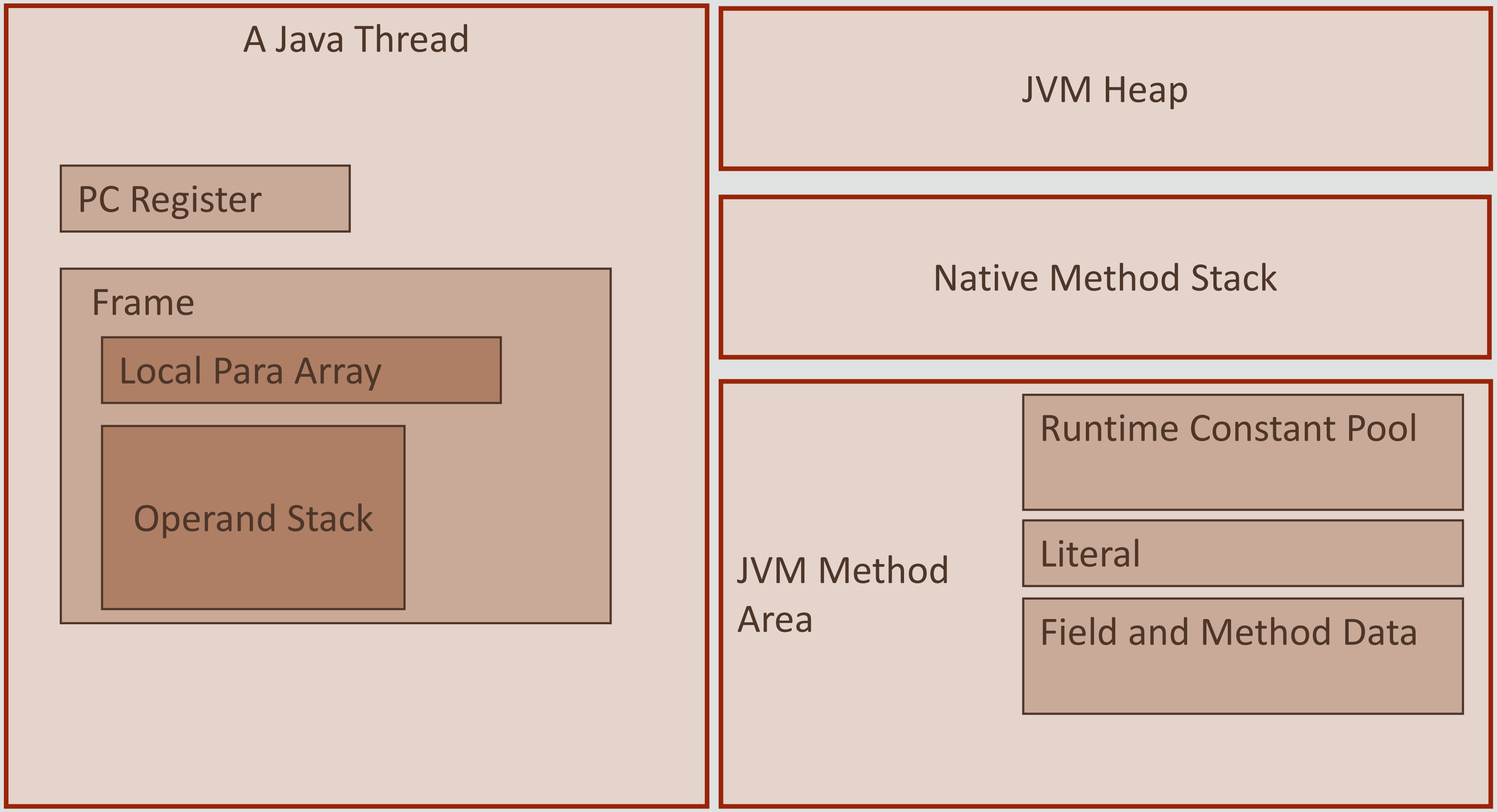
- Primitive types:
  - boolean: boolean
  - numeric integral: byte, short, int, long, char
  - numeric floating point: float, double
  - internal, for exception handling: returnAddress

