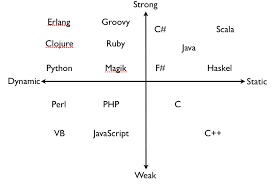
****

**Weakly typed -- as it supports implicit conversion**

**float f = 3.14f;**

**int i=f ;**

**(C supports this) but java does not support, needs explicit conversion like int i = (int)f;**

**Static :- type of the variable is known at compile time.**

**Dynamic :- type of the variable is known at run time.**

**A version number for a s/w product comprises 3 pieces of information :-**

**1. Release number 2. Revision Number 3. Patch number**

**3 types of class loaders:-**

**1. bootstrap class loader (loads all the java.lang, java.util classes)**

**2. Extensions class loader(loads all the jar files)**

**3. System class loader (class path files)**

**C follows top down approach (high level to low level)**

**Java follows bottom up approach (low level to high level)**

**Reserved words vs Keywords**

**Reserved words are (goto and const) that cannot be used as identifier**

**true,false & null are called as literals**

**(null & empty are not same) String s1=null; String s2=””; String s3 =” “;**

**Keyword is a word that has a special meaning in certain context and cannot be used as a variable, classes & methods.**

**eg. class, interface, abstract,public.**

**C vs Java**

**C is a process oriented model can be thought of code acting on data (problems with this approach appear as programs grow larger & larger)**

**OOP organises a program around it data (objects) & a set of well defined interfaces to that data (data controlling access to code)**

**(here the complexity can be managed)**

**Encapsulation is a part of Abstraction.**

**(The details of engine & gearbox are encapsulated (into the interface) in order to create an abstraction.**

**local variable cannot be static**

**variables cannot be abstract**

**class cannot be static**

**Access specifiers are : - private, protected, package, public (specifies how to access)**

**Access modifiers are :- transient, final, static, abstract, synchronized (modifies the access)**

**Rascal is a hybrid approach (both static & dynamic language). It is a meta programming meaning they work on programs instead normally other languages work on data.**

**What is JDK, JRE & JVM ?**

**JDK is a bundle of software that is used to develop java based software.**

**JRE is an implementation of the Java Virtual machine which actually executes the java programs.**

**JVM serves as a link between the java libraries and the platform specific implementation of JRE.**

**What is garbage collection ?**

**It frees the object that goes out of scope. GC is an application program that runs inside JVM. When you start JVM, the GC application also runs at that point, when you end your application, just before the JVM has to come out of its execution, the GC execution is stopped & then the JVM has to end its exection.**

**RuntimException**

**IllegalArgumentException**

**NumberFormatException**

**Exception**

**IOException**

**FileNotFoundException**

**Checked exceptions :- classnotfoundexception, ioexception, interruptedexception, sqlexception, illegalaccessexception**

**Unchecked exception :- nullpointerexception, arrayindexoutofboundsexception, numberformatexception, classcastexception, negativearraysizeexception.**

**J2SE 5.0 TIGER sep 2004**

**java se 6 MUSTANG dec 2006**

**java se 7 DOLPHIN july 2011**

**java se 8 ? mar 2014**

**how to print Hello without using semicolon in the program**

class Sreeni

{

public static void main(String[] args)

{

if(System.out.printf( "hello, world" )== null ){}

}

}

**how to print Semicolon without using Semicolon in the program**

class Sreeni

{

public static void main(String[] args)

{

if(System.out.printf("%c",59)==null) { }

}

}

**How to print Hello without using System class inside main() method**

import static java.lang.System.\*;

class Sreeni

{

public static void main(String[] args)

{

out.println("Hello there");

}

}

**How to compile & execute if class name is different from file name**

You can compile with file name & execute with class name.

say File name = Hello

Class name = Hai

c:/> javac Hello.java

c:/> java Hai

**How to compile & execute only with file name if both class & file name are different**

class Name is FirstMain, save it as SecondMain. Now we can compile & execue with SecondMain

class FirstMain

{

public static void main(String[] args)

{

System.out.println(“Hello”);

}

}

class SecondMain exends FirstMain {}

How to find the nth root of number

public class Sreeni {

public static void main(String[] args) {

int x = 27;

int n = 3;

double a = Math.log(x)/n;

double r = Math.exp(a);

//double r = Math.pow(Math.E, Math.log(x)/n);

System.out.println(r);

}}

How to print the value of PI

class Sreeni

{

public static void main(String[] args)

