សាកលវិទ្យាល័យ អាស៊ី អឺរ៉ុប

មហាវិទ្យាល័យវិទ្យាសាស្ត្រ និងបច្ចេកវិទ្យា

មុខវិជ្ជា ៖ Object Programming Language

មេរៀនទី៣ ៖ **ប្រតិបត្តិករ** (Operators)

សាស្ត្រាចារ្យ ៖ **ឈុំ ខឿន**

https://elearning.aeu.cloud

សាកលវិទ្យាល័យ អាស៊ី អឺរ៉ុប

មហាវិទ្យាល័យ វិទ្យាសាស្ត្រ និងបច្ចេកវិទ្យា

មុខវិជ្ជា៖

Object Programming Language

មេរៀនទី៣៖ ប្រតិបត្តិករ (Operators)

គណៈកម្មការអភិវឌ្ឍន៍កម្មវិធីសិក្សា

មាតិកា

ក្រោយពីបានសិក្សាជំពូកនេះ សិស្សានុសិស្សមានលទ្ធភាព៖

- ស្គាល់ និងចេះប្រើប្រាស់ប្រតិបត្តិករលេខនព្វន្តគណិត
- ស្គាល់ និងចេះប្រើប្រាស់ប្រតិបត្តិករលេខគោលពីរ
- ស្គាល់ និងចេះប្រើប្រាស់ប្រតិបត្តិកប្រៀបធៀប
- ស្គាល់ និងចេះប្រើប្រាស់ប្រតិបត្តិកប្រភេទតក្ក
- ដឹងអំពីអាទិភាពនៃប្រតិបត្តិកដែលត្រូវប្រើប្រាស់
- ចេះសរសេរកម្មវិធីចាវ៉ា ដោយប្រើប្រាស់ប្រតិបត្តិករលេខនព្វន្តគណិត ប្រតិបត្តិករលេខគោលពីរ ប្រតិបត្តិកប្រៀបធៀប និងប្រតិបត្តិកប្រភេទតក្ក

- ប្រតិបត្តិករ ត្រូវបានចែកចេញជាបួនក្រុម គឺ៖
  1. ប្រតិបត្តិករលេខនព្វន្តគណិត (Arithmetic Oparators)
  2. ប្រតិបត្តិករលេខគោលពីរ (Bitwise Operators)
  3. ប្រតិបត្តិកប្រៀបធៀប (Relational Operators)
  4. ប្រតិបត្តិកប្រភេទតក្ក (Boolean Logical Operators)
- ប្រតិបត្តិករផ្សេងៗមួយចំនួនទៀត ដែលសម្រាប់ប្រើប្រាស់នៅក្នុងស្ថានភាពពិសេសខ្លះៗ។

1. ប្រតិបត្តិករលេខនព្វន្តគណិត (Arithmetic Operators)

- ប្រតិបត្តិករលេខនព្វន្តគណិត ត្រូវបានប្រើនៅក្នុងកន្សោមគណិតវិទ្យា ដែលមានលក្ខណៈដូចនឹងការប្រើនៅក្នុងពិជគណិតដែរ។
- ខាងក្រោមនេះ គឺជាតារាងបង្ហាញពីប្រតិបត្តិករលេខនព្វន្តគណិត៖

ប្រតិបត្តិករ	គោលបំណង
+	បូក (Addition)
-	ដក (Subtraction)
*	គុណ (Multiplication)
/	ចែក (Divisioon)
%	ចែកយកសំណល់ (Modulus)
++	បង្កើនតម្លៃមួយឯកតា (Increment)

ប្រតិបត្តិករ	គោលបំណង
+=	កំណត់តម្លៃនៃផលបូក (Additional Assignment)
-=	កំណត់តម្លៃនៃផលដក (Subtraction Assignment)
*=	កំណត់តម្លៃនៃផលគុណ (Multiplication Assignment)
/=	កំណត់តម្លៃនៃផលចែក (Division Assignment)
%=	កំណត់តម្លៃនៃផលចែកយកសំណល់ (Modulus Assignment)
--	បន្ថយតម្លៃមួយឯកតា (Decrement)

- អង្គប្រមាណវិធី (Operands) របស់ប្រតិបត្តិករលេខនព្វន្តគណិត ត្រូវតែមានប្រភេទជាតម្លៃលេខ។
- យើងមិនអាចយកប្រតិបត្តិករលេខនព្វន្តគណិតទៅប្រើជាមួយនឹងប្រភេទទិន្នន័យជា boolean បានឡើយ ប៉ុន្តែយើងក៏អាចប្រើពួកវាទៅលើប្រភេទទិន្នន័យជា char បានដែរ ពីព្រោះប្រភេទ char នៅក្នុងចាវ៉ា គឺជាផ្នែកមួយនៃ int ដែរ។

1.1. ប្រតិបត្តិករលេខនព្វន្តគណិតមូលដ្ឋាន (Basic Arithmetic Operators)

