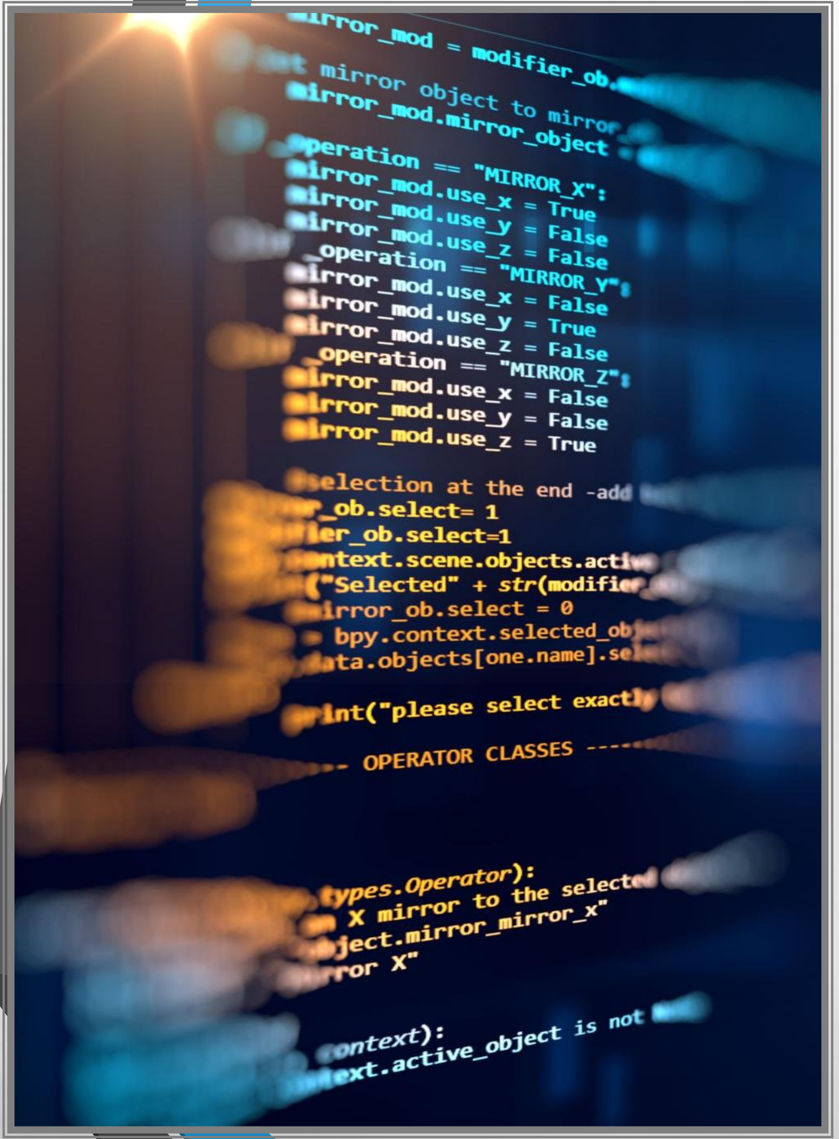
PSOOP

Lecture 01

Outline

- Introduction to Java
- Features of Java
- Java Architecture
- Source File Layout - Hello World
- Java Keywords, Identifiers and Data Types
- Primitive Data Types
- Variables
- Comments in Java
- Reference Data Types
- Java Operators
- Control Structures



Introduction to Java

- A high level programming language introduced by James Gosling
- 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
  - Unlike C++, OOP support is a fundamental component in Java

Features of Java

- Object Oriented
- Simple
  - Compared to earlier OO languages like C++, it is simple
- Robust
- Secure
  - Absence of pointers
- Support for Multithreading at language level
- Designed to handle Distributed applications
- Architecture Neutral / Portable:
  - Java code compiled on Windows can be run on Unix without recompilation

