ជំពូកទី៤

ឃ្លាផ្ទៀងផ្ទាត់លក្ខខ័ណ្ឌ

ឃ្លាផ្ទៀងផ្ទាត់លក្ខខ័ណ្ឌនៅក្នុងភាសា Java ត្រូវបានរៀបដាក់ទៅតាមប្រភេទដូចជា: Selection, Iteration និង Jump។ ឃ្លាប្រភេទ Selection អាចអោយកម្មវិធីរបស់យើងជ្រើសយកនូវតំណើរប្រតិបត្តិការផ្សេងគ្នា ដោយផ្អែកលើលទ្ធផលនៃកន្សោមមួយ ឬ សភាពនៃអញ្ញាត្តិមួយ។ ឃ្លាប្រភេទ Iteration អាចអោយការប្រតិបត្តិកម្មវិធី ធ្វើសកម្មភាពដដែលៗនូវឃ្លាមួយ ឬ ច្រើនបាន (មានន័យថា Iteration បង្កបានដំណើរវង្វិល)។ ឃ្លាប្រភេទ Jump អាចអោយកម្មវិធីរបស់យើងប្រតិបត្តិការតាមលក្ខណៈមិនលំដាប់លំដោយ។

១. ឃ្លាប្រភេទ Selection នៃ Java(*Selection statement*):

Java ផ្តល់នូវឃ្លាប្រភេទ selection ចំនួនពីរគឺ: **if** និង **switch**។ ឃ្លាទាំងនេះអាចអោយយើងផ្ទៀងផ្ទាត់តំនើរការនៃកម្មវិធី ផ្អែកលើលក្ខខ័ណ្ឌ ដែលអាចដឹងបានពេលដំណើរការ។

១.១ ឃ្លា If (If statement)

ឃ្លា If គឺចែកចេញទៅតាមលក្ខខ័ណ្ឌរបស់ Java។ វាអាចប្រើសំរាប់នាំតំណើរការកម្មវិធី តាមផ្លូវពីរផ្សេងគ្នា។ នេះជាទំរង់របស់ឃ្លា If :

```
if (condition)
        statement;
else
        statement;
```

ក្នុងនោះ statement និមួយៗ អាចជាឃ្លាទោលមួយ ឬ ច្រើនឃ្លា ដែលដាក់ក្នុងសញ្ញា {}។ condition ជាតំលៃលេខណា ដែលអោយតំលៃជា boolean។ else ជាឃ្លាដែលអាចជំរើល។ ចូរសង្កេតឃ្លាក្នុងឧទាហរណ៍ខាងក្រោម:

```
int a, b;
//...
if ( a < b ) a = 0;
else         b = 0;
```

១.១.១ ឃ្លា If នៅក្នុង If មួយទៀត (If in If)

ឃ្លា if មួយនៅក្នុង if មួយទៀត គឺជាឃ្លា if មួយដែលជាផលនៃ if ឬ else មួយ។ កាលណាយើងដាក់ if មួយនៅក្នុង if មួយទៀតនោះ បញ្ហាសំខាន់ដែលត្រូវចាំនោះ គឺ ឃ្លា else មួយ ជានិច្ចជាកាលសំដៅទៅរក if ដែលជិតបំផុត ហើយស្ថិតក្នុង block ដូចគ្នា។ នេះជាឧទាហរណ៍មួយ:

```
if ( i==10 ){
        if ( j < 20 ) a = b;
        if ( k > 100 ) c = d;          //this if is
        else a = c;                    //associated with this else
}
else
        a = d;                         //this else refers to if ( i==10 )
```

១.១.២ លំដាប់ថ្នាក់ if-else-if (If-else-If)

ការបង្កើតកម្មវិធីទូទៅមួយ ដែលមានមូលដ្ឋាននៅលើលំដាប់តគ្នានៃឃ្លា if មួយនៅក្នុង if មួយទៀត គឺជាលំដាប់ if-else-if។ វាមានទំរង់ដូចខាងក្រោម:

```
if (condition)
        statement;
else if (condition)
        statement;
else if (condition)
        statement;
        .
        .
        .
else
        statement;
```

នេះជាកម្មវិធីមួយដែលប្រើលំដាប់ថ្នាក់ if-else-if ដើម្បីកំនត់ថាតើខែណាស្ថិតនៅក្នុងរដូវណា:

```
//demostrates if-else-if statements;
class IfElse{
        public static void main(String Args[]){
                int month = 4;
                String season;

                if ( month ==12 || month == 1 || month == 2 )
                        season = "Winter";
                else if ( month == 3 || month == 4 || month == 5 )
                        season = "Spring";
                else if ( month ==6 || month == 7 || month == 8)
                        season = "Summer";
                else if ( month == 9 || month == 10 || month == 11)
                        season = "Autumn";
                else
                        season = "Bogus Month";

                System.out.println("April is in the   " + season + ".");
        }
}
```