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

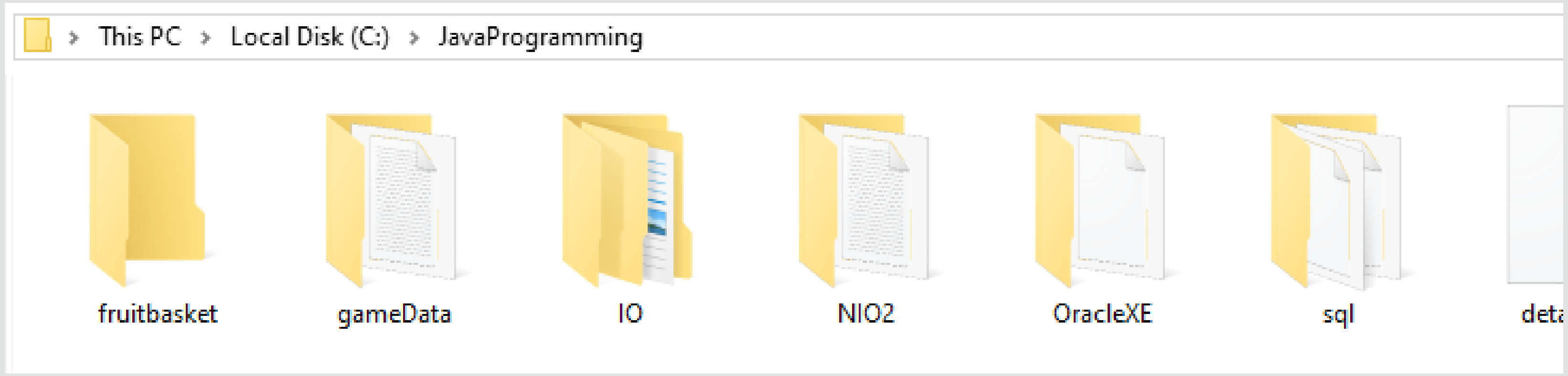


Java Application Example 3

- In this example you will create an application that will store the fruit information as an object
  - Each fruit class will be used to store the following values:
    - The name of the fruit
    - The colour of the fruit
    - The category of the fruit – pome, citrus, stone, tropical, berry, other
    - The number of each type of fruit in the bowl
  - The fruit call should have a method to return the values
  - A driver class will be used to populate and display the contents of the fruit bowl
  - A package will be used to manage the program files.

Java Application Example 3

1. In the JavaProgramming directory create a subdirectory named **fruitbasket**



This folder must be named exactly as you will name your package!
Remember: package names in java should be written in lowercase

Let's code

In the console

Java Application Example 3

2. Open a text editor and enter the following Java code:

```
package fruitbasket;

public class Fruit {
   String name;
   String colour;
   String category;
   int numberOfFruit;

   public Fruit(String name, String colour, String cat, String number) {
           this.name = name;
           this.colour = colour;
           this.category = cat;
           this.numberOfFruit = Integer.parseInt(number);
   }//end constructor
```

Code continued on the following slide….

Java Application Example 3

2. Continued Fruit code:

Continued code....

```
public String toString() {
           return "Fruit: " + name
                           + "\nColour: " + colour
                           + ", Category: " + category
                           + ", Number in bowl: " + numberOfFruit
                           + "\n";
   }//end method toString

}//end class Fruit
```

3. Save the file as **Fruit.java** in the fruitbasket subdirectory of the JavaProgramming directory on your C Drive

Java Application Example 3

4. Open a text editor and enter the following Java code:

```
package fruitbasket;

import java.util.ArrayList;

public class FruitBasket {

        public static void main(String[] args) {
                ArrayList<Fruit> fruits = new ArrayList<>();
                fillBowl(args, fruits);
                displayFruits(fruits);
        }//end method main



}//end class FruitBasket
```

Code continued on the following slide....

Java Application Example 3

4. Continued FruitBasket code:

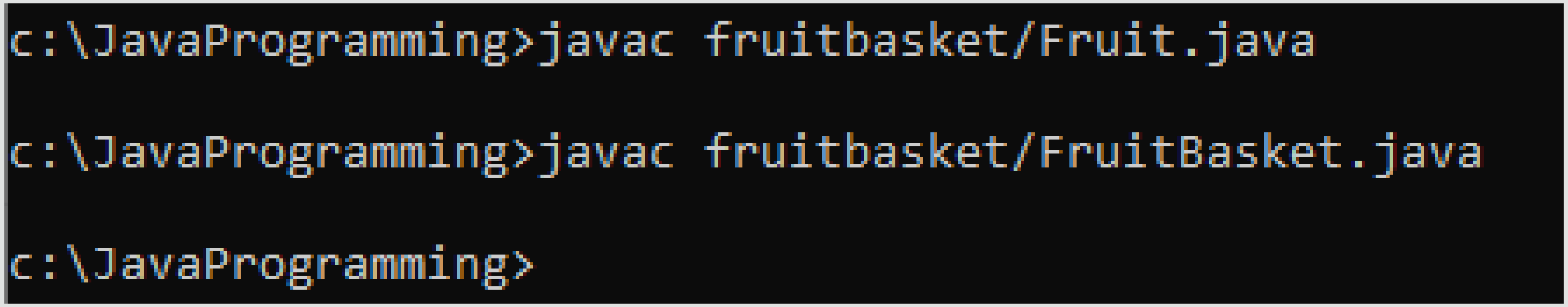
Continued code....

```
        static void fillBowl(String[] args, ArrayList<Fruit> fruits){
           int i=0;
           while(i+4 <= args.length) {
                   fruits.add(new Fruit(args[i], args[i+1], args[i+2],
args[i+3]));
              i+=4;
           }//end while
        }//end method fillBowl

        static void displayFruits(ArrayList<Fruit> fruits) {
                for(Fruit fruit : fruits)
                        System.out.println(fruit);
                //endfor
        }//end method displayFriuts
}//end class FruitBasket
```