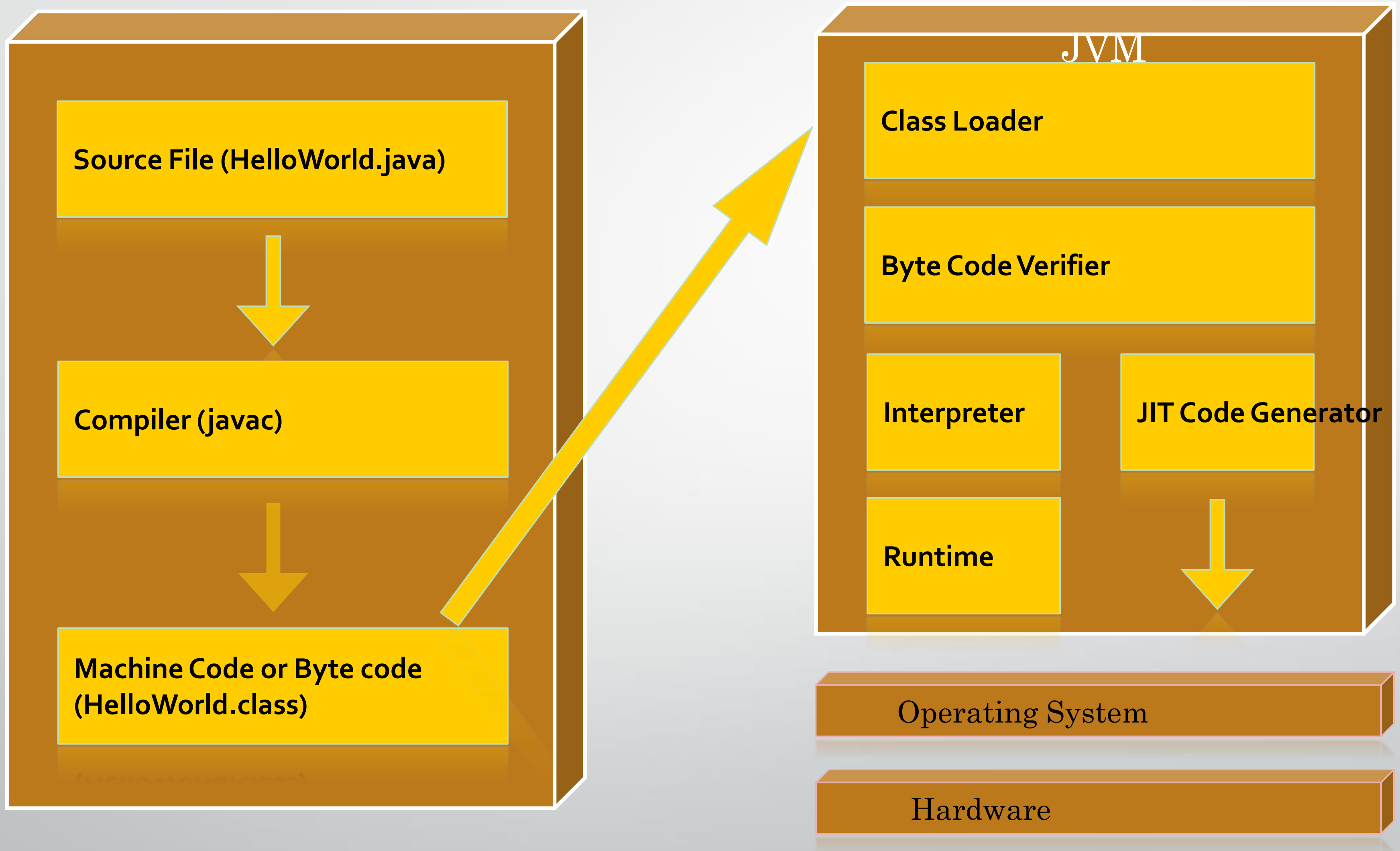
Platform Independence

- 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”

