Java Operators, Loops & Arrays - Nov 1, 2017 - Shen

Operators ---- Control Statement

data types

primitive and object

boolean, char -16, byte - 8 , short - 16, int- 32, long-64, float - 32, double -64

range  byte -2 power of 7 to 2 to power 8

Operators

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/opsummary.html>

precedence associativity

+ - ~ ++ ,  left to right or right left

1. Assignment operator   =    right to left

int a = 3;

Arithmetic operator

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

+ ,  - , \* , / , %

+   operator overloaded      adding two primitive datatype

     String concatenation

- subtraction

- multiplication

- / division

- % modulus - getting reminder

6/3 = 2

6%3 ----> 0

6%2 == 0  even or odd

a + b

Unary Operators

one operand

+  --------> int a = +5;

-  --------->  a = -5;

++  --------> increment operator

int x = 5;

x++  ---------> 6

postfix and prefix

postfix -->       a++

prefix   --->     ++a

int a = 9;            a++ / ++a ------> 10

when we use with other operator

    prefix

   int c = ++a;               ----------> c --- 10    a ---10

   first add one to a

   then assign it to c

postfix

int c = a++;

 it will first assign to c

 then add one to a

    c -----> 9         a ---->10

         1              2          4            from 4 to 2

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

{        3

   s.o.p ( "  " +  ++a);

}

o/p -> 1,   3,  5,

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

   s.o.p ( "  " +  a++ );

o/p-> 0, 2, 4

logical compliment operator ! - used for boolean

  boolean b = true;

!b

 boolean c = !b;

  sop ( "value of c" + c);

     if  !( num% 2 == 0)

**Equality and Relational Operators**

**all comparison operators return true or false;**

==      Equal to

compare two operands

 a == b       int a = 5 , b = 7;

--> returns false;

----------> for primitive data types check for value

Class Savings

Savings a , b;

a = new Savings();

//b = new Savings();

b = a ;

a == b

Only for values

!=      Not equal to

>       Greater than  
>=      Greater than or equal to  
<       Less than

<=      Less than or equal to

Conditional operators

 and &&

true  and true -> true

true  and false -> false

false  and false -> false

false and true -> false

or  || ---   only if both a false returns false others will be true

a = 5 b= 6 c= 9

if ( a== 5 || b == 6 || c == ++a )

ternary operator

(condition)? stt if true : stt if false;

String s = (a% 2 == 0)?"even" : "odd";

Nesting of ternary op is possible;

Type comparison

instanceof

 Parent - Account

 Child -  SavingsAccount extends Account

Parent - Credit

SavingsAccounts implements Credit

SavingsAccount  s = new SavingsAccount();

 if ( s instanceof Account) ---> yes

  boolean b = s instanceof Credit

b -> true;

interface

interface A

Class B implements A

Class C implements A

A  obj = new B();

A  obj1 = new C();

List list = new ArrayList();

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

Compound Operator

   += ,  -= , \*= , >>=, <<=, <<<=

   a = a+b;   a+=b

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

**Bitwise and Bit Shift Operators  for each bit**

~       Unary bitwise complement    tilde

<<      Signed left shift     the sign bit will be retained

>>      Signed right shift      the sign bit will be retained

a = 5

0000 0101

a= -5      it stores as 2's compliment of 5  (in Java -ve is stored as 2's compliment)

1111 1011

signed right shift retains 1 in the last msb

a>> 1     1011 1101

a>>> 1     0111 1101

>>>     Unsigned right shift

&       Bitwise AND

^       Bitwise exclusive OR   -   if both are same returns 0 if both are differ returns 1

|       Bitwise inclusive OR

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

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | sign- most significant bit |  |  |  |  |  |  |  |  |  |
| 2 to the powe of | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |  |
| place value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| a=5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 101 |  |
| b = 6 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 110 |  |
|  |  |  |  |  |  |  |  |  |  |  |
| ~a | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |  |  |
| -1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |  |
| a << 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | multiplies by 2 |  |
| a>>1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |  |  |
| a>>> 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |  |  |
| a=-5 |  |  |  |  |  |  |  |  | 101 | 10 |
|  |  |  |  |  |  |  |  |  | 1111010 | 01111101 |
| a = 2 |  |  |  |  |  |  | 1 |  |  |  |
| b = 6 |  |  |  |  |  | 1 | 1 | 0 |  |  |
| a & b |  |  |  |  |  | 0 | 1 | 0 | a = 2 |  |
| a | b |  |  |  |  |  | 1 | 1 | 0 | return the highest number of the 2 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| a = 2 |  |  |  |  |  | 0 | 1 | 0 |  |  |
| b = 6 |  |  |  |  |  | 1 | 1 | 0 |  |  |
| a ^ b |  |  |  |  |  | 1 | 0 | 0 | gives difference between two numbers |  |