{

double pi = Math.PI;

System.out.printf ("pi = %5.3f%n", pi);

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**// Left shifting a byte value.**

class ByteShift {

public static void main(String args[]) {

byte a = 64;

int i = a << 2;

byte b = (byte)i;

System.out.println("Original value of a: " + a);

System.out.println("i and b: " + i + " " + b);

}

}

Original value of a: 64

i and b: 256 0 100000000 00000000

**Right Shift (for all positive numbers)**

class Main {

public static void main (String args[] ) {

int a = 35;

a = a >> 2;

System.out.println("value of a: " + a);

}}

a = 00100011 // 35

a>> 2=00001000 // 8

Signed Right Shift (for negative numbers)

class Main {

public static void main (String args[] ) {

int a = -1;

a = a >> 24; // shifts 24 bits right & higher order bits filled with ones

System.out.println("value of a: " + a);

}}

+1 = 00000000 00000000 00000000 00000001

-1 = 11111111 11111111 11111111 11111111

a>>24=11111111 11111111 11111111 11111111

Unsigned Right Shift Operator

class Main {

public static void main (String args[] ) {

int a = -1;

a = a >>> 24; // shifts 24 bits right & higher order bits filled with zeros

System.out.println("value of a: " + a);

}}

-1 = 11111111 11111111 11111111 11111111

a>>>24 = 00000000 00000000 00000000 11111111 = 255

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Which is the only unsigned integer type in java ?**

char data type (Range is 0 to 2^16-1)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**What is the range of int type**

-2^31 to 2^31-1

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**What is the output ?**

public class Test {

public static void main(String[] args) {

char ch = 'd';

if(ch < 132.00){ System.out.println("hello"); }

else

System.out.println("hai"); }}

hello is printed since d = 100 (ascii value)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**What is the output ?**

public class Test { // -128 to 127

public static void main(String[] args) {

byte b=10;

b = b+10; // compilation error

b+=10; // no error 🡪output is 20 (faster than first stmt)

System.out.println("The value of b is " + b); }}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**What is the output ?**

public class Test {

public static void main(String[] args) {

int i = 0;

i=i++;

i=i++;

i=i++;

i=i++;

System.out.println("The value of i is " + i); }}

0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**What is the output ?**

int arr []= new int [5];

int ind= 0;

arr [ind]= ind = 3; // arr[0]= 0=3;

System.out.println("The value of first element is " + arr[0]);

The value of first element is 3 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**constructor circular dependency**

class A

{

public static int X;

static { X = B.Y + 1;}

}

class B

{

public static int Y = A.X + 1;

static {}

public static void main(String[] args) {

System.out.println("X = "+A.X+", Y = "+B.Y);

}

}

**X=1 Y=2**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Print from command line as java Hello CHENNAI SUPER KINGS**

class B

{

public static void main(String[] args)

{

for(int i=0;i<args.length;i++)

System.out.println("args["+i+"] = "+args[i]+" "); }}

output :-

args[0] = CHENNAI

args[1] = SUPER

args[2] = KINGS

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**How to subtract 2 numbers without using subtraction operator**

**public class hello**

**{**

**public static void main( String[] argv )**

**{**

**int a = 10, b = 5;**

**System.out.println(a + (~b + 1)); }}**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**How to add 2 numbers without using Arithmetic operators**

**public class hello {**

**public static void main( String[] argv ) {**

**int a = 3, b= 2;**

**System.out.println(add(a,b));**

**}**

**static int add(int a,int b) {**

**if(a==0)**

**return b;**

**else if(b==0)**

**return a;**

**else**

**return add((a & b) << 1, a ^ b); } }**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**How to reverse a String**

public class StringRevChar {

public static void main(String[] argv) {

String sh = "FCGDAEB";

System.out.println(sh + " -> " + new StringBuffer(sh).reverse()); }}

**String is immutable (performance is lost)**

**String Buffer is mutable (synchronised, hence thread safe)**

**String Builder is mutable(unsychronised, hence not thread safe)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**String word Palindrome**

public class hello {

public static void main(String[] args) {

String str1 = new String("madam");

String str2 = new String();

for (int i = str1.length()-1; i >= 0; i--)

str2 = str2 + str1.charAt(i);

if(str1.equals(str2))

System.out.println("palindrome");

else

System.out.println("Not a palindrome");

}}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**String phrase Palindrome**