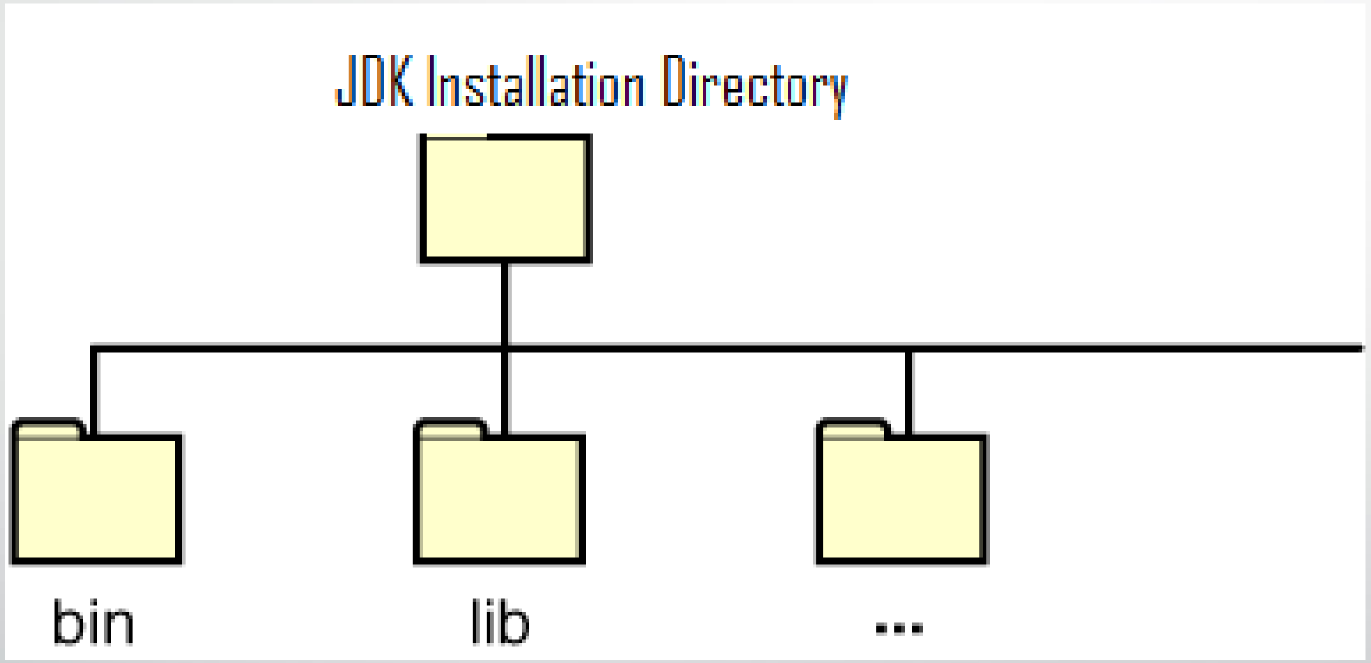
- This is made possible by the Java Virtual Machine (JVM)

