We call **C/C++** and **Java** are **strongly typed languages** and **JavaScript** and **PERL** are **loosely typed languages.** What does it mean?

Either in C or Java, when a variable is declared, it must be informed to the compiler what data type the variable stores like integer, float, double or string etc. Observe the following variable declaration in Java.



|  |  |
| --- | --- |
| 1  2  3 | int marks = 20.              String str = "way2java.com";              boolean raining = true; |

The variable must be declared along with the data type. Similarly, an array object must be instantiated with the size. This is a feature of strongly typed languages.

Just reverse with **loosely typed language**s. Observe the variable declaration in JavaScript, an object-based language.



|  |  |
| --- | --- |
| 1  2  3 | var marks = 20.              var str = "way2java.com";              var raining = true; |

The string and data types int and boolean are replaced by var that stands for variable. Depending on the value assigned, var becomes a number or a string or boolean. This is done implicitly by the JavaScript interpreter. It looks strange and unbelievable for a C or Java programmer. A JavaScript programmer need not remember the list of data types (actually, they exist), JavaScript supports.

It looks nice and easy with loosely typed languages. But I feel it raises a lot of problems in coding. Observe the following.

Say in Java language:



|  |  |
| --- | --- |
| 1  2 | int cost = 10;               int cost = 20; |

It is compilation error as two times cost is initialized. You must declare only ones and can be reassigned number of times as follows.

|  |
| --- |
| int cost = 10;              cost = 20; |

Same thing in JavaScript or PERL is not error. Observe the code.



|  |  |
| --- | --- |
| 1  2 | var cost = 10;              var cost = 20; |

The earlier cost value is overridden with 20, but not error; earlier value is lost. This is a very dangerous situation which leads problems. If an article costs Rs.10 and the programmer declared correctly earlier. In the later part of the code, he forgot that he declared and redeclares with a different value (which may be wrong). For this reason, the object-based languages like JavaScript, PERL and Ruby etc. have limited scope in programming and are not used as full-fledged languages to develop software like banking, insurance etc.

A strongly typed language compiler enforces strict rules over the operations, what programmer can do, on data types and also passing parameters and return type to a method. An advantage of strongly typed language is it gives consistency over the results (by declaring the variables with data types). Specific operations are allowed on certain types. For example boolean cannot be used in addition, but an int can be used in addition. Boolean is used in control structures. This type of accidental wrong coding raises error by a strongly typed language.

## Backslash Java

**Backslash Java:**Like C/C++, Java also comes with a group of characters treated as a single character and this single character also represents an ASCII value. These characters are known as **escape sequences** and precede (starts) with a **backward slash \** like **\n** and **\t** etc. The name is so derived as they escape the normal flow or printing of characters. The **\n**escapes of printing in the same line and instead prints in a separate line.

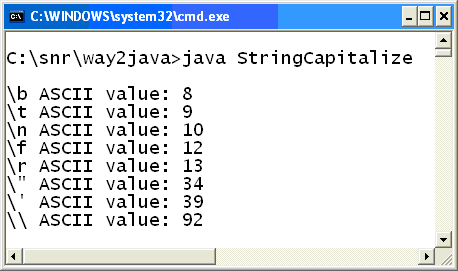
The following table prints the list of escape sequences supported by Java.

|  |  |  |
| --- | --- | --- |
| ESCAPE SEQUENCE | ASCII VALUE | MEANING |
| \b | 8 | backspace |
| \t | 9 | tab |
| \n | 10 | new line |
| \f | 12 | form feed |
| \r | 13 | carriage return |
| \" | 34 | double quote |
| \' | 39 | single quote |
| \\ | 92 | backslash |

Following program shows the way to print the ASCII values of escape sequences (the same procedure you followed in C/C++).



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | public class StringCapitalize  {    public static void main(String args[])    {      System.out.println("\\b ASCII value: " + (int)'\b');      System.out.println("\\t ASCII value: " + (int)'\t');      System.out.println("\\n ASCII value: " + (int)'\n');      System.out.println("\\f ASCII value: " + (int)'\f');      System.out.println("\\r ASCII value: " + (int)'\r');      System.out.println("\\\" ASCII value: " + (int)'\"');      System.out.println("\\' ASCII value: " + (int)'\'');      System.out.println("\\\\ ASCII value: " + (int)'\\');    }  } |

[](http://way2java.com/wp-content/uploads/2011/08/ss35.png)  
Observe the **println()** statements in the above code; all confusing with multiple backslashes. This is where the programmer should be careful with backslash menace.

