Variables in JAVA

#What is a Variable?

When we want to store any information, we store it in an address of the computer. Instead of remembering the complex address where we have stored our information, we name that address. The naming of an address is known as variable. Variable is the name of memory location.

In other words, variable is a name which is used to store a value of any type during program execution.

To declare the variable in Java, we can use following syntax:

datatype variableName;

Here, datatype refers to type of variable which can any like: int, float etc. and variableName can be any like: empId, amount, price etc.

Java Programming language defines mainly three kinds of variables.

1. Instance Variables

2. Static Variables (Class Variables)

3. Local Variables

1. **Instance variables in Java**

Instance variables are variables that are declared inside a class but outside any method, constructor or block. Instance variable are also variable of object. They are referred as object variable. Each object has it's own copy of each variable and thus, it doesn't affect the instance variable if any object changes the value of the variable.

```
class Student

{

   String name;

   int age;

}
```

Here name and age are instance variable of Student class.

2. **Static variables in Java**

Static are class variables declared with static keyword. Static variables are initialized only once. Static variables are also used in declaring constant along with final keyword.

```
class Student

{

   String name;

   int age;

   static int instituteCode=1101;

}
```

Here instituteCode is a static variable. Each object of Student class will share instituteCode property.

Additional points on static variable:

1. static variables are also known as class variable.

2. static means to remain constant.

3. In Java, it means that it will be constant for all the instances created for that class.

4. static variable need not be called from object.

5. It is called by classname.static_variable_name

Note: A static variable can never be defined inside a method i.e it can never be a local variable.

Example:

Suppose you make 2 objects of class Student and you change the value of static variable from one object. Now when you print it from other object, it will display the changed value. This is because it was declared static i.e it is constant for every object created.

```
package study;

class Student{

   int a;

   static int id = 35;
   void change()
 {
      System.out.println(id);
   }
}
class Study{
   public static void main(String[] args) {
      Student o1 = new Student();
      Student o2 = new Student();
      o1.change();
      Student.id = 1;
      o2.change();
   }
}
```

3. **Local variables in Java**

Local variables are declared in method, constructor or block. Local variables are initialized when method, constructor or block start and will be destroyed once its end. Local variable resides in stack. Access modifiers are not used for local variable.

```
float getDiscount(int price)
{
 float discount;
 discount=price*(20/100);
 return discount;
}
```

Here discount is a local variable.

# Scope of variables in Java

Scope of a variable decides it's accessibility throughout the program. As we have seen variables are of different types so they have their own scope.

Local variable: Scope of local variable is limited to the block in which it is declared. For example, a variable is declared inside a function will be accessible only within this function.

Instance variable: scope of instance variable depends on the access-modifiers (public, private, default). If variable is declared as private then it is accessible within class only.

If variable is declared as public then it is accessible for all and throughout the application.

If variable is declared as default then it is accessible within the same package.

Example:

In this example, we have created two variables a and i, first is declared inside the function and second is declared inside for loop. Both variables have their own scope in which they are declared, so accessing outside the block, reports an error.

```
class HelloWorld {
   public static void main (String[] args) {
      int a = 10;
      for(int i = 0; i<5; i++) {
         System.out.println(i);
      }
      System.out.println("a = "+a);
      System.out.println("i = "+i); // error
   }
}
```

Data Types in JAVA

Java language has a rich implementation of data types. Data types specify size and the type of values that can be stored in an identifier.