Java Architecture



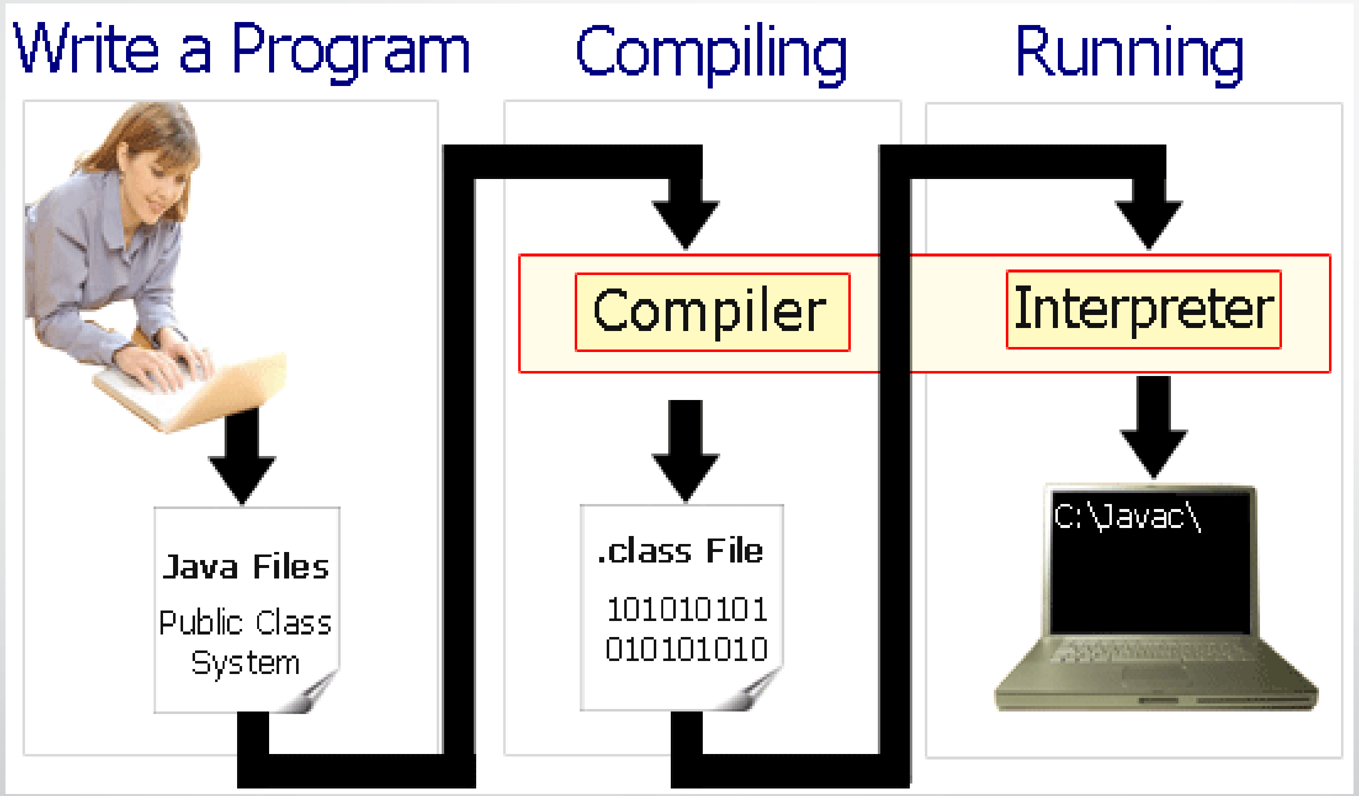
JDK Directory Structure

- After installing the software, the JDK directory will have the structure as shown



- The *bin* directory contains both, the compiler and the interpreter

Java Development Process



Java Virtual Machine (JVM)

- The source code of Java is stored in a text file with the extension .java
- The Java compiler compiles a .java file into byte code
- The byte code will be in a file with extension .class
- The generated .class file is the machine code of this processor
  - Byte code is in binary language
- The byte code is interpreted by the JVM

Java Virtual Machine (JVM) (Contd…)

- JVM makes Java platform independent
- The JVM interprets the .class file to the machine language of the underlying platform
- The underlying platform processes the commands given by the JVM