- ប្រតិបត្តិករលេខនព្វន្តគណិតសំខាន់ៗ រួមមាន៖ បូក, ដក, គុណ និងចែក។
- ប្រតិបត្តិករលេខនព្វន្តគណិតទាំងនោះ អនុវត្តទៅលើប្រភេទទិន្នន័យជាតម្លៃលេខ។

- ប្រតិបត្តិករលេខនព្វន្តដក (Subtraction) ក៏មានទម្រង់បែប Unary (ប្រើតែមួយអង្គប្រមាណវិធី) បានដែរ ដែលផ្តល់តម្លៃអវិជ្ជមាន នៅពេលប្រើទៅលើអង្គប្រមាណវិធីរបស់វា។
- នៅពេលប្រតិបត្តិករលេខនព្វន្តចែក ត្រូវបានប្រើទៅលើប្រភេទចំនួនគត់ (Integer) ពេលនោះលទ្ធផល ដែលទទួលបាននៅតែជាចំនួនគត់ ផ្នែកដែលនៅខាងក្រោយក្បៀសនឹងត្រូវបាត់បង់។
- កម្មវិធីខាងក្រោម បង្ហាញពីការប្រើប្រតិបត្តិករលេខនព្វន្តគណិត។ វាក៏បង្ហាញពីភាពខុសគ្នារវាងការប្រើចំនួនទសភាគ និងចំនួនគត់ដែរ៖

```
public class BasicMath {
    public static void main(String[] args) {
        // Arithmetic using integers
        System.out.println("Integer Arithmetic");
        int a = 1 + 1;
        int b = a * 3;
```

```
        int c = b / 4;
        int d = -c;
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
        System.out.println("d = " + d);
        // Arithmetic using double
        System.out.println("\nfloating-point Arithmetic");
        double da = 1 + 1;
        double db = da * 3;
        double dc = db - da;
        double dd = -dc;
        System.out.println("da = " + da);
        System.out.println("db = " + db);
        System.out.println("dc = " + dc);
        System.out.println("dd = " + dd);
    }
}
```

1.2. ប្រតិបត្តិករលេខនព្វន្តចែកយកសំណល់ (Modulus Operators)

- ប្រតិបត្តិករលេខនព្វន្តចែកយកសំណល់(%) នឹងទទួលបានលទ្ធផលដែលជាសំណល់របស់ប្រមាណវិធីចែក។
- វាអាចប្រើជាមួយនឹងប្រភេទចំនួនទសភាគក៏បាន និងប្រភេទចំនួនគត់ក៏បាន។
- ឧទាហរណ៍៖

```
public class Mudulus {
    public static void main(String[] args) {
        int x = 42;
        double y = 42.5;
        System.out.println("x mod 10 = " + x%10);
        System.out.println("y mod 10 = " + y%10);
    }
}
```

1.3. ប្រតិបត្តិករលេខនព្វន្តគណិតកំណត់តម្លៃ (Arithmetic Assignment Operators)

- ចាវ៉ាមានប្រតិបត្តិករលេខនព្វន្តពិសេសៗមួយចំនួន ដែលត្រូវបានប្រើដោយប្រើប្រាស់ប្រតិបត្តិករលេខនព្វន្តគណិតបន្តដោយសញ្ញាកំណត់តម្លៃពីក្រោយ។ ដូច្នេះ សម្រាប់ឃ្លាណាមួយដែលមានទម្រង់៖

  var = var op expression; អាចសរសេរជា *var op= expression;*

  - ឧទាហរណ៍៖ a = a + 4; ក្នុង Java អាចសរសេរជា a += 4;
  - ឧទាហរណ៍៖ a = a % 2; ក្នុង Java អាចសរសេរជា a %= 2;

- ប្រតិបត្តិករលេខនព្វន្តគណិតកំណត់តម្លៃ ផ្តល់ផលប្រយោជន៍ពីរ៖
  - ទី១៖ ធ្វើឱ្យយើងសន្សំសំចៃនូវចំនួន bits ពីព្រោះវាជាការសរសេរទម្រង់ខ្លី ដែលសមមូលនឹងទម្រង់វែងរបស់វា។
  - ទី២៖ ធ្វើឱ្យប្រព័ន្ធប្រតិបត្តិការអនុវត្តលឿនជាងទម្រង់វែងរបស់វា។

- កម្មវិធីខាងក្រោមនេះ បង្ហាញពីការប្រើប្រតិបត្តិករលេខនព្វន្តគណិតកំណត់តម្លៃ op= មួយចំនួន៖

```
public class OpEquals {
    public static void main(String[] args) {
        int a = 1;
        int b = 2;
        int c = 3;

        a += 5;
        b *= 4;
        c %= 2;
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
    }
}
```

1.4. ការបង្កើន និងបន្ថយតម្លៃមួយឯកតា (Increment and Decrement)

- សញ្ញា ++ ឬ -- ជាប្រតិបត្តិករលេខនព្វន្តគណិតសម្រាប់បង្កើន ឬបន្ថយតម្លៃមួយឯកតានៃអង្គប្រមាណវិធីមួយ។ ការសរសេរខាងក្រោមសមមូលគ្នា៖

