**Java Modifiers, Variables and Operators**  
   
i) Modifiers in Java  
  
ii) Java Variables  
  
iii) Java operators  
-----------------------------  
**i) Modifiers in Java**  
Modifiers are keywords that we add to those definitions to change their meaning.  
  
a) Access Modifiers  
  
b) Non-Access Modifiers  
--------------------------  
a) Access Modifiers  
We use access modifiers to define access control for classes, methods and variables.  
  
Four Access Modifiers  
  
**i) private**  
The private access modifier is accessible only within class.  
  
Ex:  
  
private int a =100;  
  
**2) default**  
If we don't specify any modifier then it is treated as default, this can be accessible only within package.  
  
class Sample{  
.  
.  
}  
  
**3) protected**  
The protected access modifier is accessible within package, outside of the package but through Inheritance only.  
  
protected class Sample{  
.  
.  
}  
  
**4) public**  
public access modifier is accessible everywhere.  
  
public class Sample {  
.  
.  
}  
---------------------------------------------------

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modifier | Within Class | Within Package | Outside of the Package  (By Sub Class/inheritance) | Outside of the Package |
| private | Y | N | N | N |
| default | Y | Y | N | N |
| protected | Y | Y | Y | N |
| public | Y | Y | Y | Y |

b) Non Access Modifiers  
**1) static**  
static modifier is used create classes, methods and variables.  
  
Ex:  
  
static int a =10;  
  
static void int add(){  
.  
.  
}

public class Hello

{

// value / method

public static String staticValue;

public String nonStaticValue;

}

class A

{

Hello hello = new Hello();

hello.staticValue = "abc";

hello.nonStaticValue = "xyz";

}

class B

{

Hello hello2 = new Hello(); // here staticValue = "abc"

hello2.staticValue; // will have value of "abc"

hello2.nonStaticValue; // will have value

}

If you declare any variable as static, it is known static variable.

* The static variable can be used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees,college name of students etc.
* The static variable gets memory only once in class area at the time of class loading.

Advantage of static variable

It makes your program **memory efficient** (i.e it saves memory).

**2) final**  
final modifier for finalizing of classes, methods and variables.  
  
Ex:  
final int a =100;  
.  
.  
.  
a=200; //Error  
---------------  
int a =100;  
.  
.  
.  
.  
a =200;   
------------------  
**3) abstract**  
abstract modifier is to create abstract classes, abstract methods  
  
ex:  
  
abstract class Sample{  
.  
.  
}

**Additional Info:**

Final Non Access Modifiers

Final modifiers are applicable to :

1. Class
2. Method
3. Instance Variable
4. Local Variable
5. Method arguments

Final Class :

A Class when set to final *cannot* be **extended** by any other Class.

**Example: A String**Class in java.lang package

Final Method :

A Method when set to final *cannot* be **overridden** by any subclass.

Final Variable :

When a variable is set to final, its value *cannot* be **changed**. Final variables are like constants.

**Example :**public static final int i = 10;

Abstract Non Access Modifier

Keyword: abstract

Abstract modifiers are applicable to:

1. Class
2. Method

Abstract Class:

An abstract Class can have abstract Methods. A Class can also be an abstract class without having any abstract Methods in it. If a Class has an abstract Method , the Class becomes an abstract Class.

Abstract Method :

Abstract Methods are those Methods which does not have a body but only a signature.

**Example :** public abstract void method();

Synchronized Non Access Modifier

Synchronized modifiers are applicable to

1. Method

Synchronized Method

Synchronized Methods can be accessed by only one thread at a time.

Native Non Access Modifier

Native modifiers are applicable to

1. Method

Native Method

Naive Methods indicate that the method is implemented on a platform dependent code.

Strictfp Non Access Modifier

Strictfp modifiers are applicable to

1. Class
2. Method

Strictfp Class / Method

Strictfp non access modifier forces floating point or floating point operation to adhere to IEEE 754 standard.

**Note\*: Strictfp**non access modifier *cannot* be applied on a variable.

-------------------------------  
**ii) Java Variables**  
1) What is Variable?  
A named memory location to store the temporary data within a program.  
  
Two types of memories in Computer environment  
  
a) Primary memory (RAM)  
  
b) Secondary memory (HDD, DVD, USB drive etc...)  
----------------------------  
2) Declaration of Variables  
Java supports Explicit declaration of Variables.  
  
Syntax and Examples:  
  
dataType variableName;  
  
int a;  
-------------  
dataType variablename=value;  
  
int b=20;  
---------------  
dataType variable1, Variable2, variable3;  
  
int a, b, c;  
-----------------  
dataType variable1=value; variable2=value; varible3=value;  
  
int a=10; b=20; c=30;  
------------------------------------  
3) Assign values to variables   
a) Initialization  
  
b) Reading  
  
Ex:  
  
int a=100; //Initialization  
  
int a=10;  
int b;  
b=a; //Reading  
---------------------------------  
4) Variable Naming Restrictions  
> Java variables are case sensitive,  
  
> Java variable name should start with a letter or $ or \_  
  
Ex:  
  
myvar(Correct)  
MYVAR  
$myvar  
\_myvar  
myvar\_1  
--------------  
1myvar(Incorrect)  
\*myvar  
----------------  
> Variable names should not match with Java keywords/Reserved words.  
  