Java Application Example 3

5. In the command line go to the JavaProgramming directory on the c drive
6. To compile the java code include the subdirectory in the path for the javac commands



```
c:\JavaProgramming>javac fruitbasket/Fruit.java

c:\JavaProgramming>javac fruitbasket/FruitBasket.java

c:\JavaProgramming>
```

7. Confirm that both class files have been created

Java Application Example 3

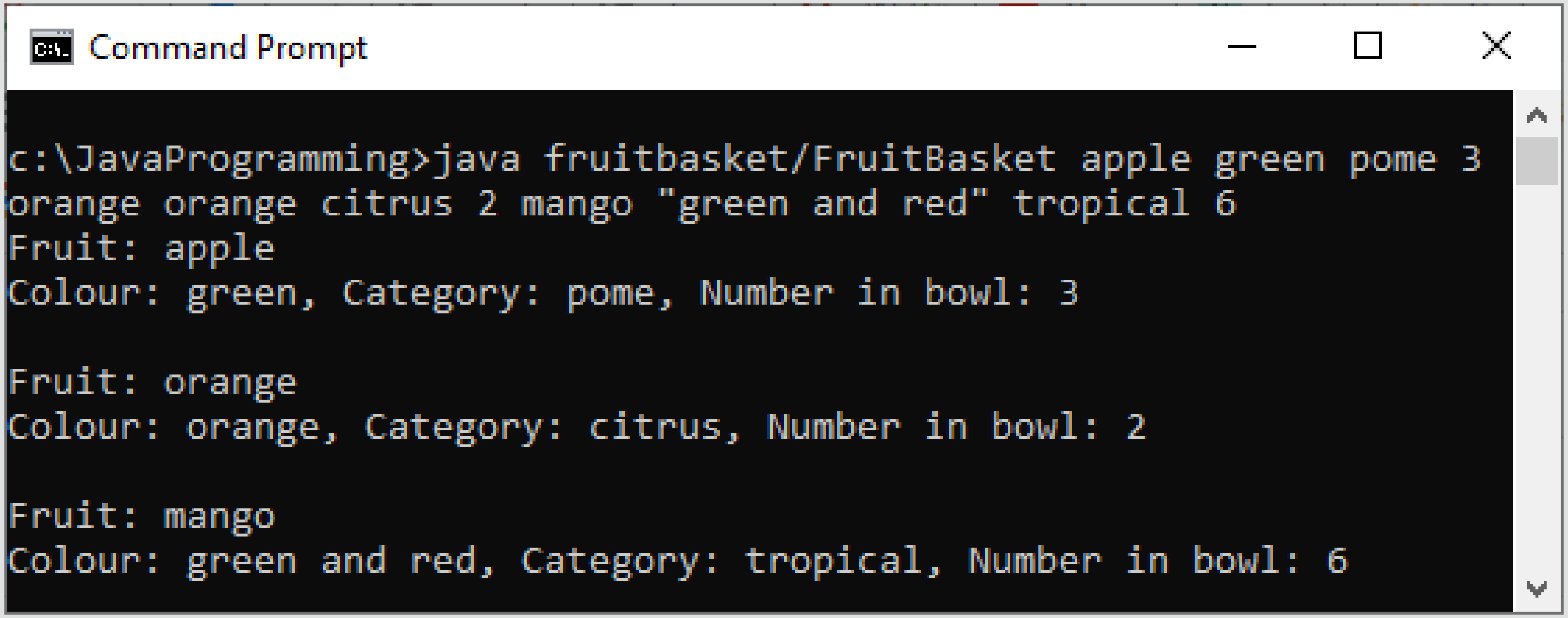
8. Run the program providing a variety of argument lists based on the following table:

name	**colour**	**category**	**number**
apple	green	pome	3
apple	red	pome	4
orange	orange	citrus	2
plum	purple	stone	1
banana	yellow	tropical	6
mango	green and red	tropical	1
raspberry	dark pink	berry	12
pear	pale yellow	pome	2

ORACLE Academy

Java Application Example 3

8. Sample code: multi word arguments are enclosed in double quotes (“ “) to be treated as a single value:



```
c:\JavaProgramming>java fruitbasket/FruitBasket apple green pome 3
orange orange citrus 2 mango "green and red" tropical 6
Fruit: apple
Colour: green, Category: pome, Number in bowl: 3

Fruit: orange
Colour: orange, Category: citrus, Number in bowl: 2

Fruit: mango
Colour: green and red, Category: tropical, Number in bowl: 6
```

ORACLE Academy

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
  - other information it extracts from loaded class files(Strings, integer and metadata and other literals)

JVM Runtime data area

- The decisions about the structural details of the runtime data areas are left to the designers of the implementations
- 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 the JVM and its 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

- 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 the Garbage Collector
  - Is created by the JVM and the size can be specified and tuned
  - Is the memory shared by all Java threads
  - Is the runtime data area from which memory for all class instances and arrays 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 or the JVM throws an OutofMemoryError exception

JVM Runtime Data Area Example

- obj local variable (reference) located in the JVM Stack
- Instances of Object class (obj) stay in the Heap
- sobj static variable locates in the method area

```
static Object sobj;

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

JVM Runtime 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 collected 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

ORACLE
Academy