Variables in Java

A variable is a name which is associated with a value that can be changed. For example when I write `int i =10;` here variable name is i which is associated with value 10, int is a data type that represents that this variable can hold integer values.

How to Declare a Variable in Java

To declare a variable follow this syntax:
data_type variable_name = value;
here value is optional because in java, you can declare the variable first and then later assign the value to it.
For example: Here num is a variable and int is a data type.
int num;
Similarly we can assign the values to the variables while declaring them, like this:
char ch = 'A';
int number = 100;
or we can do it like this:
char ch;
int number;
...
ch = 'A';
number = 100;

Variables naming convention in java

1) Variables naming cannot contain white spaces, for example: int num ber = 100; is invalid because the variable name has space in it.
2) Variable name cannot begin with special characters such as $ and _
3) As per the java coding standards the variable name should begin with a lower case letter, for example int number; For lengthy variables names that has more than one words do it like this: int smallNumber; int bigNumber; (start the second word with capital letter).
4) Variable names are case sensitive in Java.

Types of Variables in Java

There are three types of variables in Java- Local variable , Static (or class) variable and Instance variable.

Static (or class) Variable - Static variables are also known as class variable because they are associated with the class and common for all the instances of class. For example, If I create three objects of a class and access this static variable, it would be common for all, the changes made to the variable using one of the object would reflect when you access it through other objects.

Example of static variable

```
public class Ex3 {
  public static String str="BCA 5th Semester Students ";

  public static void main(String args[]){
    Ex3 obj1 = new Ex3();
    Ex3 obj2 = new Ex3();
    Ex3 obj3 = new Ex3();

    //All three will display "class or static variable"
    System.out.println(obj1.str);
    System.out.println(obj2.str);
    System.out.println(obj3.str);

    //changing the value of static variable using obj2
    obj3.str = "You are now in 6th semester ";

    System.out.println(obj1.str);
    System.out.println(obj2.str);
    System.out.println(obj3.str);

  }
}
```

Output:
BCA 5th Semester Students
BCA 5th Semester Students
BCA 5th Semester Students
You are now in 6th semester
You are now in 6th semester
You are now in 6th semester

As you can see all three statements displayed the same output irrespective of the instance through which it is being accessed. That's is why we can access the static variables without using the objects like this:
System.out.println(myClassVar);
Note- Only static variables can be accessed like this. This doesn't apply for instance and local variables.

Instance variable - Each instance(objects) of class has its own copy of instance variable. Unlike static variable, instance variables have their own separate copy of instance variable. We have changed the instance variable value using object obj2 in the following program and when we displayed the variable using all three objects, only the obj2 value got changed, others remain unchanged. This shows that they have their own copy of instance variable.

Example of Instance variable

```
public class InstanceVarExample {
  String myInstanceVar="instance variable";

  public static void main(String args[]){
        InstanceVarExample obj1 = new InstanceVarExample();
        InstanceVarExample obj2 = new InstanceVarExample();
        InstanceVarExample obj3 = new InstanceVarExample();

        System.out.println(obj1.myInstanceVar);
        System.out.println(obj2.myInstanceVar);
        System.out.println(obj3.myInstanceVar);


        obj2.myInstanceVar = "Changed Text";
        System.out.println(obj1.myInstanceVar);
        System.out.println(obj2.myInstanceVar);
        System.out.println(obj3.myInstanceVar);
  }
}
```

Output:

```
instance variable
instance variable
instance variable
instance variable
Changed Text
instance variable
```

Local Variable - These variables are declared inside method of the class. Their scope is limited to the method which means that You can't change their values and access them outside of the method.

In this example, I have declared the instance variable with the same name as local variable, this is to demonstrate the scope of local variables.

Example of Local variable

```
public class VariableExample {
  // instance variable
  public String myVar="instance variable";

  public void myMethod(){
        // local variable
        String myVar = "Inside Method";
        System.out.println(myVar);
  }
  public static void main(String args[]){
    // Creating object
    VariableExample obj = new VariableExample();
    System.out.println("Calling Method");
    obj.myMethod();
    System.out.println(obj.myVar);
  }
```

```
}
```
Output:
Calling Method
Inside Method
instance variable

If I hadn’t declared the instance variable and only declared the local variable inside method then the statement System.out.println(obj.myVar); would have thrown compilation error. As you cannot change and access local variables outside the method.