  សម្រាប់ឃ្លា x = x + 1; អាចសរសេរជា x++;

  ចំណែកឃ្លា x = x - 1; អាចសរសេរជា x--;

- ++ ឬ -- អាចប្រើដាក់ពីមុខ(Prefix) ឬពីក្រោយ(Postfix) នៃតួលេខរបស់វា។

- បើដាក់ពីមុខ នោះតួលេខត្រូវបានបង្កើន ឬបន្ថយតម្លៃ 1 មុនពេលអង្គប្រមាណវិធីនោះត្រូវបានអនុវត្ត។ ប៉ុន្តែ បើដាក់ពីក្រោយអង្គប្រមាណវិធី នោះត្រូវបានអនុវត្តសិន ទើបតម្លៃរបស់វាត្រូវបានបង្កើន ឬបន្ថយចំនួន 1 ជាក្រោយ។

- កម្មវិធីខាងក្រោម បង្ហាញពីការប្រើប្រតិបត្តិករលេខនព្វន្តគណិតបង្កើន និងបន្ថយតម្លៃមួយឯកតា៖

```
public class IncDec {
    public static void main(String[] args) {
        int a = 2;
        int b = 4;
        int c, d;

        c = ++a;
        d = b--;
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
        System.out.println("d = " + d);
    }
}
```

```
Result:
-------
 a = 3
 b = 3
 c = 3
 d = 4
```

2. ប្រតិបត្តិករលេខគោលពីរ (Bitwise Operators)

- ចាវ៉ា កំណត់នូវប្រតិបត្តិករលេខគោលពីរមួយចំនួន ដែលអាចអនុវត្តបានចំពោះគ្រប់ប្រភេទទិន្នន័យចំនួនគត់ (byte, short, int, long) ព្រមទាំងចំពោះ char ផងដែរ។
- ប្រតិបត្តិករលេខនព្វន្តទាំងនេះ សម្រាប់ប្រើដោយអាស្រ័យទៅលើ bit នីមួយៗ នៃអង្គប្រមាណវិធីរបស់វា។

សញ្ញាណលេខនព្វន្ត	គោលបំណង
~	Bitwise Unary NOT
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR (Exclusive OR)
>>	Shift Right

ប្រតិបត្តិករ	គោលបំណង
<<	Shift Left
&=	Bitwise AND Assignment
|=	Bitwise OR Assignment
^=	Bitwise XOR Assignment
>>=	Shift Right Assignment
<<=	Shift Left Assignment

- ប្រតិបត្តិករលេខគោលពីរ ប្រើ bits ក្នុងទម្រង់ជាចំនួនគត់។
- គ្រប់ប្រភេទទិន្នន័យជាចំនួនគត់ទាំងអស់(លើកលែងតែ char) គឺជាចំនួនគត់ដែលតំណាងឱ្យទាំងតម្លៃវិជ្ជមាន និងតម្លៃអវិជ្ជមាន។

- ចាវ៉ា ប្រើការបំប្លែងកូដដា two's complement មានន័យថា ចំនួនអវិជ្ជមានត្រូវបានបង្ហាញឡើង ដោយការសរសេរបញ្ច្រាស(ការប្តូរពីលេខ 1 ទៅលេខ 0 និងផ្ទុយមកវិញ) នូវគ្រប់ bits ទាំងអស់នៅក្នុងតម្លៃលេខ ហើយបន្ទាប់មកបូកបន្ថែម 1 ទៅឱ្យលទ្ធផល។

  ឧទាហរណ៍៖ -42 ត្រូវតំណាងឱ្យការសរសេរបញ្ច្រាសគ្រប់ bits ទាំងអស់របស់ 42 ឬ 00101010 ដែលទទួលបាន 11010101 បន្ទាប់មកបូកថែម 1 ដូច្នេះយើងបានលទ្ធផល 11010110 ឬ -42 ។

- ផ្ទុយមកវិញ យើងក៏អាចធ្វើពីចំនួនអវិជ្ជមានទៅចំនួនវិជ្ជមានបានដែរ គឺជាដំបូងត្រូវសរសេរបញ្ច្រាសគ្រប់ bits ទាំងអស់របស់ចំនួនអវិជ្ជមាននោះ ហើយបន្ទាប់មកបូកថែម 1 ។

- ឧទាហរណ៍៖ -42 ឬ 11010110 ត្រូវបានសរសេរបញ្ច្រាសទៅជា 00101001 ឬ 41 ដូច្នេះ ពេលយើងបូក 1 នោះយើងទទួលបាន 42 ។

2..1. ប្រតិបត្តិករលេខគោលពីរបែបតក្ក (Boolean Logical Bitwise Operators)

- ប្រតិបត្តិករលេខគោលពីរបែបតក្ក រួមមាន &, | , ^ និង ~ ។

តារាងខាងក្រោម បង្ហាញពីការប្រើប្រតិបត្តិករលេខគោលពីរបែបតក្ក៖

A	B	A | B	A & B	A ^ B	~A
0	0	0	0	0	1
1	0	1	0	1	0
0	1	1	0	1	1
1	1	1	1	0	0

2..2. ប្រតិបត្តិករលេខគោលពីរឈ្នាប់មិន (Bitwise NOT)

