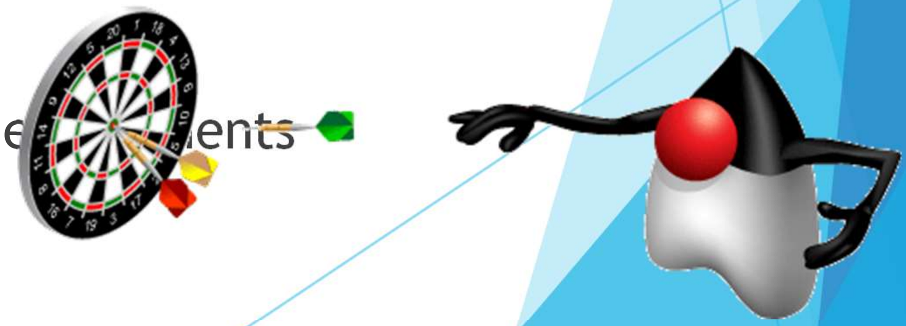
Introduction to Java

Objectives

At the end of this session, you will be able to

- Download and install Java
- Understand the Java architecture
- Compile and run Java applications
- Understand the language fundamentals
- Use the control structures in programs
- Use arrays and strings
- Write programs using command line [illegible]ents



Agenda

- Introduction to Java
- Java Architecture
- Setting up the environment
- Language Fundamentals
- Data types and operators
- Control Structures
- Arrays and Strings
- Command line arguments

Introduction to Java

- Java is :
  - A Programming language
  - A development environment
  - A deployment environment
- Similar in syntax to C++; similar in semantics to Smalltalk
- Operating system independent
- Runs on Java Virtual Machine (JVM)
  - A secure operating environment that runs as a layer on top of the OS
  - A sandbox which protects the OS from malicious code
- Object Oriented Programming language
  - In Java, everything is a class

Features of Java

- Object oriented
- Simple
- Robust
- Architecture neutral
- Portable
- Secure
- High performance
- Interpreted
- Support for Multi-threading
- Distributed

Platform Independence

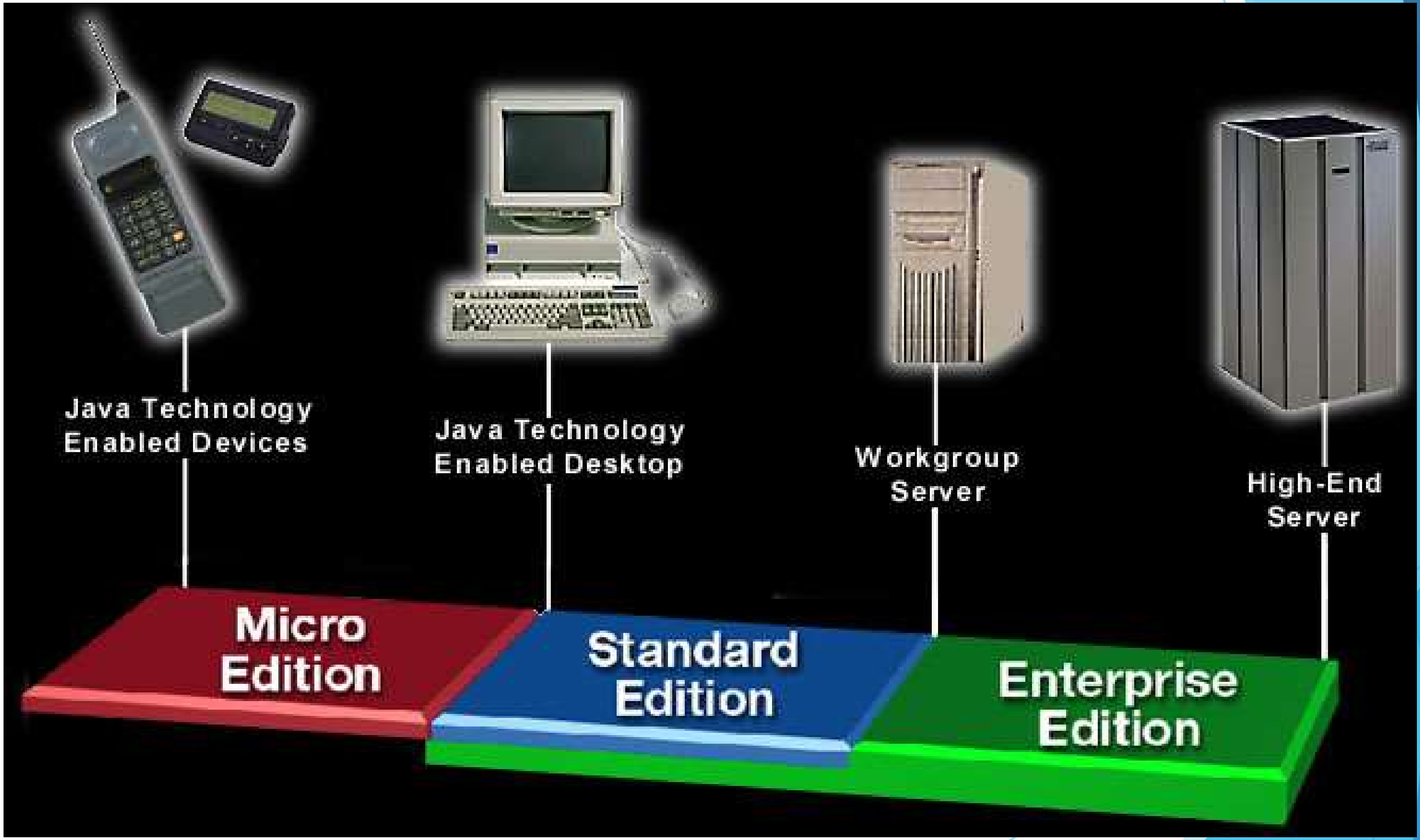
- A platform is the hardware & software environment in which a program runs

