

Assignment - 2

Page No.:

Date:

1) Explain with example Control Statements. If, else, nested if, if-else ladder, switch, while, do-while, for, for-each, break, continue.

Ans } • IF :- It will go inside the block only if the condition is true otherwise, it will not execute the block.

Eg:- if (a=0) {
 s.o.p("value of a=0");
}

• IF-Else :- In this statement, if the condition specified is true, the if block is executed. Otherwise, the else block is executed.

Eg:- if (a < 0) {
 s.o.p("a is negative");
}
else {
 s.o.p("a is positive");
}

• NESTED IF :- An if present inside an if block is known as a nested if block. It is similar to an if..else statement except they are defined inside another if..else statement.

It is example of ~~nested~~ if else ladder

```

Eg:- if (a < 0) {
    s.o.p("a is negative");
}
else if (a > 0) {
    s.o.p("a is positive");
}
else {
    s.o.p("a is zero");
}

```

• If else ladders

- It is used to decide among multiple options. The if statements are executed from the top down.
- As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed.
- If none of given condition is true then else part will be executed.

```

Eg:- int s=18;
      if (s > 10) {
          if (s % 2 == 0) {
              s.o.p("s is even & greater than 10");
          } else {
              s.o.p("s is odd & greater than 10");
          }
      }
      else {
          s.o.p("s is less than 10");
      }

```


- Switch :- It is a multi-way branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression.

E.g.:- `switch (cal)`
`{`

`case 1 :`

`s.o.p add = a + b ;`

`break ;`

`case 2 :`

`sub = a - b ;`

`break ;`

`case 3 :`

`div = a / b ;`

`break ;`

`case 4 :`

`mwl = a * b ;`

`break ;`

`default :`

`s.o.p ("Enter valid input") ;`

`}`

- while :- It loops through a block of code as long as a specified condition is true.

Eg :- `int i = 0 ;`
`while (i < 0) {`
`s.o.p (i) ;`
`i ++ ;`
`}`

- Do-while :- This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

E.g.:-

```
int i=0;  
do {  
    s.o.p(i);  
    i++;  
}  
while (i<5);
```

- for :- when you know exactly how many time you want to loop through a block of code, use the for loop instead of a while loop.

E.g.:-

```
for (int i=0; i<5; i++) {  
    s.o.p(i);  
}
```

- for-each :- It is another array traversing technique like for loop, while loop, do-while loop.

- It starts with the keyword for like a normal for-loop
- Instead of declaring and initializing a loop counter variable, you declare a variable that is the same type as the base type of the array, followed by a colon, which is then

followed by the array name.

- In the loop body, you can use the loop variable you created rather than using an indexed array element.
- It's commonly used to iterate over an array or a collections class (eg, ArrayList).

syntax:

```
for (type var : array)
{
    statements using var;
}
```

- Break:- It is used to terminate a loop and break the current flow of the program.

Eg:-

```
for (int i = 5; i < 10; i++)
{
    if (i == 8)
        break;
    s.o.p(i);
}
```

OUTPUT: 5 6 7

- Continue:- To jump to the next iteration of the loop, we make use of the continue statement.
- This statement continues the current flow of the program and skips a part of the code at the specified condition.

Eg:- for (int k=5; k<15; k++)
 {

if (k % 2 != 0)

continue;

S.O.P (k + " ");

}

OUTPUT: 6 8 10 12 14

2) write a program to find whether the given string is palindrome or not.

~~ans:-~~ import java.util.*;

class String {

static boolean isPalindrome (String str)
 {

int i=0, j=str.length()-1;

while (i<j) {

if (str.charAt(i) != str.charAt(j))

return false;

i++;

j--;

}

return true;

}

public static void main (String[] args)
 {

String str = "geeks";

if (isPalindrome (str))

S.O.P ("Yes");

else

S.O.P ("No");

}

}

OUTPUT: No

3) Write a method for computing x^y doing repetitive multiplication. x and y are of type integer and are to be given as command line arguments. Raise and handle exception(s) for invalid value of x and y .

Solⁿ:

```
// Package Power;
Public class Power_Example
{
    Public static void main (String args[])
    {
        int x, y, z;
        try
        {
            x = Integer.parseInt (args [0]);
            y = Integer.parseInt (args [1]);
            z = 1;

            x = Integer.parseInt (args [0]);
            y = Integer.parseInt (args [1]);
            z = 1;
            for (int i=0; i<y; i++)
            {
                z = z * x;
            }
            System.out.println (x+"^"+y+"="+z);
        }
        catch (NumberFormatException n)
        {
            System.out.println ("No. format error");
        }
    }
}
```

finally

{

System.out.println("It executes every time...");

}

}

}

OUTPUT:

It executes every time...

4) Write a program for given pattern

```

      1
    1 0
  1 0 1
1 0 1 0
  
```

Solⁿ: import java.util.*;

Public class Pattern

{

public static void main (String[] args)

{

Scanner sc = new Scanner (System.in)

System.out.println("Enter no. of rows");

int rows = sc.nextInt();

for (int i=1; i<=rows; i++)

{

for (int j = rows; j > i; j--)

{

System.out.print(" ");

}

for (int k=1; k <= i; k++)

{


```
if (i % 2 == 1)
{
    System.out.print("1");
}
else
{
    System.out.print("0");
}
}
System.out.println();
}
```

5) Explain in detail types of Operators in java.

- The various types of operators in java are given (Explained below):

1) Arithmetic Operators:

- They are used to perform simple arithmetic operations on primitive data types
- Multiplication (*)
- Division (/)
- Modulo (%)
- Addition (+)
- Subtraction (-)

2) Assignment Operator:

- It is used to assign any value to any variable.
- The associativity of this operator is

right to left

• $+=$, $-=$, $*=$, $/=$, $\%=$

3) Relational Operators :

- These operators are used to check for relations like equality, greater than, less than
- It gives output as a Boolean
- Equal to ($==$), Not Equal to ($!=$), less than ($<$), less than equal to ($<=$), greater than ($>$), greater than equal to ($>=$)

4) Logical Operator :

- These operators are used to perform "logical AND" and "Logical OR" operation
- ($\&\&$) logical AND, ($\|\|$) logical OR

5) Unary Operator :

- It needs only one operator and used to inc. or dec. or negate a value
- unary minus ($-$), unary plus ($+$)
- Inc. operator ($++$), dec. operator ($--$)
- logical not operator ($!$)

6) Ternary Operator :

- It is a shorthand version of if-else statement. It has three operands and hence the name ternary.

Syntax :

condition ? if true : if false :

7) Bitwise operators:-

These operators are used to perform manipulation of individual bits of a number

- Bitwise AND (&)
- Bitwise OR (|)
- Bitwise XOR (^)
- Bitwise complement (~)

8) Shift operators:-

These operators are used to shift the bits of a number left or right thereby multiplying or dividing the number by two respectively

- left shift (<<)
- signed right shift (>>)
- unsigned right shift (>>>)

6) Write a program to create circle with class with area function to calculate area of circle.

Solⁿ: import java.util.Scanner;

class Area

{

public static void main (String args[])

{

int r;

double pi = 3.14, area;

Scanner sc = new Scanner (System.in);

System.out.println("Enter radius of circle:");

r = sc.nextInt();

area = pi * r * r;

System.out.println("Area of circle:" + area);

{

{

OUTPUT :

Enter radius of circle : 5

Area of circle : 78.5