Arithmetic Operator

/ and %

If you divide any value with 0 and if the result is floating then you will get Infinity / -Infinity.

If you divide any value with 0 and if the result is integral then you will get ArithmeticException

class Test{

public static void main(String args[]){

System.out.println(15 / 2);

System.out.println(15 / 2.0);

System.out.println(10 / 3.0F);

System.out.println(10 / 3.0);

System.out.println(15 / 0.0F);

System.out.println(15 / 0.0);

System.out.println(15.0F / 0);

System.out.println(15.0 / 0);

System.out.println(15 / -0.0F);

System.out.println(15 / -0.0);

System.out.println(Byte.MAX\_VALUE);

System.out.println(Short.MAX\_VALUE);

System.out.println(Integer.MAX\_VALUE);

System.out.println(Long.MAX\_VALUE);

System.out.println(Float.MAX\_VALUE);

System.out.println(Double.MAX\_VALUE);

System.out.println(15 / 0); // 2147483647

}

}

String Concatenation Operator (Binary)

+ Operator

Any operand is String

Concatenation or append the values

Result type is always String

int x = 10 + 20;

String y = "10" + 20;

class Test{

public static void main(String args[]){

System.out.println(10 + 20);

System.out.println("10" + 20);

System.out.println(10 + "20");

int x = 10;

int y = 20;

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

System.out.printf("%d+%d=%d\n", x, y, (x+y));

System.out.println("Sum of "+x + " and " + y + " is " + (x + y));

System.out.printf("Sum of %d and %d is %d", x, y, (x+y));

}

}

// x + "+" + y + "="+ (x+y)

// int + String + int + String + int

Assignment Operator (Binary)

=

<op1> = <op2>;

<op1> must be a variable

<op2> can be value, expression, variable

class Test{

public static void main(String args[]){

// 10 = 20;

int x;

x = 23;

System.out.println(x);

x = 12 + 23;

System.out.println(x);

int y;

y = x;

System.out.println(x);

System.out.println(y);

}

}

Type Casting

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

// Assignment Compatible

Implicit Type Casting

Automatically Casting by compiler.

Lower value assigned to wider data type (Widening)

// <widerType> = <lowerType> -> Possible

class Test{

public static void main(String args[]){

byte b = 12;

short s = b; // short <- byte

char c = 'A';

int z = c; // int <- char // Widening

int x = s; // int <- short // Widening

long y = x; // long <- int // Widening

float f = y; // float <- long // Widening

double d = f; // double <- float // Widening

System.out.println(b);

System.out.println(s);

System.out.println(c);

System.out.println(z);

System.out.println(x);

System.out.println(y);

System.out.println(f);

System.out.println(d);

}

}

Explicit Type Casting

-> Can not be automatically Casting by compiler.

-> Programmer is responsible for type casting

-> Wider value assigned to lower data type (Narrowing)

<targetDataType> <targetVar> = (<targetDataType>) <sourceValue>;

class Test{

public static void main(String args[]){

double d = 65.0;

// float f = d;

float f1 = (float)d; // Narrowing

// long data = f1;

long y = (long) f1; // Narrowing

//int a = y;

int x = (int) y; // Narrowing

//char c = x;

char ch = (char)x; // Narrowing

//short sh = x;

short s = (short)x; // Narrowing

//byte b1 = s;

byte b2 = (byte) s; // Narrowing

//byte b3 = d;

byte b4 = (byte)d; // Narrowing

System.out.println(d);

System.out.println(f1);

System.out.println(y);

System.out.println(x);

System.out.println(ch);

System.out.println(s);

System.out.println(b2);

System.out.println(b4);

}

}

class Test{

public static void main(String args[]){

// boolean b1 = 0;

// boolean b2 = (boolean)0;

// String s1 = 10;

// String s2 = (String)10;

}

}

Compound Assignment Operator

+=

-=

\*=

/=

%=

class Test{

public static void main(String args[]){

byte b1 = 14;

//byte b2 = -b1;

byte b3 = (byte) -b1;

System.out.println(b1);

System.out.println(b3);

byte b4 = 12;

