**Java Constants, Variables and Data types:**

**Constants:**

Constants in java refer to fixed values that do not change during the execution of a program. Java supports several types of constants.

1. Numeric constants
2. Integer constants
3. Real constants
4. Character constants
5. Character constants
6. String constants

In java, declaring and defining (together) of constants can be done by using the keyword **final.** It needs to be donein the following format:

final <datatype> <identifier>=<value>;

e.g.

final float PI=3.14159F;

final char NEWLINE=’\n’;

final int MAX=100;

A constant or final variable can only initialized and cannot be assigned a new value thereafter. That means the compiler will not allow to change intentionally or unintentionally the value of the constant variable, as it has been defined as a constant. Any attempt to do so caught by java compiler as an error.

However, we can separate the declaration (i.e. declaring a variable of some datatype) and definition (assigning some value to that declared variable) of a final variable, i.e. we could do the following:

final float PI;

PI=3.14159F;

Note

To assign a value in the float variable we need to assign it like: PI=3.14159F; (to mention that we are assigning the value to a float variable). If we assign it like: PI=3.14159 then compiler will treat 3.14159 as double. Now a double value cannot be assigned to a float variable because of possible loss of precision. So, compiler will generate a compilation error.

Examples of different types of constants:-

* **Integer constants**

123 -321 0 654321 **legal integer constants**

15 750 20,000 $2100 12\_200 **illegal integer constants**

Because, embedded spaces, commas, and non-digit characters are not permitted between digits.

* **Real constants or floating point constants**

0.0083 -0.75 435.36 12.098F legal real constants

**Now, real nos can be represented in a different way (exponential notation or scientific notation)**. For example, the value 215.65 may also be written as 2.1565e2 in exponential notation. Where 2.1565 is the mantissa (M) e is the base and 2 is the power or exponent (E). i.e. a no. can be represented as: **M×BE**

So, the following representations of the real nos. are also valid:

0.65e4 12e-2 1.5e+5 3.18e3 -1.2e-1

* **Single character constants**

A single character constant (or simply character constant) contains a single character enclosed within a pair of single quote marks. Examples of character constants are:

‘5’ ‘X’ ‘;’ ‘ ’ **legal character constants**

* **String constants**

A string constant is a sequence of characters enclosed between double quotes. The characters may be alphabets, digits, special characters and blank spaces. Examples are:

“Hello java” “1997” “Well done” “?.....!” **legal String constants**

* **Backslash character constants**

Java supports some special backslash character constants that are used in output methods. For examples, the symbol ‘\n’ stands for newline character.

|  |  |
| --- | --- |
| **Constant** | **Meaning** |
| ‘\b’ | Back space |
| ‘\f’ | Form feed |
| ‘\n’ | New line |
| ‘\r’ | Carriage return |
| ‘\t’ | Horizontal tab |
| ‘\’ ’ | Single quote |
| ‘\” ’ | Double quote |
| ‘\ \ ’ | backslash |

**Variables:**

A variable is an identifier that denotes a storage location used to store a data value. Unlike constants that remain unchanged during the execution of a program, a variable may take different values at different times during the execution of the program.

**A variable can be declared and defined together or it can be separately declared and defined.**

**Variable declaration and definition together:**

int i=3;

**Variable declaration and definition separately**

int i; //variable declaration

i=3; //assigning value to the variable i.e. variable definition

As mentioned earlier (at the definition of identifier) variable names may consist of alphabets, digits and the underscore (\_) and dollar characters, subject to the following conditions:

* They must not begin with a digit
* Uppercase and lower cases are distinct. This means that the variable Total is not the same as total or TOTAL.
* It should not be a keyword
* White space is not allowed.
* Variable names can be of any length

**Data types**

Every variable in java has a data type. Data types specify the size and type of values that can be stored. Java language is rich in its data types. The varieties of data types available allow the programmer to select the type appropriate to the needs of application. See the following picture to know about all data types in java:

* **Integer types:**

Integer types can hold whole numbers can hold whole numbers such as 123, -96 and 5639. The size of the values that can be stored depends on the integer data type we choose. There are four integer data types.

* **Floating point types:**
* **Character type:**

In order to store character constants in memory, java provides a character data type called char. The char type assumes a size of 2 bytes but, basically, it can hold only a single character (this is because of the ASCII value of the character. ASCII value is represented in 0 and 1. And ASCII value is of unsigned integer type. **Check the Xerox note provided to you**)

* **Boolean type:**

**Boolean type** is used when we want to test a particular condition during the execution of the program. There are only two values that a boolean type can take: true or false. Remember, both these words have

been declared as keywords. Boolean type is denoted by the keyword **boolean** and uses only one bit of storage (if boolean type variable is true then it stores 1 in the storage, and if it is false, it stores 0 in the memory storage)

**Declaration of variables:**

In java, variables are the name of storage locations. After designing suitable variable names, we must declare them to the compiler. Declaration does three things:

* It tells the compiler what the variable name is.
* It specifies what type of data the variable will hold.
* The pace of declaration (in the program) decides the scope of the variables.