១.២ ឃ្លា switch (Switch statement)

ឃ្លា switch គឺជាឃ្លាដែលចែកចេញជាច្រើនផ្លូវនៅក្នុង Java។ វាផ្តល់មធ្យោបាយងាយស្រួលមួយ ដើម្បីបញ្ជូនការប្រតិបត្តិទៅអោយផ្នែកផ្សេងគ្នា នៃ code ដោយផ្នែកលើតំលៃនៃកន្សោមលេខ។ វិធីនេះផ្តល់នូវមធ្យោបាយមួយទៀតដែលល្អជាឃ្លា if-else-if។ នេះជាទំរង់ទូទៅនៃឃ្លា switch:

```
switch (expression){
        case value1:
                //statement sequence
                break;
        case value2:
                //statement sequence
                break;
```

```
            .
            .
            .
        case valueN:
            //statement sequence
            break;
        default:
            //default statement sequence
    }
```

ក្នុងនោះ expression ត្រូវតែជាប្រភេទ byte, short, int ឬ char ហើយតំលៃនិមួយៗដែលបានកំនត់នៅក្នុងឃ្លា case និមួយៗ ត្រូវតែជាប្រភេទត្រូវគ្នានឹង expression។ តំលៃរបស់ case និមួយៗ ត្រូវតែជាចំនួនថេរ មិនមែនជាអញ្ញាត្តិ។ តំលៃរបស់ case ដូចគ្នា គឺមិនអនុញ្ញាត្តិអោយប្រើទេ។

ឃ្លា break ត្រូវបានប្រើនៅក្នុង switch ដើម្បីបញ្ចប់លំដាប់ឃ្លាមួយ។ លក្ខណៈនេះ អាចអោយដំនើរការចេញផុតមកក្រៅ នៃ switch។ កម្មវិធីខាងក្រោមនេះជាឧទាហរណ៍តូចមួយពីការប្រើឃ្លា switch:

```
//simple example of switch statement
class SimpleSwitch{
        public static void main(String Args[]){
                for ( int i = 0; i<6; i++ )
                        switch ( i ){
                                case 0:
                                        System.out.println(" i is zero"); break;
                                case 1:
                                        System.out.println(" i is one"); break;
                                case 2:
                                        System.out.println(" i is two"); break;
                                case 3:
                                        System.out.println(" i is three"); break;
                                default:
                                        System.out.println(" i is greater than 3"); break;
                        }
        }
}
```

ឃ្លា break មានលក្ខណៈជំរើស (ប្រើក៏បាន មិនប្រើក៏បាន)។ បើយើងលុប break ការប្រតិបត្តិនឹងបន្តទៅកាន់ case បន្ទាប់។ ជួនកាលវាត្រូវការមាន case ជាច្រើនដោយមិនប្រើឃ្លា break នៅចន្លោះ case ទាំងនោះ។ ឧទាហរណ៍ចូរសង្កេតកម្មវិធីខាងក្រោម:

```
// In a switch, break statements are optional.
class MissingBreak{
        public static void main(String Args[]){
                for ( int i = 0; i<12; i++ ){
                        switch ( i ){
                                case 0:
                                case 1:
                                case 2:
                                case 3:
                                case 4:
                                        System.out.println("i is less than 5");
                                        break;
```

```
                        case 5:
                        case 6:
                        case 7:
                        case 8:
                        case 9:
                                System.out.println("i is less than 10");
                                break;
                        default:
                                System.out.println("i is 10 or more");
                }
        }
}
```

តំនើរការឆ្លងកាត់ចុះមកក្រោមតាម case និមយួៗ រហូតដល់ជួប break ឬ រហូលដល់ចុងបញ្ចប់នៃ switch។ កម្មវិធីខាងក្រោមនេះជាគំរូមួយ ដែលបង្ហាញអោយឃើញច្បាស់ពីការអនុវត្តន៍មានប្រសិទ្ធិភាពច្រើន។ យើងលើយកកម្មវិធីលើកមុនមកសរសេរឡើងវិញ ដូច្នេះយើងអាចសង្កេតពីការប្រើវា:

```
//an improved version of the season program.
class ImprovedSeason{
        public static void main(String Args[]){
                int month = 4;
                String season;

                switch ( month ){
                        case 12:
                        case 1:
                        case 2:
                                season = "winter"; break;
                        case 3:
                        case 4:
                        case 5:
                                season = "spring"; break;
                        case 6:
                        case 7:
                        case 8:
                                season = "summer"; break;
                        case 9:
                        case 10:
                        case 11:
                                season = "autumn"; break;
                        default:
                                season = "bogus month";
                }
                System.out.println("April is in the " + season + ".");
        }
}
```