> Must be unique in the scope of declaration.  
  
> Variable names Must not exceed 255 characters.  
-------------------------------------------  
5) Types of Variables  
**Three types of variables in Java**  
**a) Local variable**(Local variable is declared in methods or blocks.)  
  
**b) Instance variable**(Instance variables are declared in a class but outside of a method or any block)  
  
**c) Class/Static variable**A Variable that is declared as static, It cannot be local.  
----------------------------------  
Example:  
  
package xyza;  
  
public class VariablesExample {  
//a Variable is a Class/Static variable  
static int a =100;  
  
//mysalary variable is a Local variable.  
public int salary(){  
    int mysalary =10000+2000+1500;  
    mysalary=mysalary + a;  
    return mysalary;  
}  
  
public static void main (String[]args){  
//Instance variable  
int b =200;  
System.out.println(a);//100  
System.out.println(b); //200      
          
VariablesExample obj= new VariablesExample();      
System.out.println(obj.salary());   
// i is a Local Variable      
for (int i=1; i<=5; i++){  
    System.out.println(i);  
    System.out.println(a);  
    System.out.println(b);  
}  
}  
}  
----------------------------------  
**iii) Java Operators**  
Important Categories of Operators  
  
a) Arithmetic Operators  
  
b) Relational Operators  
  
c) Assignment Operators  
  
d) Logical Operators  
-------------------------------  
a) Arithmetic Operators  
1) Addition + (for Addition, String concatenation)  
  
2) Subtraction - (for Subtraction, Negation)  
  
3) Multiplication \*  
  
4) Division /  
  
5) Modules %  
  
6) Increment ++  
  
7) Decrement --  
---------------------------  
Example:  
  
public class OperatorsExample {  
    public static void main (String [] args){  
        int a =10, b=5;  
        String c ="Selenium", d= "Testing";  
  
System.out.println("Addition of a, b is: "+ (a+b));//Addition of a, b is: 15  
System.out.println("Subtraction of a, b is: "+ (a-b));          
System.out.println("Multiplication of a, b is: "+ (a\*b));      
System.out.println("Division of a, b is: "+ (a/b));      
System.out.println("Modules of a, b is: "+ (a%b));  
  
b=10;  
a = ++b;  
System.out.println(a);//11  
  
b=10;  
a = --b;  
System.out.println(a);//9  
}  
}  
---------------------------------------  
b) Relational Operators  
1) ==  
  
2) !=  
  
3) >  
  
4) >=  
  
5) <  
  
6) <=  
----------------------------------  
Note: Relational Operators return Boolean / Logical result  
  
Example:  
  
public class OperatorsExample {  
    public static void main (String [] args){  
        int a =10, b=20;  
        System.out.println((a>b));//false  
        System.out.println((a>=b));//false  
        System.out.println((a==b));//false  
          
        System.out.println((a<b));//true  
        System.out.println((a<=b));//true  
        System.out.println((a!=b));//true  
}  
}  
------------------------------------------------  
d) Logical Operators  
1) Logical Not Operator  !  
  
2) Logical And Operator &&  
  
3) Logical Or Operators ||  
  
Result Criteria  
  
Not operator  
--------------  
Operand1    Operand2    Result  
--------------------------------------  
true               true         false  
true               false        true  
false              true         true  
false              false        true  
--------------------------------------  
And operator  
--------------  
Operand1    Operand2    Result  
--------------------------------------  
true             true            true  
true             false           false  
false            true            false  
false            false           false  
--------------------------------------  
Or Operator  
  
Operand1    Operand2    Result  
--------------------------------------  
true             true             true  
true             false            true  
false            true             true  
false            false            false  
--------------------------------------  
Example:  
  
public class OperatorsExample {  
    public static void main (String [] args){  
        boolean a =true, b=false;  
        System.out.println(!(a && b));//true  
        System.out.println((a && b));//false  
        System.out.println((a || b));//true  
    }          
}  
--------------------------------------  
public class OperatorsExample {  
    public static void main (String [] args){  
        int a =1000, b=500, c=7000;  
  
if ((a>b) && (a>c)){  
    System.out.println("A is a Big Number");  
}  
else{  
    System.out.println("A is Not a Big Number");      
}  
}          
}  
--------------------------------------  
c) Assignment Operators  
1) Assignment Operator  
  
=  
  
a=10;  
  
2) Add and Assign +=  
  
3) Subtract and assign  
  
4) Multiple and assign  
-------------------------------  
Example:  
public class OperatorsExample {  
    public static void main (String [] args){  
        int a =10;  
  
System.out.println(a);//10  
a+=10;  
System.out.println(a);//20  
  
a-=10;  
System.out.println(a);//10  
  
a\*=10;  
System.out.println(a);//100  
}          
}  
-----------------------------------------------------------  
Bitwise Operators  
> Java defines several bitwise operators, which can be applied to the integer types, Bitwise operator works  
   on bits and performs bit-by-bit operation.  
  
i) The bitwise & operator performs a bitwise AND operation.  
  
ii) The bitwise ^ operator performs a bitwise exclusive OR operation.  
  
iii) The bitwise | operator performs a bitwise XOR operation.