Operators

Operators

- Java provides with series of operators to work upon and manipulate values of a variable

Operators

- Different Categories of Operators Available
  - Unary Operator
  - Arithmetic Operator
  - Comparison Operator
  - Logical Operator
  - Arithmetic Assignment Operator
  - Conditional operator

Operators

- Unary Operators works on a single operand
  - ❑++
  - ❑--
  - ❑!
  - ❑-

Operators

- Unary Operator ++ (increment) works in two ways
  - Pre increment ,also known as prefix
  - Post increment ,also known as postfix

Operators

- In pre increment operation ,value of the variable in incremented first and then this value is used
- In post increment operation ,value of variable is used first and then it is incremented
- Decrement operator works in same way for decreasing value of a variable

Lets see an example in Java

```
public class Program
      {

       public static void main(String[] args)
          {
          int x = 10;
          int y = ++x;
         System.out.println(y);
          y = x++;
          System.out.println(y);

      }
      }
```

Operators

- Arithmetic Operators
  - ❑+ Additive
  - ❑- Subtraction
  - ❑ * Multiplicative
  - ❑ / Division
  - ❑ % modulus(Reminder)

Operators

- Arithmetic Operator precedence

Operator	Category
*,/,%	Multiplicative
+ , -	Additive

Lets see an example of arithmetic operator in Java

```
public class Program
 {
  public static void main(String[] args)
     {
     int x = 5;
     int y = 7;
     int z = 9;
     int exprression = x + y * z;
    System.out.println(exprression);

 }
 }
```

Operator

- Comparison Operator
  - ❑ ==
  - ❑ !=
  - ❑ >=
  - ❑ <=
  - ❑ >
  - ❑ <

Operators

- Comparison operators compare operands and return boolean values

  Lets see an example

```
public class Program
      {

      public  static void main(String[] args)
         {
         int x = 5;
         int y = 7;
         boolean expression = x > y;
         System.out.printf("%d > %d is %s \n",x,y,expression);
         exprression = x < y;
         System.out.printf("%d < %d is %s \n",x,y,expression);
         exprression = x <= y;
        System.out.printf("%d <= %d is %s \n"  ,x,y,expression);
         exprression = x >= y;
         System.out.printf("%d >= %d is %s \n",x,y,expression);
         exprression = x == y;
        System.out.printf("%d == %d is %s \n",x,y,expression);
         exprression = x != y;
         System.out.printf("%d != %d is %s \n",x,y,expression);

      }
}
```

Operators

- Comparison Operator Precedence

Operator	Category
> , < ,>=,<= ,instanceOf	Relational an type testing
== ,!=	Equality

Operators

- Logical Operators
  - &&
  - ||
  - !

Operators

- Logical operators are used to evaluate multiple conditions and return true or false
- && operator returns true only if expression on both the side of operator is true
- || operator returns true if any of the expression on either side of operator is true
- ! Operator returns true for false expression and false for true expression

```
public class Program
    {
     static void main(String[] args)
        {
        int age = 30;
        int exp = 7;
        int marks = 72;
        boolean result=age > 29 && marks > 50;
       System.out.printf("Age is %d \n", age);
       System.out.printf("Marks are %d \n", marks);
        System.out.printf("Experiance is %d years \n", exp);
        System.out.println("Age > 29 and marks > 50");
        System.out.println(result);
        result = age > 29 && exp > 7;
       System.out.println("age > 29 and exp > 7");
       System.out.println(result);


    }
    }
```

Operators

And Evaluation

Expression 1	Expression 2	Result
True	True	True
True	False	False
False	False	False

Operators

OR Evaluation

Expression 1	Expression 2	Result
True	True	True
True	False	True
False	False	False

Operators

Not Evaluation

Expression 1	Result
True	False
False	True

Operators

- Logical Operator Precedence

Operator	Category
!	Unary
&&	Logical
||	Logical

Operators

- Arithmetic Assignment Operators
  - +=
  - -=
  - *=
  - /=
  - %=

Operators

- Arithmetic assignment operator performs operation on operand and transfers value from right to left

  Lets see an example

```
public class Program
    {

    public static void main(String[] args)
       {

       int x = 10;

       int y = 20;

       System.out.printf("Value of y is %d", y);

       x += y;

      System.out.printf("Result of x+=y  is %d", x);
      System.out.printf("Value of x is %d ", x);
     System.out.printf("Value of y is %d", y);



    }
    }
```