- Once compiled, java code runs on any platform without recompiling or any kind of modification

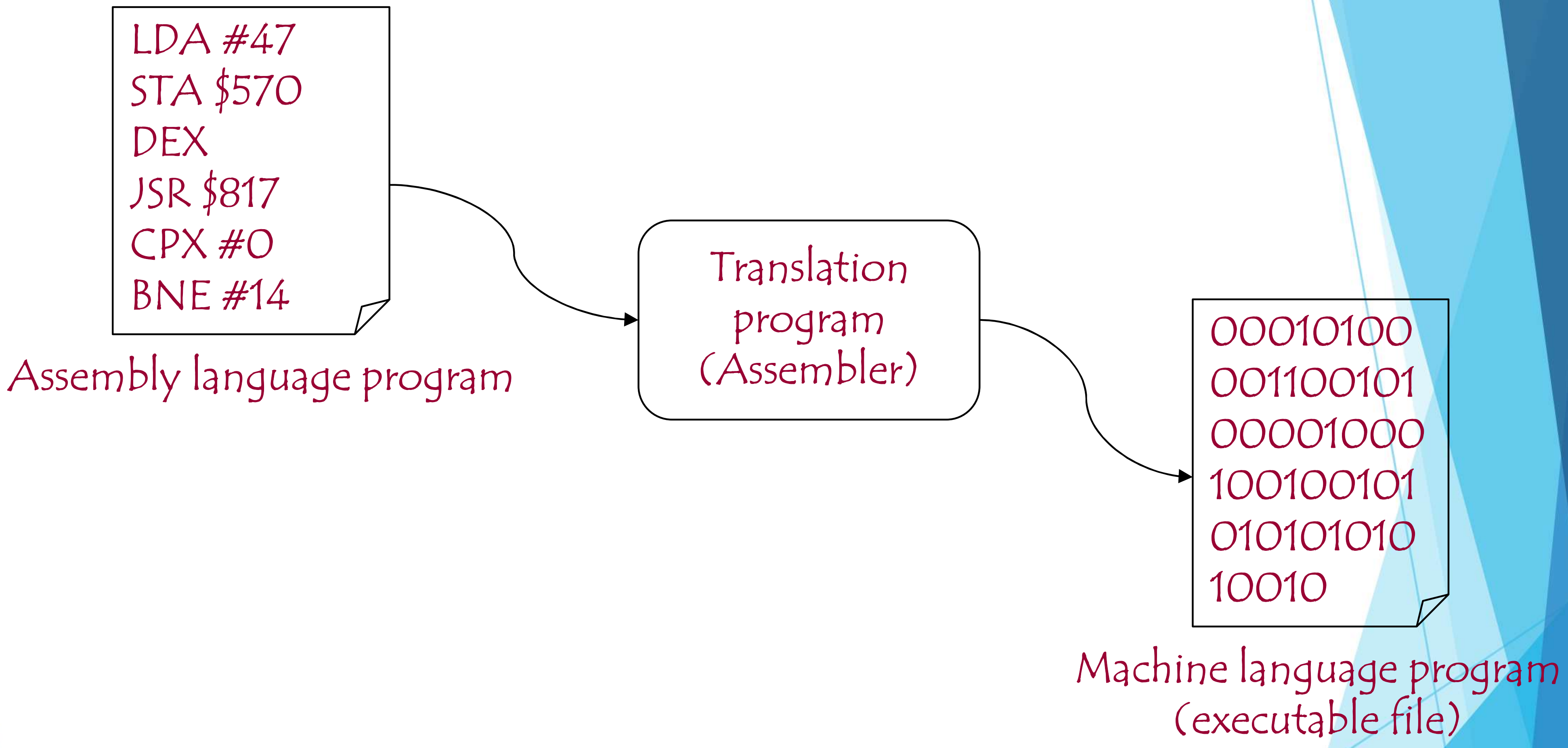
  “Write Once Run Anywhere”