- ប្រតិបត្តិករលេខគោលពីរឈ្នាប់មិន (~) ជាប្រតិបត្តិករប្រភេទ Unary NOT ឬ ហៅម្យ៉ាងទៀតថា Bitwise complement ដែលលទ្ធផលរបស់វា គឺជាការសរសេរបញ្ច្រាសនូវគ្រប់ Bits នៃអង្គប្រមាណវិធីរបស់វា។

ឧទាហរណ៍ Bitwise NOT៖

```
~ 0 0 1 0 1 0 1 0
-----------------
1 1 0 1 0 1 0 1
```

2.3. ប្រតិបត្តិករលេខគោលពីរឈ្នាប់និង (Bitwise AND)

- ប្រតិបត្តិករលេខគោលពីរឈ្នាប់និង (&) ផ្តល់លទ្ធផល 1 កាលណាតួលេខទាំងពីរជាលេខ 1 ដូចគ្នា។ ប៉ុន្តែ បើមានតួលេខមួយណា ឬតួលេខទាំងពីរជាលេខ 0 នោះលទ្ធផលដែលទទួលបានគឺលេខ 0 ។

ឧទាហរណ៍ Bitwise AND៖

```
  0 0 1 0 1 0 1 0      42
& 0 0 0 0 1 1 1 1      15
  ---------------
  0 0 0 0 1 0 1 0      10
```

2.4. ប្រតិបត្តិករលេខគោលពីរឈ្នាប់ឬ (Bitwise OR)

- ប្រតិបត្តិករលេខគោលពីរឈ្នាប់ឬ (|) ផ្តល់លទ្ធផល 1 កាលណាគូលេខទាំងពីរ ឬគូលេខណាមួយជាលេខ 1 ។ ប៉ុន្តែ បើគូលេខទាំងពីរជាលេខ 0 ដូចគ្នានោះលទ្ធផលដែលទទួលបានគឺលេខ 0 ។

ឧទាហរណ៍ Bitwise OR៖

```
  0 0 1 0 1 0 1 0      42
| 0 0 0 0 1 1 1 1      15
 -----------------
  0 0 1 0 1 1 1 1      47
```

2..5. ប្រតិបត្តិករលេខគោលពីរឈ្នាប់ XOR (Bitwise XOR)

- ប្រតិបត្តិករលេខគោលពីរឈ្នាប់ XOR (^) ផ្តល់លទ្ធផលលេខ 1 កាលណាតួលេខណាមួយជាលេខ 1 និងតួលេខមួយផ្សេងទៀតគឺជាលេខ 0 ។ ប៉ុន្តែ បើតួលេខទាំងពីរជាលេខ 1 ដូចគ្នា ឬលេខ 0 ដូចគ្នា នោះលទ្ធផលដែលទទួលបានគឺលេខ 0 ។

ឧទាហរណ៍ Bitwise XOR (Exclusive OR) ៖

```
  0 0 1 0 1 0 1 0      42
^ 0 0 0 0 1 1 1 1      15
  ----------------
  0 0 1 0 0 1 0 1      37
```

2.6. ការប្រើប្រតិបត្តិករលេខគោលពីរបែបតក្ក (Using the Bitwise Logical Operators)

- ឧទាហរណ៍ខាងក្រោម បង្ហាញពីការប្រើប្រតិបត្តិករលេខគោលពីរបែបតក្ក៖

```
// Demonstrate the bitwise logical operators.
class BitLogic {
    public static void main(String args[]) {
        String binary[] = {
                          "0000", "0001", "0010", "0011",
                          "0100", "0101", "0110", "0111",
                          "1000", "1001", "1010", "1011",
                          "1100", "1101", "1110", "1111",
                          };
```

```
        int a = 3;  // 0011 in binary
        int b = 6;  // 0110 in binary

        int c = a | b;
        int d = a & b;
        int e = a ^ b;
        int f = (~a & b) | (a & ~b);
        int g = ~a & 0x0f;

        System.out.println("    a = " + binary[a]);
        System.out.println("    b = " + binary[b]);
        System.out.println("    c = " + binary[c]);
        System.out.println("    d = " + binary[d]);
        System.out.println("    e = " + binary[e]);
        System.out.println("    f = " + binary[f]);
        System.out.println("    g = " + binary[g]);
    }
}
```

Problems @ Javadoc Declaration Console

<terminated> BitLogic [Java Application] C:\1. KHOEUN'S\eclipse

```
a = 0011
b = 0110
c = 0111
d = 0010
e = 0101
f = 0101
g = 1100
```

2.7. ប្រតិបត្តិករអំពីលេងឆ្វេង (Left Shift Operators)

- ប្រតិបត្តិករអំពីលេងឆ្វេង (<<) នឹងអំពីលគ្រប់ចំនួន Bit ដែលបានកំណត់ឱ្យ នៅក្នុងតម្លៃលេខណាមួយទៅផ្នែកខាងឆ្វេង។