Source File Layout - Hello World

- Type the source code using any text editor

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

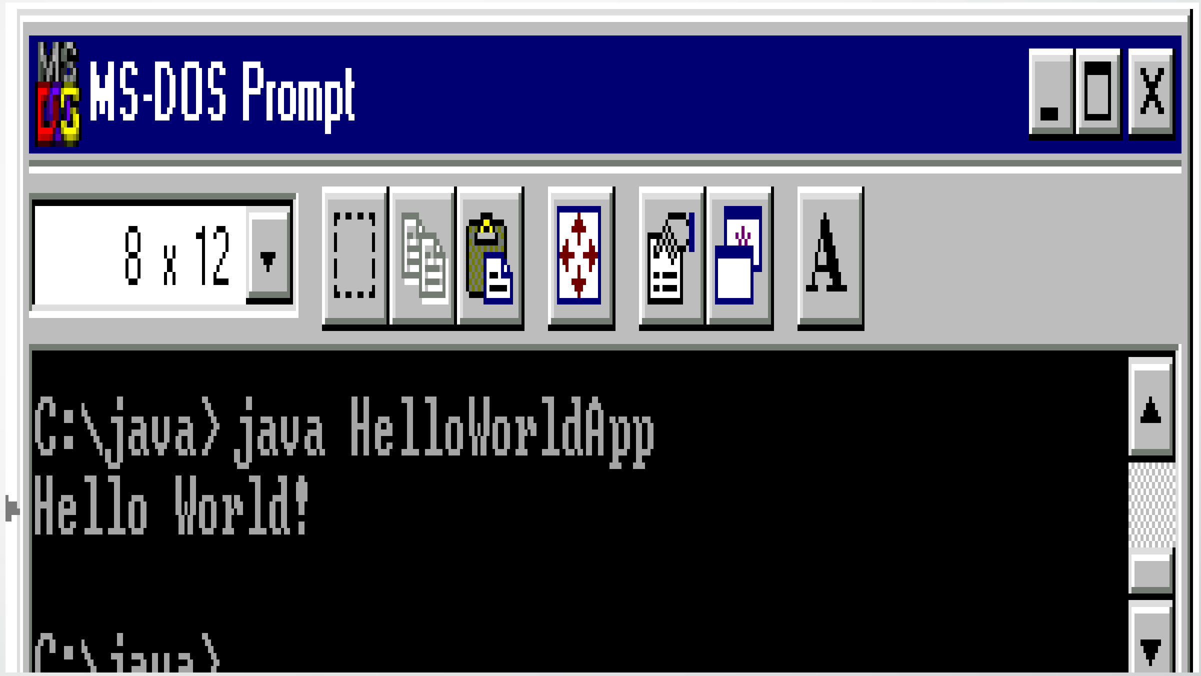
- Save this file as *HelloWorld.java*

To Compile

- Open the command prompt
- Set the environment variables
- Go to the directory in which the program is saved
- Type - `javac HelloWorldApp.java`
- If it says, “bad command or file name” then check the path setting
- If it returns to prompt without giving any message, it means that compilation is successful

