DAY 13 Saturday, 28 June 2017

1. **Java Array**

Array means collection. In Java also, an array is a collection of similar things.

**Declaration of array-**

int[] z = new int[10];

int[] z is representing that 'z' is an array of type integer ( as [ ] represents array ) and 'new int[10]' is representing that is it an array of 10 integers. In other words, we can say that 'z' is a collection of 10 integers.

ex.

class A0{

public static void main(String[] args){

int[] z = new int[10];

z[0] = 223;

z[1] = 23;

int[] x = {23,32,43,12,43};

System.out.println(z[0]);

System.out.println(z[1]);

System.out.println(x[3]);

System.out.println(x[0]);

}

}

Output

223

23

12

23

1. **Wrapper Class**

These classes are used to wrap the data in a new object which contains the value of that variable. This object can then be used in a way similar to how other objects are used.

ex.

Integer intObject = new Integer (34);

The Integer class is the wrapper class that has been provided for the int data type.

The eight classes of java.lang package are known as wrapper classes in java.

Data Type Wrapper Class

byte Byte

short Short

int Integer

long Long

float Float

double Double

char Character

boolean Boolean

ex. conversion between types as-

int x = Integer.parseInt("34"); // x=34

double y = Double.parseDouble("34.7"); // y =34.7

String s1= String.valueOf('a'); // s1="a"

String s2=String.valueOf(true); // s2="true"

1. **Recursion in Java**

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function. Using recursive algorithm, certain problems can be solved quite easily.

ex.

public class MainClass {

public static void main(String[] args) {

int theAnswer = triangle(12);

System.out.println("Triangle=" + theAnswer);

}

public static int triangle(int n) {

if (n == 1)

return 1;

else

return (n + triangle(n - 1));

}

}

Output

Triangle=78

1. **Call by Value**

Call by value method is copying the value of the parameter onto another variable and the function accesses the copied variable. Any changes on these variables won't affect the value outside as they are independent.

In call by value, the changes made to the values of the formal parameters does not affect the values of the actual parameters.

ex.

class CallByValue {

public static void main ( String[] args ) {

int x =3;

System.out.println ( "Value of x before calling increment() is "+x);

increment(x);

System.out.println ( "Value of x after calling increment() is "+x);

}

public static void increment ( int a ) {

System.out.println ( "Value of a before incrementing is "+a);

a= a+1;

System.out.println ( "Value of a after incrementing is "+a);

}

}

Output

Value of x before calling increment() is 3

Value of a before incrementing is 3

Value of a after incrementing is 4

Value of x after calling increment() is 3

//The value of x has remain unchanged, even though it was passed as a parameter to the increment() method.

1. **strictfp KEYWORD**

When operations on the floating-point variable are performed in Java, the precision may differ from one platform to another platform. To overcome this problem, Java programming language provide strictfp Keyword which when used with class, method or interface ensures same result on any platform thereby providing better control over floating-point arithmetic. But this keyword cannot be used with constructor, abstract methods and variables.

ex.

When used with class

strictfp class A{}//strictfp applied on class

When used with interface

strictfp interface M{}//strictfp applied on interface

When used with methods

class A

{

strictfp void m(){}//strictfp applied on method

}

1. **Java Doc tool**

The javadoc tool generates HTML documentation from Java source files.

The javadoc tool locates and parses the documentation comments present in a Java source file and produces a set of HTML pages describing the class, methods, and variables present in the source file.

Syntax –

javadoc [options] [packagenames] [sourcefilenames] [-subpackages pkg1:pkg2:...] [@argfiles]

1. options refers to the command line options that you can pass to the javadoc tool.
2. packagenames refers to multiple packages, separated by spaces, that javadoc should use to generate documentation. For example:
3. packagesnames java.lang java.lang.reflect java.awt
4. sourcefilenames refers to the source files for which javadoc needs to generate documentation.
5. -subpackages: Generates documentation from the source files in specified packages and recursively in their subpackages.
6. @argfiles: Refers to a file that contains Javadoc options.
7. **Command line arguments**

Java supports command line arguments, which means you can pass parameters/arguments to the program. Depending upon the parameters passed to the program, it can be made to execute differently.

ex.

class CommandLineArguments

{

public static void main(String args[])

{

if(args.length == 0)

{

System.out.println("Running the program with no arguments");

}

else if(args.length == 1)

{

System.out.println("Running the program with one argument - " + args[0]);

}

else if(args.length == 2)

{

System.out.println("Running the program with two argument - " + args[0] + " and " + args[1]);

}

else

{

System.out.println("Invalid number of arguments only 0 or 1 or 2 arguments accepted.");

}

}

}

D:\programs> java CommandLineArguments abc mac

Running the program with 2 arguments – abc and mac

1. **Object vs. Class**

|  |  |  |
| --- | --- | --- |
| No. | Object | Class |
| 1) | Object is an instance of a class. | Class is a blueprint or template from which objects are created. |
| 2) | Object is a real world entity such as pen, laptop, mobile, bed, keyboard, mouse, chair etc. | Class is a group of similar objects. |
| 3) | Object is a physical entity. | Class is a logical entity. |
| 4) | Object is created through new keyword mainly e.g. Student s1=new Student(); | Class is declared using class keyword e.g. class Student{} |
| 5) | Object is created many times as per requirement. | Class is declared once. |
| 6) | Object allocates memory when it is created. | Class doesn't allocated memory when it is created. |
| 7) | There are many ways to create object in java such as new keyword, newInstance() method, clone() method, factory method and deserialization. | There is only one way to define class in java using class keyword. |