A variable must be declared before it is used in the program.

A variable can be used to store a value of any data type. That is, the name has nothing to do with the type. Java allows any properly formed variable to have any declared data type. The declaration statement defines the type of variable. The general form of declaration of a variable is:

<type> variable1, variable2, variable3, variable4 ….variableN;

i.e. we can declare any no. of variables of same data type together. Where variable names are separated by commas. A declaration statement must end with a semicolon. Some valid declarations are:

int count;

float x,y;

double pi;

byte b;

char c1,c2,c3;

**Giving values to variables:**

A variable must be given a value after it has been declared but before it is used in an expression. This can be achieved in two ways:

* By using an assignment statement
* By using a read statement (for user input. You already know a lot of things about taking user-input. Don’t you?)
* **Assignment statement:**

A simple method of giving value to a variable is through the assignment statement as follows (we sometimes mention it as variable definition) :

<Variable name> =value;

For example:

Initialvalue=0;

Finalvalue=100;

Yes=’X’;

It is possible to assign a value to a variable at the time of its declaration. This takes the form:

<type> <variable name> =value;

**Examples:**

int Finalvalue=100;

char Yes=’X’;

double total=75.36;

The process of giving initial values to variables is known as **initialization**. The ones that are not initialized are automatically set to zero.

**The following are valid java statements:**

float x,y,z;

int m=5,n=10;

int m,n=10;

* **Read statement:**

We have already discussed a lot about read statements in the programs we have done so far. (it is needed to take the user input)

**Scope of variables:**

Java variables are actually classified into three kinds:

* instance variables
* class variables and
* local variables

instance variables and class variables both are declared inside a class.

Instance variables are created when the objects are instantiated and therefore are associated with the objects. They take different values for each object. (instance variables of a class have different values for different runtime instance (object) of the class. They are maintained differently for different instances of the class. Any data member of a class which is non-static (a static member in not maintained separately for different instances of the class. It is maintained entirely for an object). On the other hand, class variables are global to a class and belong to entire set of runtime instances of the class (i.e. entire set of objects of the class) . So, class variables must be static data members of a class, since they are not maintained separately for different instances of the class. They are maintained for the entire class.

Variables created and used inside methods are called local variables. They are called so because they are no available for use outside the method definition. Local variables can also be declared inside program blocks that are defined between an opening brace (‘{“) and a closing brace (“}”) (Remember, the creating scope lesson?) These variables are visible to the program only from the beginning of its program block to the end of the program block. When the program control leaves a block (such a block), all the variables in the block will cease to exist. The area of the program where the variable is accessible is its scope.

class Student

{

private String Name;

private int Age;

private char Gender;

private String Department;

//top four are instance variables

private static int Studentcount=0;

//last one is a class variable.

………………………………

………………………………

//class methods implementations

}

Here, in this class definition the instance variables are: **Name, Age, Gender, and Department .** They are maintained differently for each instance of the Student class (i.e. maintained differently for different students).

But, the **Studentcount** is a static int variable and it is a class variable. Suppose, we want to create a section (which consists of limited no. of students) and hence, we want to maintain a Studentcount for maintaining total entry in that particular section. So, Studentcount will be maintained for each students of that section . So, we are in need of a variable which will be maintained entirely for a set of objects of Student class (rather than maintained separately for each of the students of that set (section can be considered as a set of students)).

import java.io.\*;

class Mainsclass

{

public static void main(String args[])throws IOException

{

BufferedReader z=new BufferedReader(new InputStreamReader(System.in));

int totalcount,Age,index=0;

String SectionName;

String Name;

String Department;

char Gender;

…………………………………………………….

……………………………………………………..

}

}

Now all these variables which is declared in the method i.e. totalcount, Age, index, SectionName,Name, Department, Gender and even z (of BufferedReader class) are local variables. Because their scope in a method. Outside that method they cease to exist. (the method need not to be main necessarily. It could be any method of a class. If that method uses some variables which are not data members of the class, then they are local variables to that method) Now the question is z is an object of BufferedReader class. Though it’s scope is main method here, can it be called as a local variable? **Ans is Yes**. Because a class is a non-primitive (derived) datatype (though it is derived, it’s a datatype) and it’s object is clearly a variable of that class. Now, local variables need not to belong to a method necessarily. It could belong to a scope created by an opening brace and a closing brace.

**Check the following example:**

class Mainsclass

{

public static void main(String args[])

{

int i=10;

…………………………………………………….

…………………………………………………….

{

int i=5;

System.out.println(“displaying the value of I within scope:”+i);

}

System.out.println(“displaying the value of I within main’s scope:”+i);

……………………………………………………

…………………………………………………….

}

}

Now, the value of i within the scope created by a pair of opening and closing brace is 5. (because this value is local to the scope) and the value of i outside that scope but within main’s scope is 10.