- Java Virtual Machine (JVM) made this possible

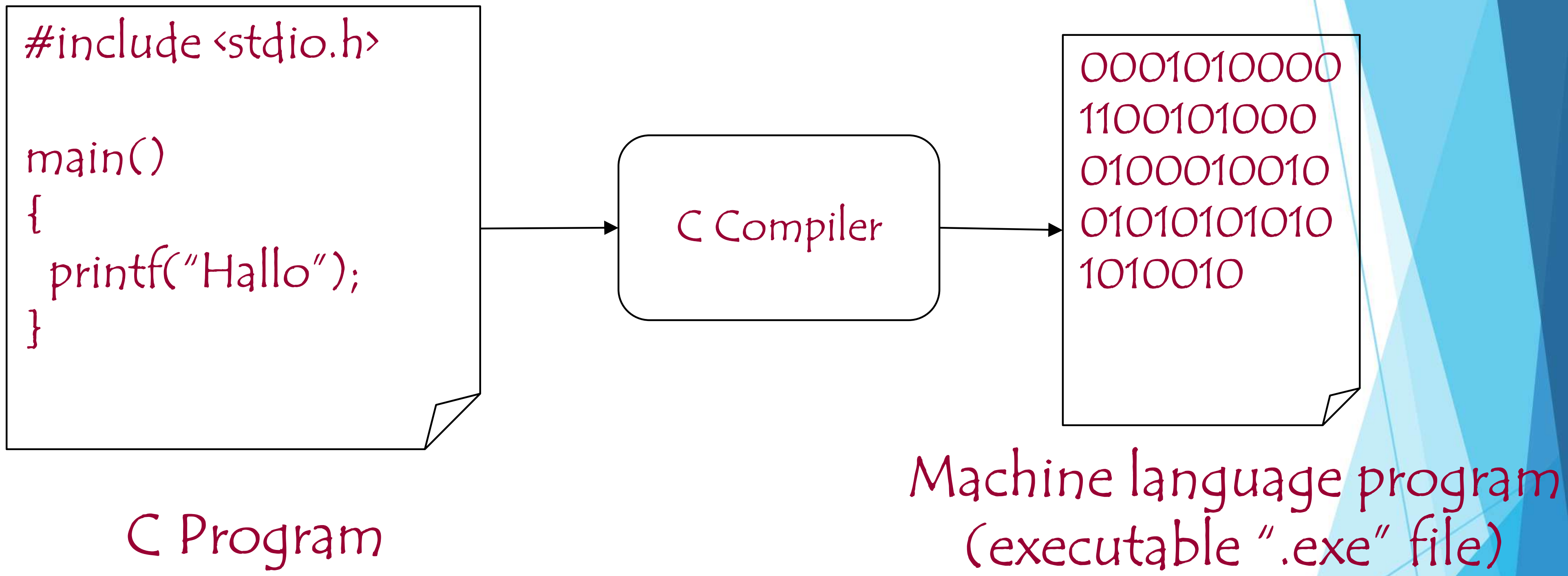
Java 2 Platform



Assembly Language


LDA #47
STA $570
DEX
JSR $817
CPX #0
BNE #14
Assembly language program
Translation program (Assembler)
00010100
001100101
00001000
100100101
010101010
10010
Machine language program (executable file)