//byte b5 = b1 + b4;

byte b6 =(byte) (b1 + b4);

System.out.println(b6);

int x = 10;

x = x+5;

System.out.println(x);

x+=5;

System.out.println(x);

int y = 10;

y = y \* 5 + 4;

System.out.println(y);

int z = 10;

z\*=5 + 4;

System.out.println(z);

byte by = 10;

// by = by + 5;

by+=5;

System.out.println(by);

}

}

Relational Operator (Binary)

>

<

>=

<=

  ==

!=

Result will be boolean

Operands will Numeric

== & != (Extra type boolean & Reference)

class Test{

public static void main(String args[]){

int x = 10;

int y = 20;

boolean b1 = x > y;

boolean b2 = y > x;

System.out.println(b1);

System.out.println(b2);

System.out.println(x >= y);

System.out.println(y >= x);

System.out.println(x < y);

System.out.println(y < x);

System.out.println(x <= y);

System.out.println(y <= x);

System.out.println("--");

System.out.println(10 == 10);

System.out.println(10 == 12);

System.out.println("--");

System.out.println(10 != 10);

System.out.println(10 != 12);

// System.out.println(true > true);

System.out.println(true == true);

//System.out.println("AB" > "AB");

System.out.println("AB" == "AB");

}

}

Logical Operator

&& AND Binary

|| OR Binary

!  NOT Unary

Operands are boolean

Result is also boolean

class Test{

public static void main(String args[]){

int sid = 99;

int phone = 9999;

boolean b1 = sid == 99;

boolean b2 = phone == 9999;

boolean b3 = sid == 99 && phone == 9999;

System.out.println("Logical AND");

System.out.println(b1 && b2);

System.out.println(b1 && b2);

System.out.println(true && true);

System.out.println(true && false);

System.out.println(false && true);

System.out.println(false && false);

System.out.println("Logical OR");

boolean b4 = sid == 99 || phone == 9999;

System.out.println(b1 || b2);

System.out.println(true || true);

System.out.println(true || false);

System.out.println(false || true);

System.out.println(false || false);

System.out.println("Logical NOT");

boolean b5 = !(sid == 99) || phone == 9999;

System.out.println(!false);

System.out.println(!true);

}

}

Conditional Operator

<booleanValue>? <value1> : <value2>

<value1> & <value2> must be assignment compatible

<booleanValue> is true then <value1> is returned

<booleanValue> is false then <value2> is returned

class Test{

public static void main(String args[]){

int x = true ? 10 : 20;

System.out.println(x);

int y = false ? 10 : 20;

System.out.println(y);

int a = 10;

int b = 20;

int max = a > b ? a : b;

System.out.println(max);

String s1 = true ? "Weekdays" : "Weekends";

System.out.println(s1);

String s2 = false ? "Weekdays" : "Weekends";

System.out.println(s2);

}

}

Increment (++) / Decrement (--) -> Unary

++ to increase the value of variable by 1

-- to decrease the value of variable by 1

Prefix (increased/decreased value will be used for expression)

++<variable>

--<variable>

Postfix (current/existing value will be used for expression)

<variable>++

<variable>--

class Test{

public static void main(String args[]){

int x = 10;

x++;

System.out.println(x);

int y = 10;

y--;

System.out.println(y);

int a = 10;

++a;

System.out.println(a);

int b = 10;

--b;

System.out.println(b);

}

}

=========

class Test{

public static void main(String args[]){

int x = 10;

System.out.println(x++ + " = " + x);

System.out.println(x);

int a = 10;

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

System.out.println(a);

}

}

=============

class Test{

public static void main(String args[]){

int x = 10;

int r1 = x++;

System.out.println(x);

System.out.println(r1);

int a = 10;

int r2 = ++a;

System.out.println(a);

System.out.println(r2);

}

}

class Test{

public static void main(String args[]){

int x = 10;

int r1 = x++ + 5;

System.out.println(x);

System.out.println(r1);

int a = 10;

int r2 = ++a + 5;

System.out.println(a);

System.out.println(r2);

}

}

Control Statement