វាមានទម្រង់៖

value << num

ក្នុងនេះ num បញ្ជាក់នូវចំនួនខ្ទង់ដែលត្រូវអំពីលទៅខាងឆ្វេងនៃតម្លៃលេខនៅក្នុង value ។ មានន័យថា << អំពីលគ្រប់ Bits នៅក្នុងតម្លៃលេខដែលបានកំណត់ទៅផ្នែកខាងឆ្វេងតាមរយៈចំនួនទីតាំងរបស់ Bits ដែលត្រូវកំណត់ដោយ num ។

- ចំពោះការរំកិលទៅផ្នែកខាងឆ្វេង គឺធ្វើឱ្យ Bit លំដាប់ខ្ពស់ត្រូវបានបំបាត់ចោល ប៉ុន្តែ ត្រូវថែម 0 នៅផ្នែកខាងស្តាំ។
- ប៉ុន្តែ នៅពេលដែលការរំកិលទៅខាងឆ្វេងត្រូវបានធ្វើឡើងចំពោះតួលេខ ដែលមានប្រភេទជា int នោះចំនួន Bits ត្រូវបានបំបាត់ចោល កាលណាគេបានរំកិលផុតពី Bit ត្រង់ទីតាំង 31 ។ ប្រសិនបើតួលេខមានប្រភេទជា long នោះចំនួន Bits ត្រូវបានបំបាត់ចោល កាលណាគេបានរំកិលផុតពី Bit ត្រង់ទីតាំង 63 ។
- ការបង្កើនប្រភេទទិន្នន័យដោយស្វ័យប្រវត្តិ, Java នឹងផ្តល់លទ្ធផលមិនដូចដែលយើងគិតទុកនោះទេ កាលណាយើងរំកិលតម្លៃដែលមានប្រភេទជា byte និង short ។

- តម្លៃអវិជ្ជមានរបស់ byte និង short មួយ នឹងត្រូវរក្សាសញ្ញា នៅពេលដែលវាត្រូវបានបង្កើនឱ្យទៅជា int ។ ដូចនេះ Bits លំដាប់ខ្ពស់នឹងត្រូវបំពេញដោយលេខ 1 ។ ហេតុនេះហើយ ដើម្បីរំកិលឆ្វេងលើតម្លៃលេខប្រភេទ byte ឬ short នោះយើងត្រូវតែបំបាត់ចោលនូវ Bits លំដាប់ខ្ពស់ នៃលទ្ធផលដែលមានប្រភេទជា int ។
- ខាងក្រោមនេះ គឺជាកម្មវិធីដែលបង្ហាញពីការរំកិលឆ្វេង៖

```
//Left shifting a byte value.
class ByteShift {
    public static void main(String args[]) {
        byte a = 64, b;
        int i;
        i = a << 2;
        b = (byte) (a << 2);
        System.out.println("Original value of a : " + a);
        System.out.println("i and b : " + i + " " + b);
    }
}
```

- កម្មវិធីខាងលើផ្តល់លទ្ធផលដូចខាងក្រោមនេះ៖

```
Original value of a : 64
i and b : 256 0
```

- ដោយ a ត្រូវបានដំឡើងឱ្យទៅជា int ក្នុងគោលបំណងធ្វើការវាយតម្លៃ ដូច្នេះការរំកិលមកឆ្វេងនូវតម្លៃ 64 (0100 0000) ២ដង ចេញលទ្ធផល i នូវតម្លៃ 256 (1 0000 0000) ។ ចំណែក b មានតម្លៃ 0 ព្រោះក្រោយពេលរំកិល Bit លំដាប់ទាបមានតម្លៃ 0។
- ដោយហេតុថា ការរំកិលឆ្វេងម្តងៗ គឺជាទ្វេគុណនៃតម្លៃដើម ជាហេតុធ្វើឱ្យអ្នកសរសេរកម្មវិធីប្រើនូវវិធីមួយ គឺយកតម្លៃដើម គុណនឹង២ ។
- ប៉ុន្តែត្រូវប្រុងប្រយ័ត្ន ប្រសិនបើរំកិល 1 bit នៅទីតាំងកម្រិតខ្ពស់ (ត្រង់ bit 31 ឬត្រង់ 63) លទ្ធផលដែលទទួលបាន គឺជាតម្លៃអវិជ្ជមាន។

- សូមពិនិត្យមើលកម្មវិធីខាងក្រោមនេះ៖

```
//Left shifting as a quick way to multiply by 2.
class MultByTwo {
    public static void main(String args[]) {
        int i;
        int num = 0xFFFFFFE;
        for (i = 0; i < 4; i++) {
            num = num << 1;
            System.out.println(num);
        }
    }
}
```

- កម្មវិធីខាងលើផ្តល់លទ្ធផលដូចខាងក្រោមនេះ៖

```
536870908
1073741816
2147483632
-32
```