#### Backslash Java Menace

*String str1 = “hello\sir”;  
System.out.println(“hello\sir”);*

The above two statements do not compile and compiler raises "**illegal escape character**" regarding **\s**. Any letter suffixed to**\** is treated as an **escape sequence** by the JVM. Now **\s** is checked with the standard list of escape sequences and did not find, the JVM raised an error. Suppose if you would like to have really **\s** in the code, instead of one **\** place two**\\**. **\\** is also an escape sequence and JVM places a single slash (instead of two slashes) in the output. The following works nice.

*String str1 = “hello\\sir”;  
System.out.println(“hello\\sir”);.*

###### A beginner should take care of the usage of Backslash Java.

In writing the regular expressions also, it must be taken care of.

*String str2 = str1.replaceAll(“\\d”, “OK”);*

The above statement is used in "Java String replaceAll" to replace each digit in the string with**OK**. The **\\d** is a **regular expression** where**d** represents each digit in the string. It is actually**\d** is required; but **\d**raises an error. Instead, **\\d** is placed which is converted to **\d** by the JVM and evaluated. This is where the programmer should be capable of.

Regular expressions are discussed, to some extent, in **way2java.com** in [JDK 1.4 (J2SE 4) Version](http://way2java.com/java-versions-2/jdk-1-4-j2se-4-version/). But regular expressions usage is a big subject which requires a good study. You are advised to refer a good Web site that primarily discusses regular expressions.

**Keywords:**

Every programming language comes with keywords.

**A keyword is just a word which has got a special meaning and purpose to the compiler**.

For this reason, we cannot use them in our program to identify our own (to give a name) a class, variable or method.

**Keywords Java:**

Some people interchangeably use the two words, **keyword** and **reserved word**. They think both are same; may be correct in many programming languages. But in Java, they are different. While Java is being developed, **the designers placed the two words, goto and const in reserved word list and not in keyword list**. Designers placed in reserved list for the reason thinking that **goto** and **cosnt** may be required in future versions of Java; any time they may be brought into keyword list. But as on today, Java proved that a programming language can exist without these two words.

## Java Keywords With Examples

1. **abstract**

abstract keyword is used to implement the abstraction in java. A method which doesn’t have method definition must be declared asabstract and the class containing it must be declared as abstract. You can’t instantiate abstract classes. Abstract methods must be implemented in the sub classes. You can’t use abstract keyword with variables and constructors.

|  |  |
| --- | --- |
|  | *abstract class AbstractClass{*  *abstract void abstractMethod();*  *}* |

1. **assert**

assert keyword is used in the assertion statements. These statements will enable you to test your assumptions about a program. Assertion statements provide the best way to detect and correct the programming errors. Assertion statements take one boolean expression as input and assumes that this will be always true. If the boolean expression returns false, AssertionError will be thrown.

|  |  |
| --- | --- |
|  | *public class MainClass{*  *public static void main(String[] args) {*  *System.out.println("Enter your marks");*  *Scanner sc = new Scanner(System.in);*  *int marks = sc.nextInt();*  *assert marks > 35 : "FAIL";*  *}*  *}* |

1. **boolean**

boolean keyword is used to define boolean type variables. boolean type variables can hold only two values – either true or false.

*boolean isActive = true;*

|  |  |
| --- | --- |
|  |  |

1. **break**

The break keyword is used to stop the execution of a loop(for, while, switch-case) based on some condition.

|  |  |
| --- | --- |
|  | *public class MainClass{*  *public static void main(String[] args)    {*  *for (int i = 0; i < 100; i++)        {*  *System.out.println(i);*  *if(i == 50)            {*  *break;*  *}*  *}*  *}*  *}* |

1. **byte**

byte keyword is used to declare byte type of variables. A byte variable can hold a numeric value in the range from -128 to 127.

|  |  |
| --- | --- |
|  | *byte b = 50;* |

1. **switch**
2. **case**

Both switch and case keywords are used in the switch-case statement.

|  |  |
| --- | --- |
|  | public class MainClass{      public static void main(String[] args)     {          Scanner sc = new Scanner(System.in);           System.out.println("Enter Day :");            int day = sc.nextInt();          switch (day){              case 1 : System.out.println("SUNDAY");              break;                case 2 : System.out.println("MONDAY");              break;                case 3 : System.out.println("TUESDAY");              break;                case 4 : System.out.println("WEDNESDAY");              break;                case 5 : System.out.println("THURSDAY");              break;                case 6 : System.out.println("FRIDAY");              break;                case 7 : System.out.println("SATURDAY");              break;                default: System.out.println("Invalid");              break;          }      }  } |