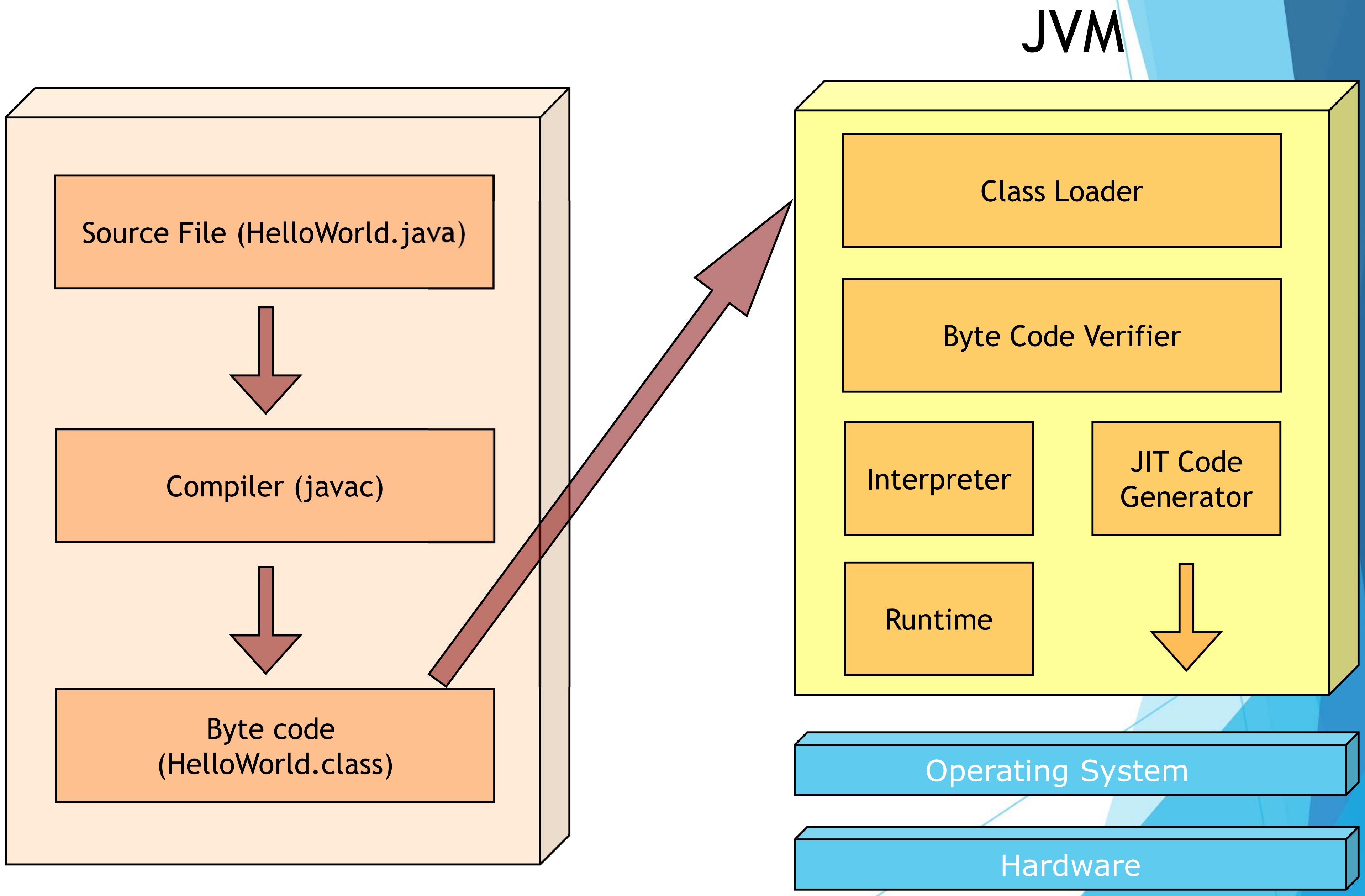
C Language



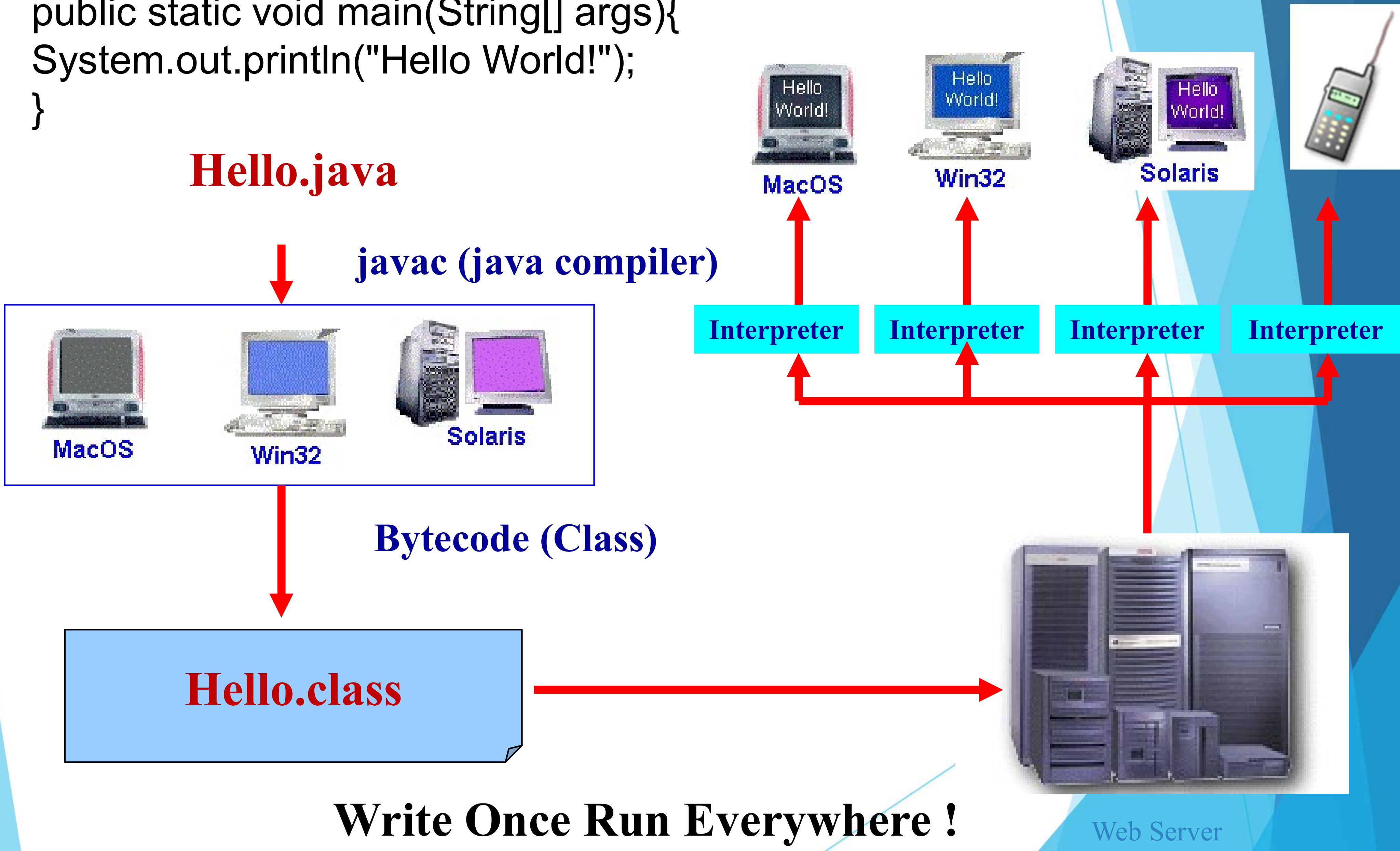
```
#include <stdio.h>

main()
{
 printf("Hallo");
}
```

```
0001010000
1100101000
0100010010
01010101010
1010010
```

Java Architecture



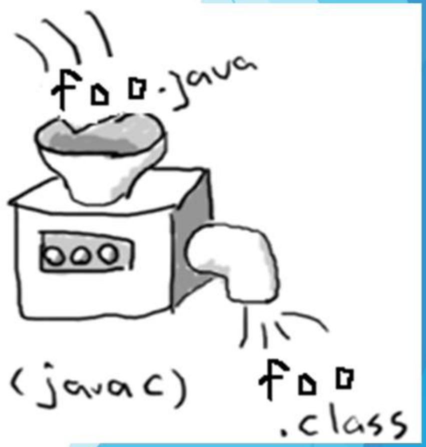
```
public class Hello
     public static void main(String[] args){
     System.out.println("Hello World!");
     }
}
```



Write Once Run Everywhere !