String s = "madam i' madam";

s = s.replaceAll("\\s+", "");

s= s.replaceAll("'", "");

System.***out***.println(s);

StringBuffer sb = **new** StringBuffer(s);

**if**(s.equals(sb.reverse().toString()))

System.***out***.println("Its a palindrome");

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**How to print Chennai Super Kings without writing anything inside the main method ?**

public class Ex1{

static{

System.out.println("Chennai");

main();

System.out.println("Kings");

System.exit(0);

}

public static void main(){

System.out.println("Super");

}

}

**To check if a number is even/odd without arithmetic operators.**

int n = 7;

System.out.println(n);

String x = Integer.toBinaryString(n).toString();

System.out.println(x);

int y = x.length();

y--;

if(x.charAt(y) != '1'){

System.out.println("Even");

}else

System.out.println("Odd");

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

public class Sreeni {

public static void main(String[] args) {

System.out.println(isOdd(5));

}

public static boolean isOdd(int i)

{

return (i & 1) != 0; // 101 & 001 = 001 so odd number, if 000 then even number

}}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**To check if an array has duplicates**

**HashSet s = new HashSet(Arrays.asList(new Integer[]{1,2,3,4,3,- 1,4,5,6,5,0,0,0,0,-1,-1,7}));**

**//System.out.println(s);**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Integer[] i = new Integer[]{1,2,3,4};**

**int n = i.length;**

**HashSet hs = new HashSet(Arrays.asList(i));**

**if(hs.size() == n){**

**System.out.println("No duplicates");**

**}else System.out.println("Duplicates");**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**To check if an array has duplicates (Long Method)**

int[] arr = new int[]{1,2,3,4,3,4,-1,-1,-1,-2,-1,0,0,0,0,7,4,-1,4};

int[] b = new int[arr.length];

length = 0;

for(int i=0,j=0;i<arr.length;i++){

if(lookup(b,arr[i])){

b[j] = arr[i];

j++;

length++;

}

}

arr = b;

display(arr);

}

public static boolean lookup(int[] b,int n){

for(int i=0;i<length;i++){

if(b[i] == n) return false;

}

return true;

}

public static void display(int[] b){

for(int i=0;i<length;i++){

System.out.print(b[i]+" ");

}System.out.println();

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**To find the vowels in the given string.(no repitititon)**

String str = "sreeni is cool";

Pattern pat = Pattern.compile("[aeiou]");

Matcher mat = pat.matcher(str);

HashSet hs = new HashSet();

while(mat.find()){

hs.add(str.charAt(mat.start()));

}

System.out.println(hs); // [e, i, o]

System.out.println(hs.size()); //3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**How to use BigDecimal (size is 32 bytes) ( BigInteger is 56 bytes )**

import java.math.BigDecimal;

public class Hello {

public static void main(String[] args) {

System.out.println(2.00-1.10);

System.out.println((new BigDecimal(Math.PI\*2).subtract(new BigDecimal(Math.PI)))); }}

output = 3.141592653589793115997963468544185161590576171875

**What is the output**

class Hello{

public static void main(String[] args) {

int a = 10;

a = a + (a=a+2); // 10+12=22

a=(a+=2)+a; //12 +12=24

System.out.println(a); }}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Jagged Array example**

public class Hello {

public static void main(String args[]) {

int[][][] jag = new int[2][2][];

jag[0][0] = new int[] {1};

jag[0][1] = new int[] {2,3};

jag[1][0] = new int[] {4};

jag[1][1] = new int[] {5,6};

for(int i=0;i<jag.length;i++)

{

for(int j=0;j<jag.length;j++)

{

for(int k=0;k<=j;k++)

System.out.print(jag[i][j][k] + " ");

System.out.println();

}

System.out.println();

}}}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

String s1 = "spring ";

String s2 = s1 + "summer ";

s1.concat("fall ");

s2.concat(s1);

s1 += "winter ";

System.out.println(s1 + " " + s2);

**What is the output? For extra credit, how many String objects and how many**

**reference variables were created prior to the println statement?**

Answer: The result of this code fragment is spring winter spring summer.