1. **try**
2. **catch**
3. **finally**

try, catch and finally keywords are used to handle the exceptions in java. The statements which are to be monitored for exceptions are kept in the try block. The exceptions thrown by the try block are caught in the catch block.

finally block is always executed.

|  |  |
| --- | --- |
|  | *public class MainClass{*  *public static void main(String[] args) {*  *try{*  *int i = Integer.parseInt("abc");*  *}*  *catch(NumberFormatException ex){*  *System.out.println(ex);*  *}*  *finally{*  *System.out.println("This will be always executed");*  *}*  *}*  *}* |

1. **Char**

char keyword is used to declare primitive char type variables. char represents the characters in java.

|  |  |
| --- | --- |
|  | *char a = 'A';*  *char b = 'B';*  *char c = 'C';* |

1. **class**

class keyword is used to define the classes in java.

|  |  |
| --- | --- |
|  | *class MyClass{*  *class MyInnerClass{*  *//Inner Class*  *}*  *}* |

1. **continue**

continue keyword is used to stop the execution of current iteration and start the execution of next iteration in a loop.

|  |  |
| --- | --- |
|  | *public class MainClass{*  *public static void main(String[] args){*  *for (int i = 0; i <= 100; i++){*  *if(i % 5 != 0){*  *continue;*  *}*  *System.out.println(i);*  *}*  *}*  *}* |

1. **default**

default keyword is used to define the default methods in an interface (From Java 8). default keyword is also used in the switch-casestatements.

|  |  |
| --- | --- |
|  | *interface MyInterface{*  *public default void myDefaultMethod(){*  *System.out.println("Default Method");*  *}*  *}* |

1. **do**

do keyword is used in a do–while loop. do-while loop is used to execute one or more statements repetitively until a condition returns false.

|  |  |
| --- | --- |
|  | *public class MainClass{*  *public static void main(String[] args){*  *int a = 10;*  *int b = 20;*  *do{*  *a = a + b;*  *b = b + 10;*    *System.out.println("a = "+a);*  *System.out.println("b = "+b);*    *} while (a <= 100);*  *}*  *}* |

1. **double**

double keyword is used to declare primitive double type of variables.

|  |  |
| --- | --- |
|  | *public class MainClass{*  *public static void main(String[] args){*  *double d1 = 23.56;*  *double d2 = 56.23;*  *double d3 = d1 + d2;*  *System.out.println(d3);*  *}*  *}* |

1. **if**
2. **else**

if and else keywords are used in if-else block.

|  |  |
| --- | --- |
|  | *public* *class* *MainClass{*  *public* *static* *void* *main(String[] args){*  *Scanner sc = new* *Scanner(System.in);*   *System.out.println("Enter a string :");*   *String input = sc.next();*  *if(input.equalsIgnoreCase("JAVA")) {*  *System.out.println("It's JAVA");*  *}*  *else{*  *System.out.println("It's not JAVA");*  *}*  *}*  *}* |

1. **enum**

enum keyword is used to define enum types.

|  |  |
| --- | --- |
|  | *enum* *MyEnums{*  *A, B, C, D;*  *}* |

1. **extends**

extends keyword is used in inheritance. It is used when a class extends another class.

|  |  |
| --- | --- |
|  | *class SuperClass{*  *//Super Class*  *}*  *class SubClass extends SuperClass{*  *//Sub Class*  *}* |

1. **final**

final keyword is used when a class or a method or a field doesn’t need further modifications. final class can’t be extended, final methodcan’t be overridden and the value of a final field can’t be changed. [[***See more***](http://javaconceptoftheday.com/final-keyword-in-java/)]

|  |  |
| --- | --- |
|  | *final* *class* *FinalClass{*  *final* *int* *finalVariable = 10;*  *final* *void* *finalMethod()*  *{*  *//final method*  *}*  *}* |

1. **float**

float keyword indicates primitive float type of variables.

|  |  |
| --- | --- |
|  | *public class MainClass{*  *public static void main(String[] args)     {*  *float f1 = 45.26f;*  *float f2 = 84.25f;*  *float f3 = f2 - f1;*  *System.out.println(f3);*  *}*  *}* |

1. **for**

for loop is used to execute the set of statements until a condition is true.

|  |  |
| --- | --- |
|  | *public class MainClass{*  *public static void main(String[] args) {*  *for (int i = 0; i <= 10; i++){*  *System.out.println(i);*  *}*  *}*  *}* |