Java Compiler

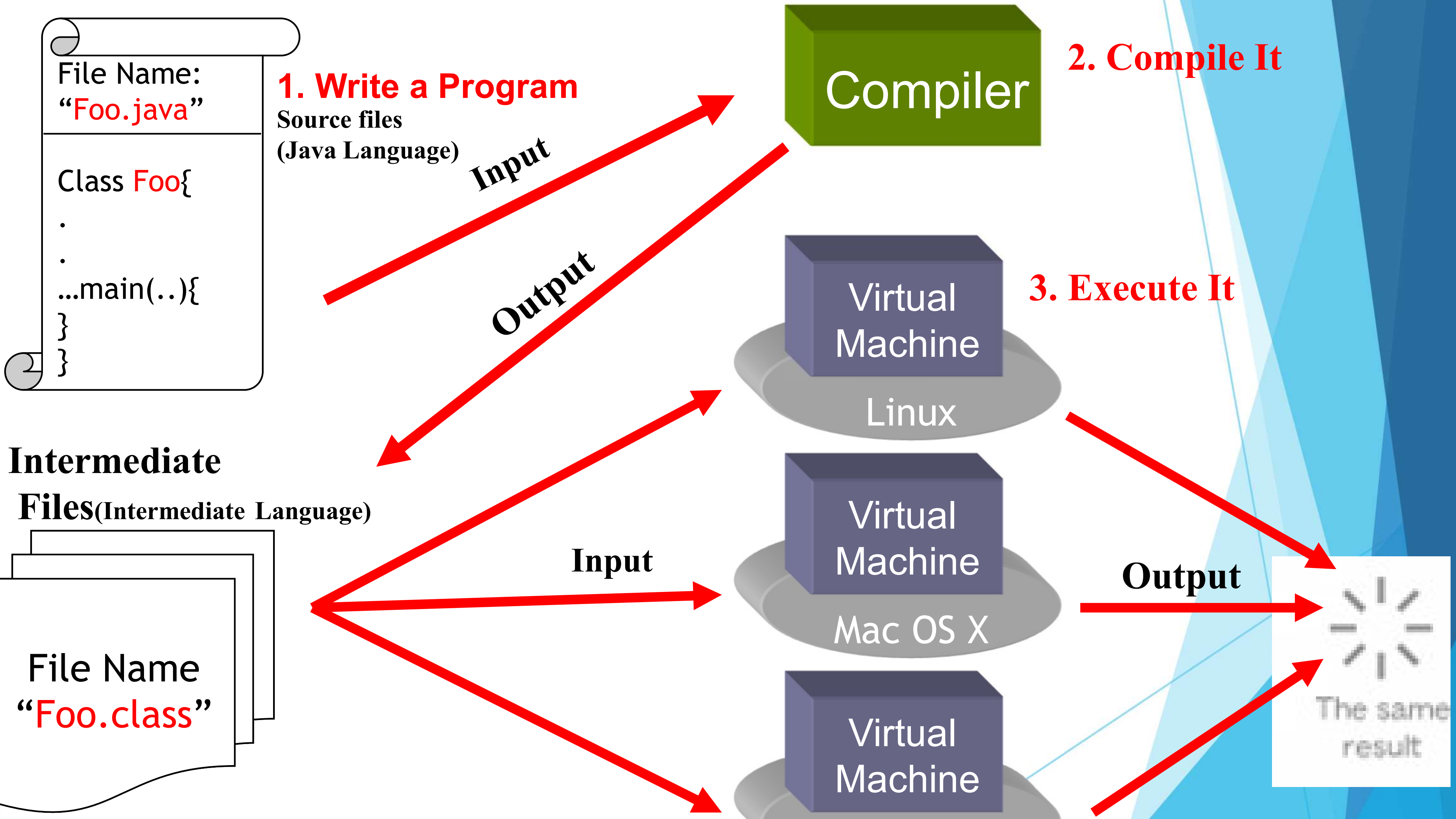
- Translates java code into byte code
- The byte code will be in a file with extension .class
- Byte code is in binary language to be interpreted by JVM
- Also performs strong type checking, prevents access violations etc.



Java Virtual Machine

- JVM makes Java platform independent
- Reads and executes compiled byte codes
- Is implemented as software and is specific to each platform
- The JVM interprets the .class file to the machine language of the underlying platform
- The underlying platform then processes the commands given by the JVM
- JVM provides definitions for the
  - Instruction set (CPU)
  - Class file format
  - Stack
  - Garbage-collection
  - Memory management

Development and Execution Infrastructure

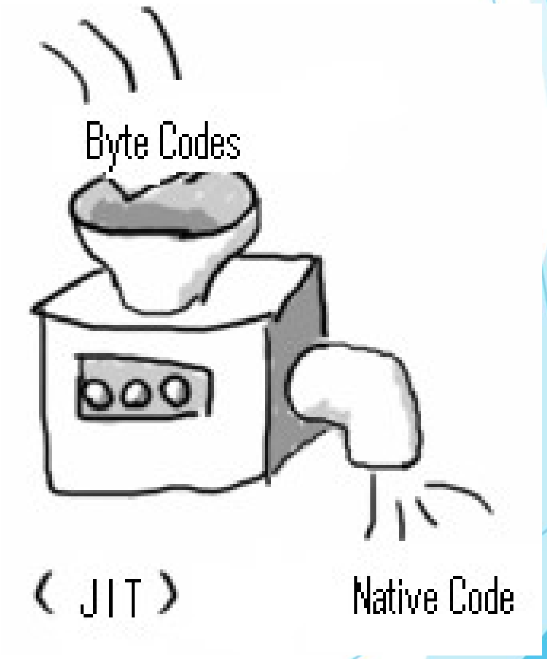


Just-In-Time Compiler (JIT)

- Converts a part of the byte code to native code.

- Requires more memory because both byte code and the corresponding native

  code are in memory at the same time.