There are two reference variables, s1 and s2. There were a total of eight String objects

created as follows: "spring", "summer " (lost), "spring summer", "fall" (lost), "spring

fall" (lost), "spring summer spring" (lost), "winter" (lost), "spring winter" (at this point

"spring" is lost). Only two of the eight String objects are not lost in this process.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**length vs length()**

public class ArrayTest {

public static void main(String[] args) {

int[] test = {12,1,4};

String test2 = "Hoo";

System.out.println(test.length);

System.out.println(test2.length());

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Even/Odd without using mod operator**

**public class Interview {**

**public static void main(String[] args) {**

**System.out.println(isOdd(5));**

**}**

**public static boolean isOdd(int i)**

**{**

**return (i & 1) != 0;**

**}**

**}**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**To find the powers of 2 (without using pow() function)**

// 1 2 4 8 16 32 64 128 256 512

public class Ex1 {

public static void main(String... args) {

for(int i=0;i<10;i++)

System.out.println(fun(i));

}

static int fun(int x)

{

return (1<<x);

}}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**sign & zero extension**

System.out.println((int)(char)(byte)-1); // 65535 zero extension

System.out.println((byte)(int)(char)-1); // -1 sign extension

System.out.println((int)(char)-1); // 65535 zero extension

System.out.println((byte)(char)-1); // -1 sign extension

System.out.println((int)(byte)(char)-1); // -1 sign extension

System.out.println((byte)(char)(int)-1); // -1 sign extension

sign extension is performed if the type of the original value is signed.

zero extension if it is a char, regardless of the type to which it is being converted.

1111 1111 1111 1111 1111 1111 1111 1111

(byte)-1 = 11111111 = -1

(char)-1 = 11111111 11111111 = 65535

(int)65535 = 00000000 00000000 11111111 11111111 = 65535

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**What is the output ?**

char x = 'X';

int i=0;

System.out.println(true ? x : 0);

System.out.println(false ? i : x);

X88

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**How to print "Hello" like this with quotes**

public class Hello {

public static void main(String[] args) {

System.out.println("\"Hello\"");

}}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**How to print true without using double quotes**

public class Hello {

public static void main(String[] args) {

System.out.println(10>9);

}}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Without using printf or println howto print "HelloWorld"**

import java.io.\*;

public class PrintStreamDemo {

public static void main(String[] args) {

String s = "Hello World";

// create printstream object

PrintStream ps = new PrintStream(System.out);

// format a string

ps.format("%s", s);

// flush the stream

ps.flush();

// System.out.format("%s","Hello"); } }

**//** When you write data to a stream, it is not written immediately, and it is buffered. So use flush() when //you need to be sure that all your data is written. (writes saved before closing stream)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**How to print 123 from the following code:-**

char[] numbers = {'1', '2', '3' };

String s = new String(numbers);

System.*out*.println(s);

**How to print ABC from the following code:-**

**byte**[] numbers = {65,66,67 };

String s = **new** String(numbers);

System.***out***.println(s);

**Without using for loop how to print the alphabets //from A..Z in Java**

**byte** ascii[] = {65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,

87,88,89,90};

String alpha = **new** String(ascii);

System.***out***.println(alpha);

String value="";

**for**(**int** i=0;i<alpha.length();i++)

value+=Character.*toString*(alpha.charAt(i))+" ";

System.***out***.println(value);

// to give spaces for the alphabets

//alternative method to give spaces for the alphabets

**char** ch;

**for**(**int** i=0;i<alpha.length();i++)

{

ch=alpha.charAt(i);

System.***out***.print(ch + " ");

}

### Logical AND (&&)

or conditional AND operator is also called short-circuit operator

### Logical OR (||)

### Boolean AND (&)

### Boolean OR (|)

### Boolean XOR (^)

### Boolean NOT (!) is a unary operator

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**How to print your names continuously in runtime**

import java.util.Scanner;

class Hello

{

public static void main(String args[])

{

char ch;

do

{

Scanner sc = new Scanner(System.in);

String name = sc.nextLine();

System.out.println("Your Names" + name);

System.out.println("Your Other names, continue type y/n");

ch = sc.next().charAt(0);

}while(ch=='y');

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**How to swap 2 numbers in a single statement**

int x=11 ,y=8;

x=x+y-(y=x);

System.***out***.println(x + " " + y);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**5**

**4 5**

**3 4 5**

**2 3 4 5**

**1 2 3 4 5**

**0 1 2 3 4 5**

package mypack;

public class Hello

{

public static void main (String arg[ ])

{

String k= " " ;

for (int i = 5; i >= 0; i--)

{

k = i + " " + k;

System.out.println (k + "\n");

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**How to write constructor for class P**

class M

{

M() { }

class N extends M

{

N() { super(); } // super() invokes the object constructor

}

class P extends N

{

// insert code here

}

}

P() { M.this.super(); } // super() invokes the N constructor

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* **Arrays & String-handling**
  + **Understanding references**

Arrays are objects in Java. And references to objects are passed by value.