In java, data types are classified into two categories:

Primitive Data type

Non-Primitive Data type

1) Primitive Data type

A primitive data type can be of eight types:

byte, short, int, long, float, double, char, boolean

Once a primitive data type has been declared it’s type can never be changed, although in most cases it’s value can be changed. These eight primitive types can be put into four groups,

Integer

This group includes byte, short, int, long

byte : It is 1 byte(8-bits) integer data type. Value range from -128 to 127. Default value zero. example: byte b=10;

short : It is 2 bytes(16-bits) integer data type. Value range from -32768 to 32767. Default value zero. example: short s=11;

int : It is 4 bytes(32-bits) integer data type. Value range from -2147483648 to 2147483647. Default value zero. example: int i=10;

long : It is 8 bytes(64-bits) integer data type. Value range from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Default value zero. example: long l=100012;

Example:

Lets create an example in which we work with integer type data and we can get idea how to use datatype in the java program.

```
class Demo
{
public static void main(String[] args) {
        // byte type
        byte b = 20;
        System.out.println("b= "+b);
        // short type
        short s = 20;
        System.out.println("s= "+s);
        // int type
        int i = 20;
        System.out.println("i= "+i);
        // long type
        long l = 20;
        System.out.println("l= "+l);
  }
}
```

Floating-Point Number

This group includes float, double

float : It is 4 bytes(32-bits) float data type. Default value 0.0f. example: float ff=10.3f;

double : It is 8 bytes(64-bits) float data type. Default value 0.0d. example: double db=11.123;

Example:

In this example, we used floating point type and declared variables to hold floating values. Floating type is useful to store decimal point values.

```
class Demo {
   public static void main(String[] args) {
         // float type
         float f = 20.25f;
         System.out.println("f= "+f);
         // double type
         double d = 20.25;
         System.out.println("d= "+d);
   }
}
```

Characters

This group represents char, which represent symbols in a character set, like letters and numbers.

char : It is 2 bytes(16-bits) unsigned unicode character. Range 0 to 65,535. example: char c='a';

Example

Char type in Java uses 2 bytes to unicode characters. Since it works with unicode then we can store alphabet character, currency character or other characters that are comes under the unicode set.

```
class Demo {
 public static void main (String[] args) {
       char ch = 'S';
       System.out.println(ch);
       char ch2 = '&';
       System.out.println(ch2);
       char ch3 = '$';
       System.out.println(ch3);
   }
}
```

Boolean

This group represent boolean, which is a special type for representing true/false values. They are defined constant of the language. example: boolean b=true;

Example

Boolean type in Java works with two values only either true or false. It is mostly use in conditional expressions to perform conditional based programming.

```
class Demo

{

 public static void main(String[] args)

  {

       boolean t = true;

       System.out.println(t);

       boolean f = false;

       System.out.println(f);

   }

}
```

2) Non-Primitive (Reference) Data type

A reference data type is used to refer to an object. A reference variable is declared to be of specific and that type can never be change.

For example: String str, here str is a reference variable of type String. String is a class in Java.

Reference type are used to hold reference of an object. Object can be instance of any class or entity.

In object-oriented system, we deal with object that stores properties. To refer those objects, we use reference types.

Identifiers in Java

All Java components require names. Name used for classes, methods, interfaces and variables are called Identifier. Identifier must follow some rules. Here are the rules:

All identifiers must start with either a letter(a to z or A to Z) or currency character($) or an underscore.

After the first character, an identifier can have any combination of characters.

Java keywords cannot be used as an identifier.

Identifiers in Java are case sensitive, for and For are two different identifiers.

Some valid identifiers are: int a, class Car, float amount etc.

Keywords in JAVA

Java keywords are also known as reserved words. Keywords are particular words that act as a key to a code. These are predefined words by Java so they cannot be used as a variable or object name or class name.

List of Java Keywords

A list of Java keywords or reserved words are given below:

1. abstract: Java abstract keyword is used to declare an abstract class. An abstract class can provide the implementation of the interface. It can have abstract and non-abstract methods.

2. boolean: Java boolean keyword is used to declare a variable as a boolean type. It can hold True and False values only.

3. break: Java break keyword is used to break the loop or switch statement. It breaks the current flow of the program at specified conditions.

4. byte: Java byte keyword is used to declare a variable that can hold 8-bit data values.

5. case: Java case keyword is used with the switch statements to mark blocks of text.

6. catch: Java catch keyword is used to catch the exceptions generated by try statements. It must be used after the try block only.