Java SE Platform Versions

Version Name	Code Name	Release Date
JDK 1.0	Oak	January 1996
JDK 1.1	(none)	February 1997
J2SE 1.2	Playground	December 1998
J2SE 1.3	Kestrel	May 2000
J2SE 1.4	Merlin	February 2002
J2SE 5.0	Tiger	September 2004
Java SE 6	Mustang	December 2006
Java SE 7	Dolphin	July 2011
Java SE 8	*Spider*	March 2014

Java SE Development Kit (JDK)

- Allows programmers to create, compile, and execute Java programs on a particular platform
- Includes the command-line Java compiler (`javac`) and the Java Runtime Environment (JRE)
- The JRE provides the runnable Java platform which supplies the `java` command needed to execute Java applications
- Download and install **JDK 7** from http://www.oracle.com/technetwork/java/javase/downloads/index.html

Environment Variables

- JAVA_HOME: Java Installation Directory

Windows	`set JAVA_HOME=C:\jdk1.7.0_51`
UNIX	`export JAVA_HOME=/var/usr/java`

- CLASSPATH: Used to locate class files

Windows	`set CLASSPATH=%CLASSPATH%;%JAVA_HOME%\lib\tools.jar`
UNIX	`set CLASSPATH=$CLASSPATH:$JAVA_HOME/lib/tools.jar`

- PATH: Used by OS to locate executable files

Windows	`set PATH=%PATH%;%JAVA_HOME%\bin;.`
UNIX	`set PATH=$PATH:$JAVA_HOME/bin:.`

Hello World Program

- Type the source code using any text editor
- Save this as HelloJava.java

```
class HelloJava {
  public static void main(String[] args) {
    System.out.println(“Hello World”);
  }
}
```

- Compilation:

```
C:\> javac HelloJava.java
```

- Running your first java program:

```
C:\> java HelloJava
```

Try it out

1. The Java compiler for Windows and Linux are same (True/False)

2. The JVM for Windows and Linux are same. (True/False)

3. Java compiler translates .java source files into ___________?

   a) .exe file b) .bat file c) .doc file d) .class file

4. _________ converts the bytecode to native code.

   a) JRE b) JVM c) JIT d) JDK

5. Which statement is used to compile a source code file HelloJava.java?.

   a) C:\> java HelloJava.java b) C:\> javac HelloJava

   c) C:\> javac HelloJava.java d) C:\> javac Hello.Java.java

Java Language Fundamentals

Comments

- A single line comment starts with //

  // This is a single line comment in Java

- A multi line comment starts with /* and ends with */

  /* This is a

  Multi line

  Comment in Java */

Keywords

abstract	continue	for	new	switch
assert	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const*	float	native	super	while

* Keywords not in use now

Identifiers & Literals

- An identifier (name) must begin with a letter, a dollar sign ($) or an underscore
  character (_). Subsequent characters may be letters, $, _ or digits.

- Classes, methods and variables cannot have Java Keywords as Identifiers
  (names).

- true, false and null are literals (not keywords), but they can't be used as
  identifiers as well.

Variables

- A named storage location in the computer's memory that stores a value of a particular type for use by program.
- Example :

```
int myAge=28, cellPhone;
double salary;
char tempChar;
```

- Variables can be declared anywhere in the program

```
for (int count=0; count < max; count++) {
   int z = count * 10;
}
```

- If a local variable is used without initializing, the compiler will show an error. The below is an error

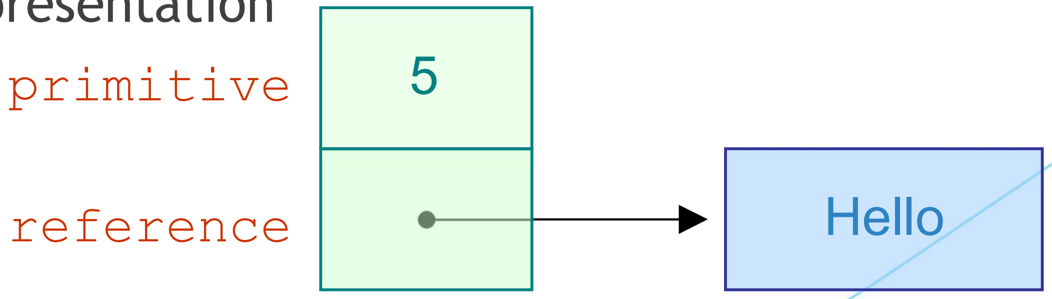
```
int tempVal;
System.out.println(tempVal);
```

Data Types

- The Data type can either be:
  - Built-in primitive type
    - Holds value
  - Reference data type
    - Holds reference to objects
- Example

```
int primitive = 5;
String reference = “Hello” ;
```

- Memory Representation



Primitive Data Types

Data Type	Size (bits)	Type	Signed	Default Value	Min Value	Max Value
boolean	1	N/A	N/A	false	false	True
char	16	Single Unicode Character	×	'\u0000'	'\u0000' (0)	'\uFFFF' (2^{16} – 1)
byte	8	Integer	✓	0	-128(-2^{7})	127 (2^{7} – 1)
short	16	Integer	✓	0	-2^{15}	2^{15} – 1
int	32	Integer	✓	0	-2^{31}	2^{31} – 1
long	64	Integer	✓	0L	-2^{63}	2^{63} – 1
float	32	Floating Point Number	✓	0.0F	1.4E-45	3.4028235E38
double	64	Floating Point Number	✓	0.0	4.9E-324	1.7976931348623157 E308

Append uppercase or lowercase "L" or "F" to the number to specify a long or a float number.