- តម្លៃចាប់ផ្តើមនៃកម្មវិធីខាងលើ ត្រូវបានជ្រើសមកប្រើដោយហ្មត់ចត់ ដើម្បីឱ្យក្រោយពីការរំកិលទៅឆ្វេងត្រង់ខ្ទង់ Bit ទី ៤ វានឹងផ្តល់ -32 ។
- ដូចបានឃើញនៅក្នុងលទ្ធផលហើយថា កាលណា 1 bit ត្រូវរំកិលទៅដល់ bit 31 លទ្ធផលដែលទទួលបានគឺជាចំនួនអវិជ្ជមាន។

2.8. ប្រតិបត្តិកររំកិលស្តាំ (Right Shift Operators)

- ប្រតិបត្តិកររំកិលស្តាំ (>>) នឹងរំកិលគ្រប់ចំនួន Bit ដែលបានកំណត់ឱ្យនៅក្នុងតម្លៃលេខណាមួយទៅផ្នែកខាងស្តាំ។ វាមានទម្រង់៖

```
value >> num
```

ក្នុងនេះ num បញ្ជាក់នូវចំនួនខ្ទង់ដែលត្រូវរំកិលទៅខាងស្តាំនៃតម្លៃលេខនៅក្នុង value ។ មានន័យថា >> រំកិលគ្រប់ Bits នៅក្នុងតម្លៃលេខដែលបានកំណត់ទៅផ្នែកខាងស្តាំតាមរយៈចំនួនទីតាំងរបស់ Bits ដែលត្រូវកំណត់ដោយ num ។

- ខាងក្រោមនេះ គឺជាកូដ ដែលត្រូវរំកិលចំនួន ២ខ្ទង់ទៅខាងស្តាំ នៅក្នុងតម្លៃលេខ 32 ហើយលទ្ធផល ក្រោយពីរំកិលគឺ 8 ។

```
int a = 32;

a = a >> 2; // a now contains 8
```

កាលណាតម្លៃមួយមាន Bits ដែលត្រូវបានរំកិលផុត ចំនួន Bits ទាំង នោះត្រូវបាត់បង់។

- ខាងក្រោម គឺជាកូដមួយទៀត ដែលត្រូវរំកិលចំនួន ២ខ្ទង់ ទៅខាងស្តាំ នៅក្នុងតម្លៃលេខ 35 ដែលធ្វើឱ្យ 2 bits លំដាប់ទាបត្រូវបាត់បង់ ហើយលទ្ធផលក្រោយពីរំកិល គឺ 8 ។

```
int a = 35;

a = a >> 2;      // a still contains 8
```

- សូមពិនិត្យមើលការអនុវត្តដោយប្រើ Binary ៖

  00100011 //35

  >>2

  00001000 //8

  នៅពេលយើងរំកិលម្តងៗ នៅក្នុងតម្លៃណាមួយទៅផ្នែកខាងស្តាំ វានឹងចែកតម្លៃនោះនឹង២។

- នៅពេលដែលយើងធ្វើការរំកិលស្តាំទៅលើចំនួនអវិជ្ជមាន នោះនឹងមានការបន្ថែមសញ្ញា សម្រាប់តម្រូវទៅនឹងចំនួនអវិជ្ជមាននោះ។

ឧទាហរណ៍៖

-8 >> 1 គឺ 4 ហើយបើគិតនៅក្នុង Binary គឺ៖

11111000 -8

>>1

11111100 -4

2.9. ការកំណត់តម្លៃប្រតិបត្តិករលេខគោលពីរ

- រាល់ប្រតិបត្តិករលេខគោលពីរទាំងអស់ មានទម្រង់អក្សរកាត់ស្រដៀងប្រតិបត្តិករលេខនព្វន្តគណិតដែរ ដែលត្រូវផ្សំគ្នារវាងសញ្ញាកំណត់តម្លៃជាមួយនឹងប្រតិបត្តិករលេខគោលពីរ។
- ឃ្លាខាងក្រោមនេះ សមមូលគ្នា៖

  a = a >> 4;

  a >>= 4;

- កម្មវិធីខាងក្រោម មានការប្រកាសអថេរជាប្រភេទទិន្នន័យចំនួនគត់ បន្ទាប់មក ប្រើទម្រង់សរសេរកាត់របស់ការកំណត់តម្លៃប្រតិបត្តិករលេខគោលពី ដើម្បីអនុវត្តទៅលើអថេរទាំងនោះ៖

```
public class OpBitEquals {
    public static void main(String[] args) {
        int a = 1;
        int b = 2;
        int c = 3;
        a |= 4;
        b >>= 1;
        c <<= 1;
        a ^= c;
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
    }
}
```

3. ប្រតិបត្តិករប្រៀបធៀប (Relational Operators)

- ប្រតិបត្តិកររប្រៀបធៀប កំណត់នូវទំនាក់ទំនងរវាងអង្គប្រមាណវិធីមួយទៅនឹងអង្គប្រមាណវិធីមួយផ្សេងទៀត។ ជាពិសេស វាកំណត់នូវភាពស្មើគ្នា ឬភាពតូចជាង ឬធំជាងនៃអង្គប្រមាណវិធីទាំងពីរ។ ខាងក្រោមនេះ បង្ហាញពីប្រតិបត្តិកររប្រៀបធៀប៖