To Run the class file

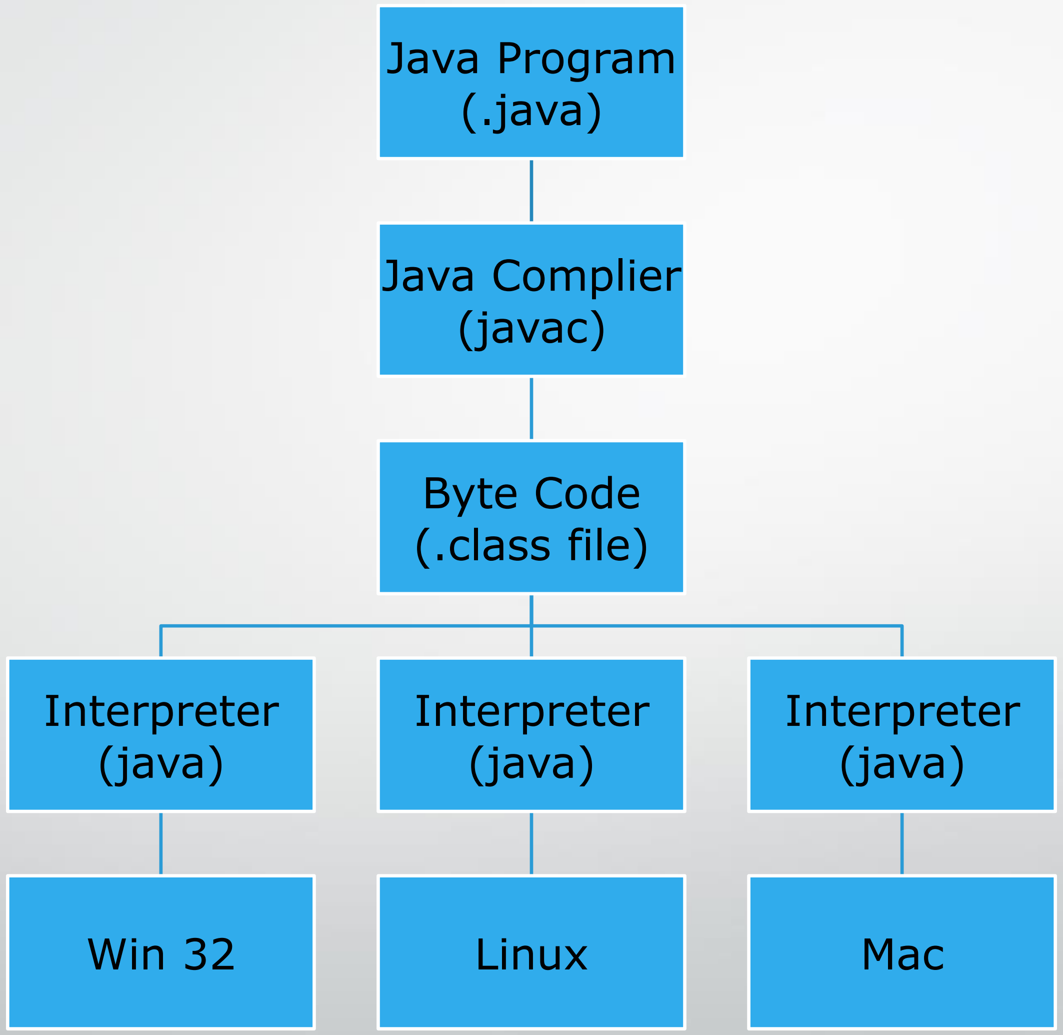
- Type the command - java HelloWorldApp
- The result will be



```
C:\java>java HelloWorldApp
Hello World!

C:\java>
```

Compilation & Execution



Best Practices

- Only put one class in one source file
- Provide adequate comments in the program
- Properly indent the program
- Follow coding standards for identifiers

Java Keywords

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

* Keywords not in use now

Java Identifiers

- Declared entities such as variables, methods, classes & interfaces are Java Identifiers
- Must begin with a letter, underscore (_) or dollar sign ($)
- May contain letters, digits, underscore(_) & dollar sign ($)

Data Types in Java

- Java is a strongly typed language
  - Unlike C, type checking is strictly enforced at run time
  - Impossible to typecast incompatible types
- Data types may be:
  - Primitive data types
  - Reference data types

Primitive Data Types in Java

Integer Data Types

byte	(1 byte)
short	(2 bytes)
int	(4 bytes)
long	(8 bytes)

Floating Data Types

float	(4 bytes)
double	(8 bytes)

Character Data Types

char	(2 bytes)

Logical Data Types

boolean	(1 bit) (true/false)

- All numeric data types are signed
- The size of data types remain same on all platforms
- *char* data type is 2 bytes as it uses the UNICODE character set.

Variables

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

```
DataType      variableName
int         myAge, cellPhone;
double          salary;
char        tempChar;
```

- The data type can either be:
  built-in *primitive* types (e.g. int, double, char object classes)
  reference data types (e.g. String,BufferedReader)

Naming Convention →

Variable Name: First word lowercase & rest initial capitalized (Camel Casing) e.g. thisIsALongVariableName

Variables (Contd…)

- Using primitive data types is similar to other languages

```
int count;
int max=100;
```

- Variables can be declared anywhere in the program

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

BEST PRACTICE
Declare a variable in program only when required
Do not declare variables upfront like in C

- In Java, if a local variable is used without initializing it, the compiler will show an error