Unicode Character Set

- char data type in Java is 2 bytes because it uses UNICODE character set to support internationalization

- UNICODE character set supports all known scripts and languages in the world

Operators

Type of Operators	Operators	Associativity
Postfix operators	[] . (parameters) ++ --	Left to Right
Prefix Unary operators	++ -- + - ~ !	Right to Left
Object creation and cast	new (type)	Right to Left
Multiplication/Division/Modulus	* / %	Left to Right
Addition/Subtraction	+ -	Left to Right
Shift	>> >>> <<	Left to Right
Relational	< <= > >= instanceof	Left to Right
Equality	== !=	Left to Right
Bit-wise/Boolean AND	&	Left to Right
Bit-wise/Boolean XOR	^	Left to Right
Bit-wise/Boolean OR	|	Left to Right
Logical AND (Short-circuit or Conditional)	&&	Left to Right
Logical OR (Short-circuit or Conditional)	||	Left to Right
Ternary	? :	Right to Left
Assignment	= += -= *= /= %= <<= >>= >>>= &= ^= |=	Right to Left

Conversions and Casting

- Automatic type changing is known as **Implicit Conversion**
  - A variable of smaller capacity can be assigned to another variable of bigger capacity. This is called widening

```
int i = 10;

double d;

d = i;
```

  - Legal conversions are

$$\begin{tikzcd}
\textbf{byte} \arrow[r] & \textbf{short} \arrow[r] & \textbf{int} \arrow[r] & \textbf{long} \arrow[r] & \textbf{float} \arrow[r] & \textbf{double} \\
& & \textbf{char} \arrow[u] & & &
\end{tikzcd}$$

- Whenever a larger type is converted to a smaller type, we have to explicitly specify the type cast operator as narrowing is not allowed

```
double d = 10

int i;

i = (int) d;
```

Type cast operator

Control Structures

- Work the same as in C / C++

  `if/else, for, while, do/while, switch`

```
i = 0;
while(i < 10) {
    a += i;
    i++;
}
```

```
for(i = 0; i < 10; i++) {
    a += i;
}
```

```
i = 0;
do {
    a += i;
    i++;
} while(i < 10);
```

```
if(a > 3) {
    a = 3;
}
else {
    a = 0;
}
```

```
switch(i) {
    case 1:
        msg = “foo”;
         break;
    case 2:
        msg = “bar”;
         break;
    default:
        msg = “”;
}
```

Flow Control Statements

- Java supports continue & break keywords

- A break statement will cause the current iteration of the innermost loop to stop and the next line of code following the loop to be executed.

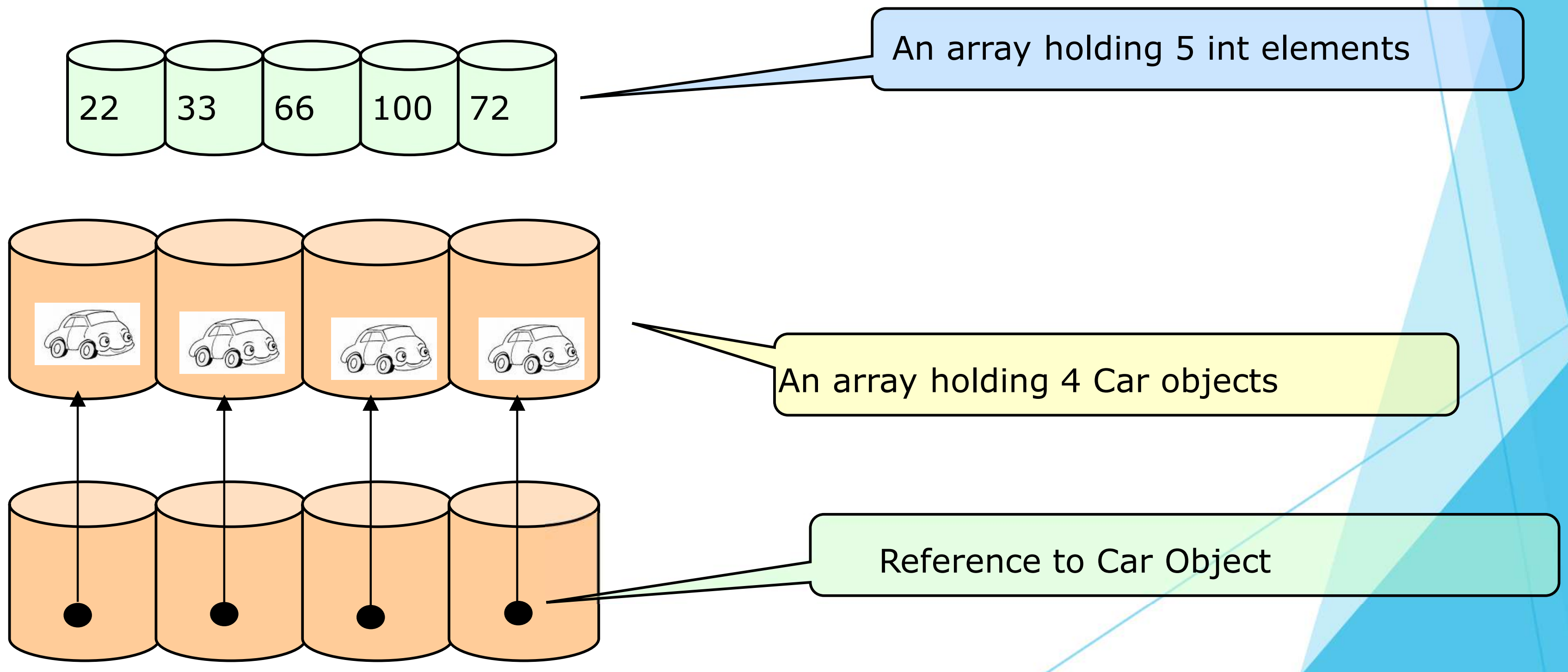
- A continue statement will cause the current iteration of the innermost loop to stop, and the condition of that loop to be checked, and if the condition is met, perform the loop again.

```
for(i = 0; i < 10; i++) {
    if(i == 5)
        continue;
    a += i;
}
```

```
for(i = 0; i < 10; i++) {
    a += i;
    if(a > 100)
        break;
}
```

Arrays

- An ordered collection of homogeneous data elements
- Size to be specified at compile time and is static – cannot be modified
- Arrays in Java are objects and can be of primitive data type or reference type
- The length property on arrays tells the size of the array


22
33
66
100
72
An array holding 5 int elements
An array holding 4 Car objects
Reference to Car Object