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

**Java Operator Precedence Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Precedence** | **Operator** | **Type** | **Associativity** |
| 15 | () [] · | Parentheses Array subscript Member selection | Left to Right |
| 14 | ++ -- | Unary post-increment Unary post-decrement | Right to left |
| 13 | ++ -- + - ! ~ ( *type* ) | Unary pre-increment Unary pre-decrement Unary plus Unary minus Unary logical negation Unary bitwise complement Unary type cast | Right to left |
| 12 | \*  /  % | Multiplication Division Modulus | Left to right |
| 11 | + - | Addition Subtraction | Left to right |
| 10 | << >> >>> | Bitwise left shift Bitwise right shift with sign extension Bitwise right shift with zero extension | Left to right |
| 9 | < <= > >= instanceof | Relational less than Relational less than or equal Relational greater than Relational greater than or equal Type comparison (objects only) | Left to right |
| 8 | == != | Relational is equal to Relational is not equal to | Left to right |
| 7 | & | Bitwise AND | Left to right |
| 6 | ^ | Bitwise exclusive OR | Left to right |
| 5 | | | Bitwise inclusive OR | Left to right |
| 4 | && | Logical AND | Left to right |
| 3 | || | Logical OR | Left to right |
| 2 | ? : | Ternary conditional | Right to left |
| 1 | = += -= \*= /= %= | Assignment Addition assignment Subtraction assignment Multiplication assignment Division assignment Modulus assignment | Right to left |

*Larger number means higher precedence*.

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

**break; -----> in java shifts control out of the block**

**{**

**----------line1**

**-------line2**

**cond is satisfied then dont execute line 4 onwards**

**break;**

**-------line3**

**-----------line4**

**}**

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

**Control statement**

conditional & loops

conditional statements

1.  if ( condition ) { statement }

2.

if (cond)   {

    statemnts1

}

else {

   stt 2

}

3.  Nested if

   if () {

   }

   else if () { }

           else

  switch ( variable)

  {

     case   var1 :  stt1

     case   var2 :  stt2

     ..

     default :   default stt;

}

 variable - char, int, float, enum, double

int accounttype = Account.SAVINGS\_ACCOUNT;

switch (accounttype)

{

    case  Account.SAVINGS\_ACCOUNT :    savings.calcRate();

                                                                        break;

    case  Account.CHECKING\_ACCOUNT : checking.calcRate();

                                                                         break;

   default : sop ("account not matching");

}

for ( char i = 'a'; i <='z' ; i++)

{

     switch(i)

     {

          case 'a' :

           case 'e' :

           case 'i' :

           case 'o' :

           case 'u' :  vowel++; break;

           default : conson++;

       }

}

Loops

   do while, while, for

do {

        sttt;

    }while( condition);

int a = 11;

do {

    sop ( "a= " + a);

    a++;

   } while (a <= 10);

while(condition)

{

   statements;

}

  while( a <=10)

{

sop ( "a= " + a);

    a++;

}

          first            1                 3

for( initialize ;  condition ; inc /dec)

{      2

    statements;

}

continue ---- in java loops, continue keyword is used to move to next iteration in loops without executing the remaining code.

for each /  enum

package com.wbl.basic;

public class Operators {

public void printFizzBuzz() {

// divisible by 3 -- fizz

// 5 -- buzz

// 3 & 5 ---fizzbuzz -- assignment with switch

for(int a=1; a<=100; a++) {

if(a%3 == 0){

System.out.print("\t fizz");

}

else if (a%5==0) {

System.out.print("\t buzz");

}

else System.out.print("\t"+a);

if(a%10==0) System.out.println();

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

  new Operators().printFizzBuzz();

}

}

--------------------------ARRAYS --------------------------

Arrays - is collection of similar datatype

  int []

  String[]

  Account[]

  single dimensional

  multi dimensional

Arrays are references

String[]   names = {"rupa", "vidya", "pratibha"};

int[]  a = { 1, 2, 3, 4};

datatype[]    var = {value1, value2,  };

datatype  var[];

datatype   []var;

a[0] = 1;

a[3]= 4;

a.length;  returns length of the array

 for( int i = 0; i< a.length; i++)

   {

       sop ( " " + a[i]);

   }

pass by value and pass by reference

package com.wbl.basic;

public class ArrayExample {

public void printArray(int arr[])

{

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

System.out.println(" "+ arr[i]);

arr[3]= 100;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

  int a[] = {23, 45, 67, 90};

  ArrayExample arrExample = new ArrayExample();

  arrExample.printArray(a);

  System.out.println("after printArray");

  for(int i=0; i<a.length; i++)

  System.out.println(" "+ a[i]);

}

}

Assignment

1. Write a program to print 1 to 100

      print 'fizz' if divisible by 3

       print 'buzz' if divisible by 5

       print 'fizzbuzz' if divisible by 3 & 5

2.   print the below pattern

      \*

      \* \* \*

      \* \* \* \* \*

      \* \* \*

      \*

3.   Calculate the average of all elements in the integer array

4.   Find the highest and lowest value in an integer array

5.   In an online shopping site, the  order numbers are stored in an array in the same

      sequence in which they were received. Given the order number find the  sequence

      number in the array. (Basically, given the value find the index in an array)