Give this a Try…

- How many of these are valid Java Identifiers?

```
78class       Class87       sixDogs
User$ID           Jump_Up_      DEFAULT_VAL
False             Private       Average-Age
Hello!        First One     String

A. 5
B. 6
C. 7
D. 8
E. 9
```

Give this a Try…

- What will be the output of the following code snippet when you try to compile and run it?

```
class Sample{
   public static void main (String args[]){
       int count;
       System.out.println(count);
   }
}
```

Comments in Java

- A single line comment in Java starts with //

```
// This is a single line comment in Java
```

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

```
/* This is a multi line
   comment
   in Java */
```

Reference Data Types

- Hold the reference of dynamically created objects which are in the heap
- Can hold three kinds of values:
  - Class type: Points to an object / class instance
  - Interface type: Points to an object, which is implementing the corresponding interface
  - Array type: Points to an array instance or "*null*"

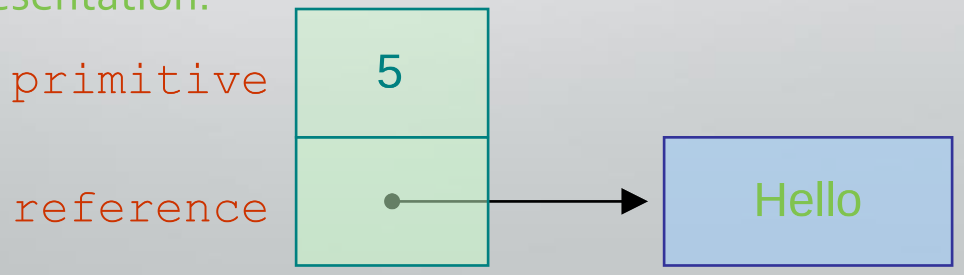
- Difference between Primitive & Reference data types:
  - Primitive data types hold values themselves
  - Reference data types hold reference to objects, i.e. they are not objects, but reference to objects

Reference Data Types (Contd...)

- Objects & Arrays are accessed using *reference variables* in Java
- A reference variable is similar to a pointer (stores memory address of an object)
- Java does not support the explicit use of addresses like other languages
- Java does not allow pointer manipulation or pointer arithmetic

```
int primitive = 5;
String reference = "Hello" ;
```

- Memory Representation:



Reference Data Types (Contd…)

- A reference type cannot be cast to primitive type
- A reference type can be assigned ‘null’ to show that it is not referring to any object

Typecasting Primitive Data Types

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

```
int i = 10;
 double d;
d = i;
```

- Whenever a larger type is converted to a smaller type, we have to explicitly specify the *type cast operator*

```
double d = 10
int i;
i = (int) d;
```

Type cast operator

- This prevents *accidental loss* of data

Java Operators

- Used to manipulate primitive data types
- Classified as unary, binary or ternary
- Following are different operators in Java:
  - Assignment
  - Arithmetic
  - Relational
  - Logical
  - Bitwise
  - Compound assignment
  - Conditional

note

Java Operators (Contd…)

Assignment Operators	=
Arithmetic Operators	- + * / % ++ --
Relational Operators	> < >= <= == !=
Logical Operators	&& || & | ! ^
Bit wise Operator	& | ^ >> >>>
Compound Assignment Operators	+= -= *= /= %= <<= >>= >>>=
Conditional Operator	?:

Give this a Try…

```
int x = 5;
int y = 10;
int z = ++x * y--;
```

- What is the result of the following code fragment?

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:
 string = "foo";
case 2:
  string = "bar";
default:
  string = "";
}
```

Control Structures (Contd…)

- Java supports continue & break keywords also
- Again, work very similar to as in C / C++
- Switch statements require the condition variable to be a char, byte, short or int

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

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

Give this a Try…

- What do you think is the output if aNumber is 3?

```
if (aNumber >= 0){


 if (aNumber == 0)
   System.out.println("first string");
else
 System.out.println("second string");
 System.out.println("third string");
}
```

Thank You