Java will create a ***copy or copies*** of the values inside the original variable(s) and pass that to the method as arguments – and that is why it is called pass by ***value***.

private void demo() {

int[] array = new int[5];

System.out.println(Arrays.toString(array)); // 0, 0, 0, 0, 0

fillArray(array);

System.out.println(Arrays.toString(array)); // 0, 1, 2, 3, 4

}

private void fillArray(int[] array) {

for (int i = 0; i < array.length; i++) {

array[i] = i;

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In Java, an array is passed by reference, but the reference is passed by value. Suppose you have an array arr. When you pass it, you can change the array that arr refers to, but you cannot change which array arr refers to; i.e. inside a method, you can modify the referenced object but you cannot modify the passed variable that will still be a reference to the same object.

public static void swap(StringBuffer a, StringBuffer b)

{

StringBuffer t = a;

a = b;

b = t;

}

public static void change(StringBuffer a, StringBuffer b)

{

a = a.append("1");

b = b.append("2");

}

public static void main(String[] args)

{

StringBuffer a = new StringBuffer("First");

StringBuffer b = new StringBuffer("Second");

swap(a, b);

System.out.println(a + " " + b);

change(a, b);

System.out.println(a + " " + b);

}

Output:

First Second

First1 Second2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Everything in Java are passed-by value.. In case of Array(Which is nothing but an Object), array reference is passed by value.. (Just like an object reference is passed by value)..

When you pass an array to other method, actually the reference to that array is copied..

* Any changes in the content of array through that reference will affect the original array..
* But changing the reference to point to a new array will not change the existing reference in original method..

**public static void changeContent(int[] arr) {**

**// If we change the content of arr.**

**arr[0] = 10; // Will change the content of array in main()**

**}**

**public static void changeRef(int[] arr) {**

**// If we change the reference**

**arr = new int[2]; // Will not change the array in main()**

**arr[0] = 15;**

**}**

**public static void main(String[] args) {**

**int [] arr = new int[2];**

**arr[0] = 4;**

**arr[1] = 5;**

**changeContent(arr);**

**System.out.println(arr[0]); // Will print 10..**

**changeRef(arr);**

**System.out.println(arr[0]); // Will still print 10..**

**// Change the reference doesn't reflect change here..**

**}**

Important point to note is that **“the reference is copied as a value”** to a new variable and it is given as formal parameter to the called method. It does not get a1 variable which is the actual argument in scope. This is the key difference between pass by value and pass by reference.

**Java uses pass by value. There is no pass by reference in Java**

**Pass by value:-**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**class Swap {**

**public static void main(String args[]) {**

**Animal a1 = new Animal("Lion");**

**Animal a2 = new Animal("Crocodile");**

**System.out.println("Before Swap:- a1:" + a1 + "; a2:" + a2);**

**swap(a1, a2);**

**System.out.println("After Swap:- a1:" + a1 + "; a2:" + a2);**

**}**

**public static void swap(Animal animal1, Animal animal2) {**

**Animal temp = new Animal("");**

**temp = animal1;**

**animal1 = animal2;**

**animal2 = temp;**

**}**

**}**

**class Animal {**

**String name;**

**public Animal(String name) {**

**this.name = name;**

**}**

**public String toString() {**

**return name;**

**}**

**}**

**Output:-**

**Before Swap:- a1:Lion; a2:Crocodile**

**After Swap:- a1:Lion; a2:Crocodile**

**Passing reference as value:-**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**public class Swap {**

**public static void main(String args[]) {**

**Animal a = new Animal("Lion");**

**System.out.println("Before Modify: " + a);**

**modify(a);**

**System.out.println("After Modify: " + a);**

**}**

**public static void modify(Animal animal) {**

**animal.setName("Tiger");**

**}**

**}**

**class Animal {**

**String name;**

**public Animal(String name) {**

**this.name = name;**

**}**

**public String toString() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**}**

**Output:-**

**Before Modify: Lion**

**After Modify: Tiger**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

String

--------

**public** **class** Test {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

String a =**new** String("abc");

String b = a;

*f*(b);

b+="fg";

System.***out***.println(a + " " + b);

