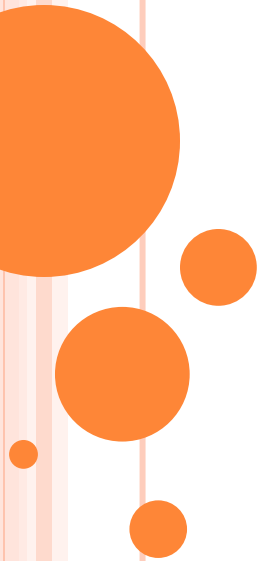


JAVA

Operators

String

Program Structure

Operators

- Arithmetic Operators
- Relational Operators
- Bitwise Operators
- Logical Operators
- Assignment Operators

The Arithmetic Operators

Operator	Description	Example
+	Addition - Adds values on either side of the operator	A + B will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	A - B will give -10
*	Multiplication - Multiplies values on either side of the operator	A * B will give 200
/	Division - Divides left hand operand by right hand operand	B / A will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	B % A will give 0
++	Increment - Increases the value of operand by 1	B++ gives 21
--	Decrement - Decreases the value of operand by 1	B-- gives 19

The Arithmetic Operators

```
public class Test
{
public static void main(String args[]){
int a =10;
int b =20;
int c =25;
int d =25;
System.out.println("a + b = "+(a + b));
System.out.println("a - b = "+(a - b));
System.out.println("a * b = "+(a * b));
System.out.println("b / a = "+(b / a));
System.out.println("b / a = "+(b / a));
System.out.println("b % a = "+(b % a));
System.out.println("c % a = "+(c % a));
System.out.println("a++ = "+(a++));
System.out.println("b-- = "+(a--));
// Check the difference in d++ and ++d
System.out.println("d++ = "+(d++));
System.out.println("++d = "+(++d));
}
}
```

Relational Operators

Assume variable A holds 10 and variable B holds 20, then:

Operator	Description	Example
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Relational Operator

```
public class Test{


public static void main(String args[]){

int a =10;

int b =20;

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

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

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

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

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

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

}

}
```

The Bitwise Operators

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) will give 61 which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -60 which is 1100 0011
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 1111
>>>	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.	A >>>2 will give 15 which is 0000 1111

```
public class Test
{
public static void main(String args[])
{
 int a =60; /* 60 = 0011 1100 */
int b =13; /* 13 = 0000 1101 */
int c =0; c = a & b;/* 12 = 0000 1100 */
System.out.println("a & b = "+ c ); c = a | b;/* 61 = 0011 1101 */
System.out.println("a | b = "+ c ); c = a ^ b;/* 49 = 0011 0001 */
System.out.println("a ^ b = "+ c ); c =~a;/*-61 = 1100 0011 */
 System.out.println("~a = "+ c ); c = a <<2;/* 240 = 1111 0000 */
 System.out.println("a << 2 = "+ c ); c = a >>2;/* 215 = 1111 */
System.out.println("a >> 2 = "+ c ); c = a >>>2;/* 215 = 0000 1111 */
 System.out.println("a >>> 2 = "+ c );
} }
```

The Logical Operators

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true.

```
public class Test
{
 public static void main(String args[])
{
 boolean a =true;
 boolean b =false;
System.out.println("a && b = "+(a&&b));
System.out.println("a || b = "+(a||b));
System.out.println("!(a && b) = "+!(a && b));
}
}
```

The Assignment Operators

=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator	C &= 2 is same as C = C & 2
^=	bitwise exclusive OR and assignment operator	C ^= 2 is same as C = C ^ 2
|=	bitwise inclusive OR and assignment operator	C |= 2 is same as C = C | 2

```
public class Test

{

 public static void main(String args[])

{

 int a =10; int b =20; int c =0;

c = a + b;

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

c += a ; System.out.println("c += a = "+ c ); c -= a ;
System.out.println("c -= a = "+ c ); c *= a ; System.out.println("c *= a =
"+ c ); a =10; c =15; c /= a ; System.out.println("c /= a = "+ c );

a =10; c =15; c %= a ; System.out.println("c %= a = "+ c ); c <<=2;
System.out.println("c <<= 2 = "+ c ); c >>=2; System.out.println("c >>=
2 = "+ c ); c >>=2; System.out.println("c >>= a = "+ c ); c &= a ;
System.out.println("c &= 2 = "+ c ); c ^= a ; System.out.println("c ^= a
= "+ c ); c |= a ; System.out.println("c |= a = "+ c ); } }
```

Misc Operators

- Conditional Operator (? :)
  - ternary operator

- variable x = (expression) ? value if true : value if false

```
public class Test {

   public static void main(String args[]){
      int a , b;
      a = 10;
      b = (a == 1) ? 20: 30;
      System.out.println( "Value of b is : " +  b );

      b = (a == 10) ? 20: 30;
      System.out.println( "Value of b is : " + b );
   }
}
```

INSTANCEOF OPERATOR

- (Object reference variable) instanceof (class/interface type)

- Example
  - String name = “James”;
  - boolean result = name instanceof String; // This will return true since name is type of String

```
Class Vehicle{}
public class Car extends Vehicle
{
 public static void main(String args[])
{
 Vehicle a =newCar();
 boolean result = a instanceofCar;
 System.out.println(result);
 }
 }
```

String

```
public class StringDemo{

   public static void main(String args[])
{
      char[] helloArray = { 'h', 'e', 'l', 'l', 'o', '.'};
      String helloString = new String(helloArray);
      System.out.println( helloString );
      String greeting = "Hello world!";
      System.out.println( greeting );

   }
}
```

String

```
public class StringDemo {

   public static void main(String args[]) {
      String palindrome = "Dot saw I was Tod";
      int len = palindrome.length();
      System.out.println( "String Length is : " + len );
   }
}
```

Concatenating Strings

- string1.concat(string2);
- "My name is ".concat("Zara");

```
public class StringDemo {

   public static void main(String args[]) {
      String string1 = "saw I was ";
      System.out.println("Dot " + string1 + "Tod");
   }
}
```

Creating Format Strings

- printf() and format() methods to print output with formatted numbers
- format() method allows you to create a formatted string that you can reuse

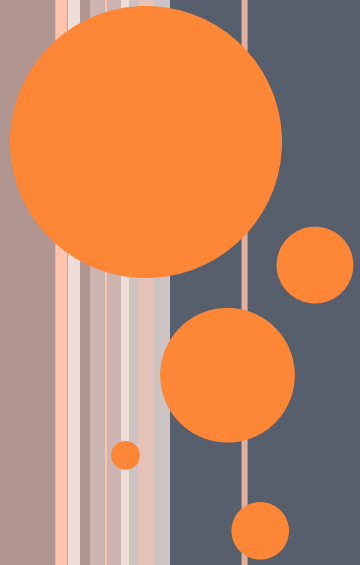
```
System.out.printf(" The value of the float variable is
                  %f, while the value of the integer " +
                  "variable is %d, and the string " +
                  "is %s ", floatVar, intVar, stringVar);

-------------------------------------------------------

String fs;
fs = String.format("The value of the float variable is " +
                   "%f, while the value of the integer " +
                   "variable is %d, and the string " +
                   "is %s", floatVar, intVar, stringVar);
System.out.println(fs);
```

String Functions

- String methods.docx



Program Structure

```
public  class Sample
{
public static void main(String []a)
{
System.out.println("Welcome to Java
   Programming…");
}
}
```