1. **implements**
2. implements keyword is used while implementing an interface.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | interface MyInterface  {      void myMethod();  }    class MyClass implements MyInterface  {      public void myMethod()      {          System.out.println("My Method");      }  } |

1. **25) import**
2. import keyword is used to import the members of a particular package into current java file. [[***See more***](http://javaconceptoftheday.com/static-import-java/)]
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3 | import java.sql.\*;  import java.util.Arrays;  import java.util.Scanner; |

1. **26) instanceOf**
2. instanceOf is used to check whether an object is of specified type. The syntax for using instanceOf keyword is “**Object\_ReferenceinstanceOf Type**“.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | class A  {    }    public class MainClass  {      public static void main(String[] args)      {          A a = new A();            if(a instanceof A)          {              System.out.println("a is of type A");          }      }  } |

1. **27) int**
2. int keyword is used to declare primitive integer type of variables.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | public class MainClass  {      public static void main(String[] args)      {          int i1 = 10;            int i2 = 20;            int i3 = i1 \*  i2;            System.out.println(i3);      }  } |

1. **28) interface**
2. interface keyword is used to define the interfaces in java.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4 | interface MyInterface  {      void myMethod();  } |

1. **29) long**
2. long is used to define the primitive long type variables.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | public class MainClass  {      public static void main(String[] args)      {          long l1 = 101;            long l2 = 202;            long l3 = l1 +  l2;            System.out.println(l3);      }  } |

1. **30) native**
2. native keyword is used with a method to indicate that a particular method is implemented in native code using Java Native Interfaces(JNI).
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4 | class AnyClass  {      public native void anyMethod(int i, double d);  } |

1. **31) new**
2. new keyword is used while creating the instances of a class.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | class A  {    }    public class MainClass  {      public static void main(String[] args)      {          A a = new A();      }  } |

1. **32) package**
2. package keyword is used to specify a package to which the current file belongs to.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | package pack1;    class A  {    } |

1. **33) private**
2. private keyword is used to declare a member of a class as private. private methods and fields are visible within the class in which they are defined.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | class A  {      private int i = 111;   //private field        private void method()      {          //private method      }  } |

1. **34) protected**
2. protected keyword is used to declare a member of a class as protected. protected members of a class are visible within the package only, but they can be inherited to any sub classes.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | class A  {      protected int i = 111;   //protected field        protected void method()      {          //protected method      }  } |

1. **35) public**
2. public keyword is used to declare the members of a class or class itself as public. public members of a class are visible from anywhere and they can be inherited to any sub classes.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | public class A  {      public int i = 222;   //public field        public A()      {          //public constructor      }        public void method()      {          //public method      }  } |

1. **36) return**
2. return keyword is used to return the control back to the caller from the method.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | class A  {      int method(int i)      {          return i\*i;     //method returning a value      }  } |

1. **37) short**
2. short keyword is used to declare primitive short type variables.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3 | short s1 = 11;    short s2 = 22; |

1. **38) static**
2. ***static*** keyword is used to define the class level members of a class. static members of a class are stored in the class memory and you can access them directly through class name. No need to instantiate a class. [[See more](http://javaconceptoftheday.com/static-members-java/)]
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | class A  {      static int staticField = 555;    //Static Field        static void staticMethod()      {          //Static method      }  }    public class MainClass  {      public static void main(String[] args)      {          System.out.println(A.staticField);    //Accessing staticField via class name            A.staticMethod();     //Accessing staticMethod via class name      }  } |

1. **39) strictfp**
2. strictfp keyword is used to implement the strict precision of floating point calculations on different platforms. strictfp can be used with classes, interfaces and methods.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | strictfp interface I  {      //strictfp applied on interface  }    strictfp class C  {      //strictfp applied on class  }    class A  {      strictfp void method()      {          //strictfp applied on method      }  } |

1. **40) super**
2. super keyword is used to access super class members inside a sub class.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | class A  {      int i;        public A(int i)      {          this.i = i;      }        void methodA()      {          System.out.println(i);      }  }    class B extends A  {      public B()      {          super(10);    //Calling super class constructor      }        void methodB()      {          System.out.println(super.i);    //accessing super class field            super.methodA();    //Calling super class method      }  } |

1. **41) synchronized**
2. synchronized keyword is used to implement the synchronization in java. only one thread can enter into a method or a block which is declared as synchronized. Any thread which wants to enter synchronized method or block must acquire object lock of those methods or blocks. [[***See more***](http://javaconceptoftheday.com/synchronization-in-java/)]
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | class AnyClass  {      synchronized void synchronizedMethod()      {          //Synchronized method      }        void anyMethod()      {          synchronized (this)          {              //Synchronized block          }      }  } |

