Java vs C++

Java	C++
Platform Independent	C++ is not platform independent
Used for application programming (window/web mobile)	Used mainly for system programming
Doesn’t support goto statement	Supports goto statement
Doesn’t support multiple inheritance	Support multiple inheritance
Doesn’t support operator overloading	Support operator overloading
Supports pointers internally, but restricted	Support pointers
Java uses compiler and interpreter	Only uses compiler
Only support call by value and not call by reference	Support both call by value and call by reference
Doesn’t support structure and union	Support structure and union
Java has built in thread support	Doesn’t have built in support for thread, relies in third party libraries

Basic Syntax

1. Java is case sensitive. Identifier Hello and hello have different meaning in Java
2. Every statement should end with semicolon (;)
3. Java program processing starts from the main() method which is a mandatory part of every Java program.
4. Every open brackets should have closed brackets

Java Naming conventions

Name	Convention
class name	should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc.
interface name	should start with uppercase letter and be an adjective e.g. Runnable, Remote, ActionListener etc.
method name	should start with lowercase letter and be a verb e.g. actionPerformed(), main(), print(), println() etc.
variable name	should start with lowercase letter e.g. firstName, orderNumber etc.
package name	should be in lowercase letter e.g. java, lang, sql, util etc.
constants name	should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc

Note:- Java **naming convention** is a rule to follow as you decide what to name your identifiers such as class, package, variable, constant, method etc. But, it is not forced to follow.

Java Identifiers

All Java components require names. Names used for classes, variables, and methods are called identifiers.

- All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_).
- After the first character, identifiers can have any combination of characters.
- A keyword cannot be used as an identifier.
- Most importantly, identifiers are case sensitive.
- No whitespace allowed

Java Keywords

- Keywords are the reserved words in Java
- The reserved words may not be used as constant or variable or any other identifier names.
- Eg: abstract, Boolean, break, byte, case, catch, char, class, const, continue, default, do, double, else, extends, final, finally, float, for, goto, if, implements, import, instanceof, int, interface, long, native

Comments in Java

All characters available inside any comment are ignored by Java compiler.

There are three ways for commenting in Java :-

1. Single line comment //
2. Multiline comment /* */
3. Documentation comment /** */

Data types

1. Primitive

There are eight primitive datatypes supported by Java. Primitive datatypes are predefined by the language and named by a keyword.

1. Non Primitive

Reference variables are created using defined constructors of the classes. They are used to access objects. These variables are declared to be of a specific type that cannot be changed.

Class objects and various type of array variables come under reference datatype. Default value of any reference variable is null. A reference variable can be used to refer any object of the declared type or any compatible type.

Data type	Size	Default Value
byte	1 byte	0
short	2 byte	0
int	4 byte	0
long	8 byte	0L
char	2 byte	‘\u0000’
float	4 byte	0.0f
double	8 byte	0.0d
boolean	1 bit	false

Variables

A variable provides us with named storage that our programs can manipulate. Each variable in Java has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

Variables have to be declared before they can be used.

The syntax is:-

data type variable [= value][, variable [= value] ...] ;

There are three types of variables

1. Local Variable
2. Instance Variable
3. Static variable

Local Variables

- Local variables are declared in methods, constructors, or blocks.
- Local variables are created when the method, constructor or block is entered and the variable will be destroyed once it exits the method, constructor, or block.
- Access modifiers cannot be used for local variables.
- Local variables are visible only within the declared method, constructor, or block.
- Local variables are implemented at stack level internally.
- There is no default value for local variables, so local variables should be declared and an initial value should be assigned before the first use.

Instance Variable

- Instance variables are declared in a class, but outside a method, constructor or any block.
- When a space is allocated for an object in the heap, a slot for each instance variable value is created.
- Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.
- Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.
- Instance variables can be declared in class level before or after use.
- Access modifiers can be given for instance variables.
- The instance variables are visible for all methods, constructors and block in the class. Normally, it is recommended to make these variables private (access level). However, visibility for subclasses can be given for these variables with the use of access modifiers.
- Instance variables have default values. For numbers, the default value is 0, for Booleans it is false, and for object references it is null. Values can be assigned during the declaration or within the constructor.
- Instance variables can be accessed directly by calling the variable name inside the class. However, within static methods (when instance variables are given accessibility), they should be called using the fully qualified name. *ObjectReference.VariableName*.

Static Variables

- Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.
- There would only be one copy of each class variable per class, regardless of how many objects are created from it.
- Static variables are rarely used other than being declared as constants. Constants are variables that are declared as public/private, final, and static. Constant variables never change from their initial value.
- Static variables are stored in the static memory. It is rare to use static variables other than declared final and used as either public or private constants.
- Static variables are created when the program starts and destroyed when the program stops.
- Visibility is similar to instance variables. However, most static variables are declared public since they must be available for users of the class.
- Default values are same as instance variables. For numbers, the default value is 0; for Booleans, it is false; and for object references, it is null. Values can be assigned during the declaration or within the constructor. Additionally, values can be assigned in special static initializer blocks.
- Static variables can be accessed by calling with the class name *ClassName.VariableName*.
- When declaring class variables as public static final, then variable names (constants) are all in upper case. If the static variables are not public and final, the naming syntax is the same as instance and local variables.