Conditional Operator

- Conditional operator returns a value based on evaluation of a boolean expression
- Syntax for conditional operator

variable=[boolean expression]?[value 1]:[value2]

Lets see an example

Example for conditional operator in Java

```
public class Program
   {
      public static void main(String [] args)
      {
         int x = 10;
         int y = 20;
         int z = x > y ? x : y;
       System.out.println(z);
      }
   }
```

Lets Summarize

- Operators are special symbols used to manipulate value of a variable
- Some category of operators are Arithmetic ,Comparison, Logic and Arithmetic Assignment operators

Exercises

- Predict the result for each expression

```
int x=5;
 int y=7
```

- int z=++x;
- int z=x++;
- int z=x++ + y++;
- int z=++x + x++;

Exercise

2. solve below given ternary expression

```
int x=12;
int y=13;
  String s= x>y?"x is greater ":"y is greater"
```

Exercise

- 3.boolean z=true;

  boolean result=z?false:true;

  System.out.println(result)

Exercise

4.int x=20;

int y=7;

a int result=x%y;

b int result=y%x;

c x%=y

Conditional Constructs and Loops

Java provides us with logic building tools like conditional constructs and loops

for implementing logic in a program

conditional constructs

#if else

#switch case

Loops

#while

#do while

#for ,for(in)

If else

if else

syntax

```
if(condition)
  {
   }
  else
{
 }
```

if else lab 1

Write a program to evaluate and print a message for below conditions

String signal

can have different values

if signal is RED

STOP

if signal is GREEN

GO

if signal is WAIT

Slow down

if signal is WATCH

Please move carefully

if signal is NO HORN

pleas do not use horn

Switch case

- switch case is used in scenario where a variable can have one of many possible values

and an expression has to be evaluated based on a possible value

switch case is used in menu driven programming

Switch case

```
switch(variable)
 {
case [CONSTANT]:
 //expression if constant matches with value of variable
 break;

...
...

default:
 //none of the statement matches


}
```

Scenario

Jason has to write a class for a game where he has to calculate score .The game to be displayed is of cricket .Jason has to compare the shot by player with different scenarios like

SINGLE
DOUBLE
SIXER
MISSED
FOUR
LEGBYE
WIDE

for each scenario ,score has to increment accordingly

he has decided to use switch case for the same .Help Jason write the program

Loop

Java provides for loops to create constructs ,that can implement iteration in a program

Two type of loops are there

- Variable loop
- Fixed loop

Variable loop

Execution of loop depends on condition.The loop will end when condition becomes false

Fixed loop

In fixed loop no of iterations are known and loop will terminate after fixed no of iterations

Java provides us with different loop constructs

- while loop
- do while loop
- for loop
- enhanced for loop

While loop

while loop

A while executes based on condition

syntax

```
while(condition)
{
 //loop body
}
```

Do while loop

In a do while loop ,loop body is executed first

```
do
 {

 }while(condition);
```

For loop

can be used to create a fixed loop

syntax

```
for(initialization;condition;increment/decrement
expression)
  {

 }
```

Enhanced For Loop

Enhanced for loop is used to iterate through arrays and collections

```
for(<data type> var : <collection/array>)
{
}
```

Java lab for loop

1. Write a program to calculate sum of first 10 even numbers using while loop
2. Write a program to display sum of all even numbers for 2 to 50
3. Write program to print below given design

```
    *
   * *
  * * *
 * * * *
* * * * *
```

Arrays in Java

- Arrays are group of values of same data type stored at adjacent memory location
- Arrays are utilized in cases where more then one value of same data type needs to be stored
- Values in array are accessed using index position
- Size of an array is fixed programmatically and cannot be changed during execution of program
- In java memory to array is allocated at run time using new operator.So array in java is treated as reference type .It also has a length variable that stores size of array

Syntax for creating array

data_type [] identifier=new data_type[size];

position/indexing in array starts from 0;

Java Array Lab

Java Array Lab

1.From the below array find the largest and smallest value

12,34,11,19,55,33,26

2.Sort the below given array

22,11,19,55,34,23,19

Types of Arrays

- Single Dimensional array

  Values are stored as row or column

- Multi Dimensional array

  Values are stored as row and columns similar to a tabular structure