ប្រតិបត្តិករ	គោលបំណង
==	ស្មើគ្នានឹង (Equal to)
!=	មិនស្មើគ្នានឹង (Not equal to)
>	ធំជាង (Greater than)
<	តូចជាង (Less than)
>=	ធំជាង ឬស្មើ (Greater than or equal to)
<=	តូចជាង ឬស្មើ (Less than of equal to)

- លទ្ធផលរបស់ប្រតិបត្តិករទាំងនេះ គឺជាតម្លៃតក្ក (boolean) ។ គេច្រើនប្រើពួកវានៅក្នុង if ឬ Loops ជាដើម។
- កូដខាងក្រោមនេះ ប្រើប្រតិបត្តិករប្រៀបធៀប និងផ្តល់លទ្ធផលជាតម្លៃតក្ក (boolean) ៖

```
int a = 4;
int b = 1;
boolean c = a < b;
```

លទ្ធផលរបស់ a < b ដែលបានពីកូដខាងលើនេះ គឺ false ហើយត្រូវបានផ្ទុកនៅក្នុង c ។

- ប្រសិនបើអ្នកធ្លាប់បានសិក្សា C/C++ សូមកំណត់ចំណាំឃ្លា៖

```
int done;
if( !done )…      // valid in C/C++
if( done )…..     // but not valid in Java
```

- នៅក្នុងចាវ៉ា សម្រាប់ឃ្លាដូចខាងលើយើងត្រូវសរសេរ៖

```
if( done == 0 )...  // This is Java style

if( done != 0 )....
```

នេះជាហេតុផលមួយ ដែលថា Java មានការប្រើប្រាស់ true និង false ខុសពី C/C++ ។

- នៅក្នុង C/C++, true គឺជាតម្លៃមិនសូន្យ (Nonzero) ហើយ false គឺជាតម្លៃសូន្យ (Zero)។
- រីឯនៅក្នុងចាវ៉ា true ឬ false គឺជាតម្លៃមិនមែនលេខ ដែលមិនពាក់ព័ន្ធនឹងសូន្យ ឬមិនសូន្យទេ។
- ដូច្នេះ ដើម្បីធ្វើងផ្ទាត់លេខសូន្យ ឬលេខមិនសូន្យ នោះយើងត្រូវប្រើប្រតិបត្តិករប្រៀបធៀបមួយ ឬច្រើនដោយបញ្ជាក់ឱ្យបានច្បាស់។

4. ប្រតិបត្តិករប្រភេទតក្ក (Boolean)

- ប្រតិបត្តិកប្រភេទតក្ក (Boolean) ដែលត្រូវលើកយកមកបង្ហាញនៅទីនេះ ធ្វើប្រមាណវិធីនៅលើតែអង្គប្រមាណវិធីដែលមានប្រភេទតក្ក (Boolean) ប៉ុណ្ណោះ។
- គ្រប់ប្រតិបត្តិករលេខគោលពីរបែបតក្កទាំងអស់ ត្រូវបានផ្សំគ្នានូវតម្លៃប្រភេទជាតក្ក (Boolean) ពីរ ដើម្បីបង្កើតលទ្ធផលរបស់វាជាតម្លៃប្រភេទតក្ក (Boolean) ៖

ប្រតិបត្តិករ	គោលបំណង
&	Logical AND
|	Logical OR
^	Logical XOR (Exclusive OR)
||	Short-circuit OR

ប្រតិបត្តិករ	គោលបំណង
&&	Short-circuit AND
!	Logical unary NOT
&=	AND assignment
|=	OR assignment
^=	XOR assignment
==	Equal to
!=	Not equal to
?:	Ternary *if-then-else*

- ប្រតិបត្តិករបែបតក្កប្រភេទ Boolean (&, | និង ^) នឹងធ្វើប្រមាណវិធីលើតម្លៃ boolean ដូចវាបានធ្វើប្រមាណវិធីនៅលើ bits របស់ចំនួនគត់ដែរ។
- ប្រតិបត្តិករបែបតក្ក NOT (!) គឺជាការសរសេរបញ្ច្រាសគ្នារវាង true និង false គឺ៖

  !true == false និង !false == true

- តារាងខាងក្រោម បង្ហាញពីលទ្ធផលនៃការប្រើប្រតិបត្តិករបែបតក្ក៖

A	B	A|B	A&B	A^B	!A
false	false	false	false	false	true
true	false	true	false	true	false
false	true	true	false	true	true
true	true	true	true	false	false

- កម្មវិធីខាងក្រោមនេះ បង្ហាញពីការប្រើប្រតិបត្តិករបែបតក្កប្រភេទ Boolean ៖

```
public class BoolLogic {
    public static void main(String[] args) {
        boolean a = true;
        boolean b = false;
        boolean c = a | b;
        boolean d = a & b;
        boolean e = a ^ b;
        boolean f = (!a & b) | (a & b);
        boolean g = !a;
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
        System.out.println("d = " + d);
        System.out.println("e = " + e);
        System.out.println("f = " + f);
        System.out.println("g = " + g);
    }
}
```