7. char: Java char keyword is used to declare a variable that can hold unsigned 16-bit Unicode characters

8. class: Java class keyword is used to declare a class.

9. continue: Java continue keyword is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.

10. default: Java default keyword is used to specify the default block of code in a switch statement.

11. do: Java do keyword is used in the control statement to declare a loop. It can iterate a part of the program several times.

12. double: Java double keyword is used to declare a variable that can hold 64-bit floating-point number.

13. else: Java else keyword is used to indicate the alternative branches in an if statement.

14. enum: Java enum keyword is used to define a fixed set of constants. Enum constructors are always private or default.

15. extends: Java extends keyword is used to indicate that a class is derived from another class or interface.

16. final: Java final keyword is used to indicate that a variable holds a constant value. It is used with a variable. It is used to restrict the user from updating the value of the variable.

17. finally: Java finally keyword indicates a block of code in a try-catch structure. This block is always executed whether an exception is handled or not.

18. float: Java float keyword is used to declare a variable that can hold a 32-bit floating-point number.

19. for: Java for keyword is used to start a for loop. It is used to execute a set of instructions/functions repeatedly when some condition becomes true. If the number of iterations is fixed, it is recommended to use for loop.

20. if: Java if keyword tests the condition. It executes the if block if the condition is true.

21. implements: Java implements keyword is used to implement an interface.

22. import: Java import keyword makes classes and interfaces available and accessible to the current source code.

23. instanceof: Java instanceof keyword is used to test whether the object is an instance of the specified class or implements an interface.

24. int: Java int keyword is used to declare a variable that can hold a 32-bit signed integer.

25. interface: Java interface keyword is used to declare an interface. It can have only abstract methods.

26. long: Java long keyword is used to declare a variable that can hold a 64-bit integer.

27. native: Java native keyword is used to specify that a method is implemented in native code using JNI (Java Native Interface).

28. new: Java new keyword is used to create new objects.

29. null: Java null keyword is used to indicate that a reference does not refer to anything. It removes the garbage value.

30. package: Java package keyword is used to declare a Java package that includes the classes.

31. private: Java private keyword is an access modifier. It is used to indicate that a method or variable may be accessed only in the class in which it is declared.

32. protected: Java protected keyword is an access modifier. It can be accessible within the package and outside the package but through inheritance only. It can't be applied with the class.

33. public: Java public keyword is an access modifier. It is used to indicate that an item is accessible anywhere. It has the widest scope among all other modifiers.

34. return: Java return keyword is used to return from a method when its execution is complete.

35. short: Java short keyword is used to declare a variable that can hold a 16-bit integer.

36. static: Java static keyword is used to indicate that a variable or method is a class method. The static keyword in Java is mainly used for memory management.

37. strictfp: Java strictfp is used to restrict the floating-point calculations to ensure portability.

38. super: Java super keyword is a reference variable that is used to refer to parent class objects. It can be used to invoke the immediate parent class method.

39. switch: The Java switch keyword contains a switch statement that executes code based on test value. The switch statement tests the equality of a variable against multiple values.

40. synchronized: Java synchronized keyword is used to specify the critical sections or methods in multithreaded code.

41. this: Java this keyword can be used to refer the current object in a method or constructor.

42. throw: The Java throw keyword is used to explicitly throw an exception. The throw keyword is mainly used to throw custom exceptions. It is followed by an instance.

43. throws: The Java throws keyword is used to declare an exception. Checked exceptions can be propagated with throws.

44. transient: Java transient keyword is used in serialization. If you define any data member as transient, it will not be serialized.

45. try: Java try keyword is used to start a block of code that will be tested for exceptions. The try block must be followed by either catch or finally block.

46. void: Java void keyword is used to specify that a method does not have a return value.

47. volatile: Java volatile keyword is used to indicate that a variable may change asynchronously.

48. while: Java while keyword is used to start a while loop. This loop iterates a part of the program several times. If the number of iterations is not fixed, it is recommended to use the while loop.