1. **42) this**
2. this keyword is used to access other members of the same class.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | class AnyClass  {      int i;        AnyClass()      {          System.out.println("First Constructor");      }        AnyClass(int j)      {          this();    //calling statement to First Constructor          System.out.println("Second Constructor");      }        void methodOne()      {          System.out.println("From method one");      }        void methodTwo()      {          System.out.println(this.i);  //Accessing same class field          this.methodOne();      //Accessing same class method      }  } |

1. **43) throw**
2. throw keyword is used to throw the exceptions manually. [[***See more***](http://javaconceptoftheday.com/throwing-rethrowing-exception-java/)]
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | public class MainClass  {      public static void main(String[] args)      {          try          {              //throwing NumberFormatException manually                throw new NumberFormatException();          }          catch(Exception ex)          {              System.out.println(ex);          }      }  } |

1. **44) throws**
2. throws keyword is used to specify the exceptions which the current method may throw. [[***See more***](http://javaconceptoftheday.com/throws-keyword-java/)]
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | class A  {      void method() throws NumberFormatException      {          int i = Integer.parseInt("abc");      }  } |

1. **45) transient**
2. transient keyword is used in serialization. A variable which is declared as transient will not be eligible for serialization.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | class MyClass implements Serializable  {      int a;        transient String s;   //This will not be serialized        double d;  } |

1. **46) void**
2. void keyword is used to indicate that method returns nothing.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | class A  {      void methodReturnsNothing()      {          //Method returns no value      }  } |

1. **47) volatile**
2. volatile keyword is used in the concurrent programming. The value of a variable which is declared as volatile will be written into or read from the main memory.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

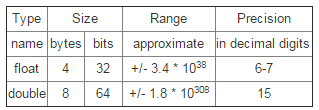
|  |  |
| --- | --- |
| 1  2  3  4 | class A  {      public volatile int counter = 0;  } |

1. **48) while**
2. while keyword is used in the while loop.
3. [?](http://javaconceptoftheday.com/java-keywords-with-examples/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | public class MainClass  {      public static void main(String[] args)      {          int i = 10;            while (i <= 100)          {              System.out.println(i);                i = i + 10;          }      }  } |

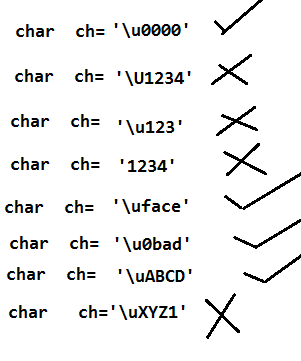
1. **49) goto        50) const**
2. Both goto and const are reserved words in java but they are currently not used.
3. **Note** : true, false and null are not the keywords. They are literals in java.

Floating point literal



**character literal can also be created using UNICODE format**

syntax:

 '\uxxxx' where xxxx can be any 4 digit hexa decimal number (0-9,a-f/A-F)

**Java coding conventions**

In Java For all predefined classes , methods, variables,...... java people already following some coding conventions and recommending us to follow same coding conventions for our own classes , methods, variables,......

**Following are different java coding conventions**

**1. coding convention for a class**

A class name can contain any number of words but every word first letter should be in Capital letter.

Eg:

String, StringBuffer, InputStreamReader, VarDemo,Student,...

**2. coding convention for interfaces**

An interface name can contain any number of words but everyword first letter should be in capital.

Eg:

Clonable, Runnable, Serializable, ActionListener, MyInterface,......

**3. coding convention for methods**

A method name can contain any number of words where first word all letters should be in lowercase and second word onwards every word first letter should be in capital.

Eg:

main(),println(),lastIndexOf(),getAgeOfPerson(),getName(), displayData(),....

**4. coding convention for a variables**

A variable name can contain any number of words where first word all letters should be in lowercase and second word onwards everyword first letter should be in capital.

Eg:

length,age,endOfTheYear,toDay,.....

**5. coding convention for constants**

A constant name contains all letters in capital and if we have multiple words in constant name then they should separated by using underscore.

Eg:

PI,MAX\_VALUE,MIN\_VALUE,EXIT\_ON\_CLOSE,MY\_CONSTANT...

**6. coding convention for packages**

A package name contains all letters in lowercase we may or may not contain multiple words.

Eg:

java,lang,io,awt,mypackage,......