4.1. ប្រតិបត្តិករបែបតក្ក Short-circuit

- ចារ៉ា មានប្រតិបត្តិករ ដែលគួរឱ្យកត់សំគាល់ពីរ ដែលមានប្រភេទជា Boolean ដែលមិនសូវមានប្រើ នៅក្នុងភាសាកុំព្យូទ័រផ្សេងទៀយ ។
- ប្រតិបត្តិករទាំងពីរនោះ គឺ Short-circuit OR (||) និង Short-circuit AND (&&) ដែលជា Version បន្ទាប់ពីប្រតិបត្តិករបែបតក្ក AND (&) និង OR(|) ។
- បើយើងប្រើទម្រង់ Short-circuit OR(||) និង Short-circuit AND (&&) ជំនួសឱ្យការប្រើទម្រង់ | និង & ជាហេតុធ្វើឱ្យកម្មវិធី Java មិនមានការរំខានដល់ការរាយតម្លៃចំពោះតួលេខនៅខាងស្តាំទៀយ នៅពេលដែលលទ្ធផលនៃកន្សោមអាចត្រូវបានកំណត់ដោយតួលេខតែមួយ។
- ករណីនេះ មានប្រយោជន៍ណាស់ នៅពេលតួលេខខាងស្តាំពឹងផ្អែកតួលេខខាងឆ្វេង ដែល true ឬ false ដើម្បីកំណត់តួនាទីបានត្រឹមត្រូវ។

- សូមពិនិត្យមើលកូដខាងក្រោម ដែលបង្ហាញពីផលប្រយោជន៍នៃការប្រើ Short-circuit ៖

  if (denom!=0 && num/denom>10)

- ដោយប្រើទម្រង់ Short-circuit របស់ AND(&&) ជាហេតុធ្វើឱ្យកម្មវិធីគ្មាន Error នៅពេល Run កាលណាតម្លៃ denom = 0 ។ ប៉ុន្តែ បើប្រើ & វានឹងអនុវត្តទាំងពីរផ្នែក ពេលនោះវានឹងកើតមាន Error នៅពេល Run កាលណាតម្លៃ denom = 0 ។
- ដូច្នេះគួរតែប្រើទម្រង់ Short-circuit OR (||) និង Short-circuit AND (&&) ក្នុងករណីដែលមានការប្រើជាមួយនឹង Boolean Logic ។

4.2. ប្រតិបត្តិករលក្ខខណ្ឌ Ternary (?:)

- ចាវ៉ា មានប្រតិបត្តិករលក្ខខណ្ឌ Ternary (three-way) ពិសេសមួយ អាចប្រើជំនួសឃ្លា if-then-else បាន គឺ **?:** ប្រើដូចក្នុង C/C++ ដែរ។
- ទម្រង់ទូទៅរបស់ប្រតិបត្តិករលក្ខខណ្ឌ Ternary (**?:**) គឺ៖

condition ? result_1 : result_2

condition នៅទីនេះ គឺជាកន្សោមឃ្លាមួយ ដែលមានតម្លៃជា boolean ។ បើ *condition* មានតម្លៃ true ពេលនោះ *result_1* ត្រូវបានអនុវត្ត តែបើពុំដូច្នោះទេ *result_2* ត្រូវបានអនុវត្ត។

- លទ្ធផលរបស់ **?:** គឺជាតម្លៃនៃកន្សោមឃ្លាដែលត្រូវបានអនុវត្ត។

ឧទាហរណ៍៖

```
radio = denom == 0 ? 0 : num/denom;
```

- កម្មវិធីខាងក្រោមនេះ បង្ហាញពីការប្រើប្រតិបត្តិករលក្ខខណ្ឌ ?: ដើម្បីរកតម្លៃដាច់ខាត៖

```
// Demonstrate ?:
public class Ternary {
    public static void main(String[] args) {
        int i, k;

        i = 10;
        k = i < 0 ? -i : i; // get absolute value of i
        System.out.println("Absolut value of ");
        System.out.println(i + " i " + k);

        i = -10;
        k = i < 0 ? -i : i; // get absolute value of i
        System.out.println("Absolute value of ");
        System.out.println(i + " is " + k);
    }
}
```

5. អាទិភាពនៃប្រតិបត្តិករ

- ខាងក្រោមនេះ បង្ហាញពីអាទិភាពនៃប្រតិបត្តិករក្នុងភាសាចាវ៉ា៖

Highest

( )	[]	.	
++	--	~	!
*	/	%	
>>	<<		
>	>=	<	<=
==	!=		
&			
^			
|			
&&			
||			
?:			
=	op=		

Lowest

សាកលវិទ្យាល័យ អាស៊ី អឺរ៉ុប
ASIA EURO UNIVERSITY

អរគុណ
សម្រាប់ការយកចិត្តទុកដាក់

https://elearning.aeu.cloud