}

**public** **static** **void** f(String b){

b+="de";

b=**null**;

}

}

Output:-

abc abcfg

Array of Objects

class Student {  
   int marks;  
}

public static void main(String[] args) {  
    Student[] studentArray = new Student[7];  
    studentArray[0] = new Student();  
    studentArray[0].marks = 99;  
    System.out.println(studentArray[0].marks); // prints 99  
    modify(studentArray[0]);  
    System.out.println(studentArray[0].marks); // prints 100 and not 99  
    // code  
}

public static void modify(Student s) {  
    s.marks = 100;  
}

creates the array which can hold references to seven Student objects. It doesn't create the Student objects themselves. They have to be created separately using the constructor of the Student class.

**class** A

{

**int** i = 10;

}

**class** B **extends** A

{

**int** j = 20;

}

**class** C **extends** B

{

**int** k = 30;

}

**class** D **extends** C

{

**int** m = 40;

}

**public** **class** Test {

**public** **static** **void** main(String[] args) {

A[] a = {**new** A(), **new** B(), **new** C(), **new** D()};

System.***out***.println(a[3].i); // 1

// System.out.println(a[2].j); error

// System.out.println(a[1].k); error

// System.out.println(a[0].m); error

}}

**Array normal copy vs System.copy vs clone vs Array.copyOf**

import java.util.Arrays;

import java.util.HashSet;

import java.util.Set;

class A {}

public class ArrayExample {

public static void main(String[] args) {

int[] a = {1,2,3};

//copies into an existing array. Both points to same object

int[] b =a;/

int[] c = new int[a.length];

//copies into an existing array. Both points to same //object. But faster than the normal copy

System.arraycopy(a, 0, c, 0, a.length);

System.out.println(c[0]); // 1

//Have different object references but contents will be same

int[] d = c.clone(); //

System.out.println(d[0]); // 1

a[0]=0;

System.out.println(a[0]); // 0

System.out.println(b[0]); // 0

System.arraycopy(a, 0, c, 0, a.length);

System.out.println(c[0]); // 0

System.out.println(d[0]); // 1 }}

**int**[] copied = Arrays.copyOf(arr, 10); *//10 is the length of the new array*

System.out.println(Arrays.toString(copied));

copied = Arrays.copyOf(arr, 3);

System.out.println(Arrays.toString(copied));

//Arrays.copyOf does not only copy elements, it also creates a new array. Internally it uses

System.arraycopy()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

String s1 = "hill5";

String s2 = "hill" + "5";

System.out.println(s1==s2); // known at compile time (reference //values are same)

}

public static void jill() {

String s1 = "hill5";

String s2 = "hill" + s1.length(); // known at runtime (reference //values are different)

System.out.println(s1==s2);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PATH and CLASSPATH are operating system level environment variables. PATH is used to define where the system can find the executables (.exe) files whereas CLASSPATH is used to specify the location of .class files.

eg. eclipse.exe can be set in the PATH itself to execute from the command prompt.

eg. notepad.exe is set in the classpath as it is a system command already present in c:/windows/system32

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Arrays can’t be redefined

**int**[] a={1,2,3};

a={1,2,3,4};// compilation error

a = new int[4]; // original array cannot be redefined but can be redefined by making same array to point to another new location.

Arrays can’t be resized

(you are only taking copy of array arr & resizing the original array cannot be resized)

**int**[] arr={1,2,3};

arr = Arrays.*copyOf*(arr, arr.length+1);

Removing duplicates fromarray

package mypack;

import java.util.List;

import java.util.\*;

class Chennai {

public static void main(String[] args) {

int a[]={-1,-1,10,20,-4,-4,-4,10,20, 0, 0};

System.out.print("Before Sorting:");

for (int i=0;i<a.length; i++ )

{

System.out.print(a[i]+"\t");

}

System.out.print ("\nAfter Sorting:");

for(int i=0;i<a.length;i++)

{

for(int j=i;j<a.length;j++)

{

if(a[i]>a[j])

{

int temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

}

//After sorting

for(int i=0;i<a.length;i++)

{

System.out.print(a[i]+"\t");

}

System.out.print("\nAfter removing duplicates:");

int b=0;

a[b]=a[0];

for(int i=0;i<a.length;i++)

{

if (a[b]!=a[i])

{

b++;

a[b]=a[i];

}

}

for (int i=0;i<=b;i++ )

{

System.out.print(a[i]+"\t");

}

}

}