Arrays

- Arrays should be

  Declared

  ```
  int[] a; String b[]; Object []c;
  ```

  Allocated (constructed)

  ```
  a = new int[10];
      b = new String[arraysize]
  ```

  Initialized

  ```
  for (int i = 0; i < a.length; a[i++] = 0);
  ```

- An array can also be initialized while it is declared as follows:

  ```
  int [] x = {1, 2, 3, 4};

  char [] c = {‘a’, ‘b’, ‘c’};
  ```

- Unlike C, Java checks the boundary of an array while accessing an element in it

Multidimensional Arrays

- A Multi-dimensional array is an array of arrays

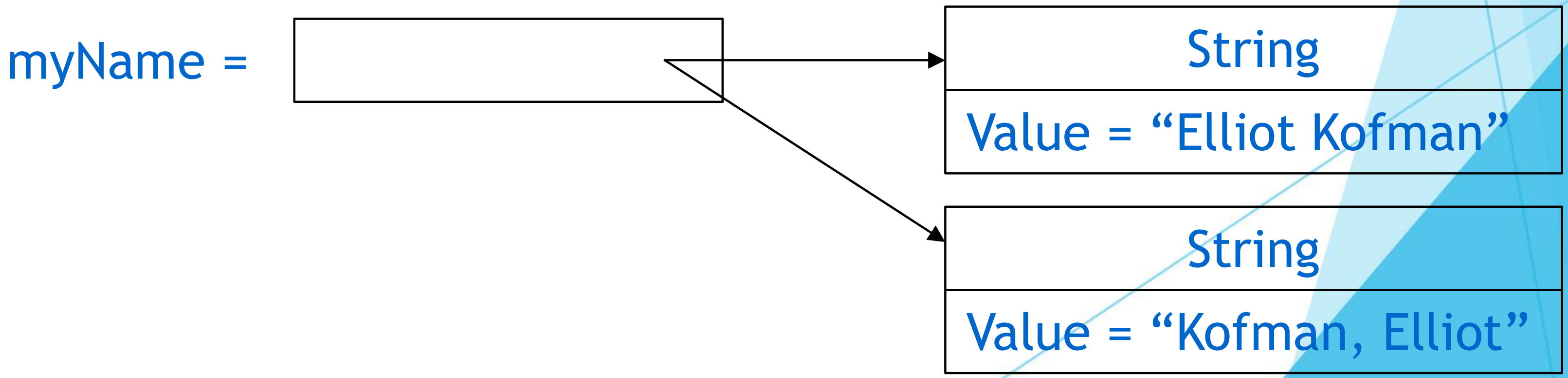
- To declare a multidimensional array, specify each additional index using another set of square brackets

```
int [][] x;
//x is a reference to an array of int arrays

x = new int[3][4];
//Create 3 new int arrays, each having 4 elements

//x[0] refers to the first int array, x[1] to the second and so on
//x[0][0] is the first element of the first array
//x.length will be 3
//x[0].length, x[1].length and x[2].length will be 4
```

Strings

- An object of the String class represents a fixed length, immutable sequence of unicode characters
- All String operations (concat, trim, replace, substring etc) construct and return new strings.
- String objects are immutable. If modified, Java creates a new object having the modified character sequence

```
String myName = “Elliot Koffman”;
myName = “Koffman, Elliot”;
```



Strings

```
public class StringDemo {

   public static void main(String args[]){

       char letter = 'a';

       String string1 = "Hello";     // String literals are stored as String objects
       String string2 = "World";
       String string3 = "";
       String dontDoThis = new String ("Bad Practice");    // Use is discouraged

      string3 = string1 + string2;       // + is to concatenate strings

      System.out.println("Output: " + string3 + " " + letter);

   }
}
```

String Operations

```
public class StringOperations {
    public static void main(String arg[]){
        String string2 = "World";
        String string3 = "";

        string3 = "Hello".concat(string2);
        System.out.println("string3: " + string3);

        // Get length
       System.out.println("Length: " + string1.length());

       // Get SubString
       System.out.println("Sub: " + string3.substring(0, 5));

       // Uppercase
       System.out.println("Upper: " + string3.toUpperCase());
    }
}
```

Arrays and for-each loop

```
public class ArrayOperations {
  public static void main(String args[]){

     String[] names = new String[3];

     names[0] = "Blue Shirt";
     names[1] = "Red Shirt";
     names[2] = "Black Shirt";

     int[] numbers = {100, 200, 300};

     for (String name:names){
        System.out.println("Name: " + name);
     }

     for (int number:numbers){
        System.out.println("Number: " + number);
     }
  }
}
```

Command Line Arguments

- Information that follows program's name on the command line when it is executed

- This data is passed to the application in the form of String arguments

```
class Echo {
    public static void main (String args[]) {
        for (int i = 0; i < args.length; i++)
            System.out.println(args[i]);
    }
}
C:\> java Echo Drink Hot Java
Drink
Hot
Java
```

Try it out

1. Which of the following conversions are legal?

   a) byte to int   b) int to char   c) float to double   d) double to int

2. Array index starts at ___. Index is of ____ data type.

   a) 1,int   b) null, char   c) 0,int

3. Java supports pointer arithmetic. (True/False)

4. Which of the below is a legal identifier?

   a) _a   b) –a   c) 7g   d) my Name

5. Will the below code compile?

```
int[5] scores;
```

6. Local variables get default values if not initialized. (True/False)

Summary

In this session, we have covered:

- Java2 platform and its components
- Java language fundamentals
- Identifiers & Literals
- Primitive Data Types & their Conversion
- Operators & basic flow controls in java
- Arrays & Strings
- Command line arguments

Thank you