យើងអាចប្រើ switch ជាផ្នែកនៃលំដាប់ឃ្លារបស់ switch នៅខាងក្រៅបាន។ លក្ខណៈនេះហៅថា switch មួយនៅក្នុង switch មួយទៀត (*nested switch*)។ ដោយឃ្លា switch កំនត់នូវ block រៀងខ្លួន ធ្វើអោយវាមិនមានការប៉ះទង្គិចគ្នា រវាងតំលៃថេររបស់ case នៅ switch ខាងក្នុង ហើយនិង switch ខាងក្រៅ។ ឧទាហរណ៍សរសេរឃ្លាខាងក្រោមនេះពិតជាត្រឹមត្រូវ:

```
switch ( count ){
        case 1:
                switch ( target ){
                        case 0:
                                System.out.println (“target is zero”); break;
                        case 1:
                                System.out.println (“target is one”); break;
                }
        case 2:
                .
                .
                .
}
```

២. ឃ្លាប្រភេទ Iteration នៃ Java(*Iteration statement*):

ឃ្លាប្រភេទ Iteration នៅក្នុង Java គឺ **for**, **while** និង **do-while**។ ឃ្លាទាំងនេះបង្កើតឡើងនូវអ្វីដែលយើងហៅជាទូទៅថា ដំនើរវង្វិល (*loop*)។

២.១ ដំនើរវង្វិល while (While loop statement)

ដំនើរវង្វិល while គឺជាឃ្លាដំនើរបង្វិលសំខាន់បំផុតរបស់ Java។ វាប្រព្រឹត្តដដែលៗនូវឃ្លាមួយ ឬ block មួយ នៅពេលដែលកន្សោមលក្ខណ្ឌពិត។ នេះជាទំរង់របស់វា:

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

នៅក្នុងនោះ condition អាចជាកន្សោម boolean ណាមួយ។ តួនៃតំនើរវង្វិល នឹងប្រតិបត្តិរហូត បើកន្សោមលក្ខណ្ឌពិត។ នៅពេលដែល condition មិនពិត នោះតំណើរផ្ទៀងផ្ទាត់ នឹងឆ្លងទៅបន្ទាត់បន្ទាប់នៃ code ដែលនៅជាប់នឹង តំនើរវង្វិល។ បើមានឃ្លាតែមួយប៉ុណ្ណោះ នៅក្រោមលក្ខណ្ឌដែលត្រូវផ្ទៀងផ្ទាត់ដដែលៗនោះ គេមិនចាំបាច់ប្រើសញ្ញា {} ទេ។

ឧទាហរណ៍នេះជាវង្វិល while ដែលរាប់លេខថយពី 10 មក ហើយបោះចេញនូវពាក្យ “”tick” ចំនួន 10 បន្ទាត់។

```
//demonstrate the while loop
class WhileLoop{
        public static void main(String Args[]){
                int n = 10;
                while ( n > 0 ){
                        System.out.println("tick " + n);
                        n--;
                }
        }
}
```

ដោយសារតែ while រង្វាយតំលៃក្នុងកន្សោមលក្ខណ្ឌនៅខាងលើតំនើរវង្វិល តួនៃតំនើរវង្វិល នឹងមិនប្រតិបត្តិឡើយ ប្រសិនបើលក្ខណ្ឌមិនពិត នៅពេលចាប់ផ្តើម។ ឧទាហរណ៍code នៅខាងក្រោមនេះ ការប្រើ println() មិនប្រតិបត្តិទាល់តែសោះ។

```
int a = 10, b = 20;
while ( a > b )
        System.out.println("This will not be displayed");
```

តួរបស់ while (ឬ តំនើរវង្វិលផ្សេងទៀតរបស់ Java) អាចស្ថិតនៅទទេ។ នេះមកពីវាជាឃ្លាទទេ (ជាឃ្លាដែលមានតែសញ្ញា ; ប៉ុណ្ណោះ) ដែលជាឃ្លាត្រឹមត្រូវ នៅក្នុងភាសា Java ។ ឧទាហរណ៍ ចូរសង្កេតកម្មវិធីខាងក្រោម:

```
//The target of a loop can be empty
class NoBody{
        public sattic void main(String Args[]){
                int i, j;

                i= 100;
                j= 200;

                while ( ++i < --j );
                System.out.println("Midpoint is " + i);
                }
        }
}
```

២.២ <u>តំនើរវង្វិល do-while</u> (Do-While loop statement)

ដំនើរវង្វិល do-while ជានិច្ចជាកាលប្រតិបត្តិការក្នុងតួរបស់វា យ៉ាងតិចម្តង ពីព្រោះកន្សោមលក្ខណ្ឌរបស់វា នៅខាងក្រោមនៃដំនើរវង្វិល។ ទំរង់ទូទៅរបស់វាគឺ:

```
do {
        //body of loop
} while ( condition );
```

នៅក្នុងដំនើរវង្វិល do-while ប្រតិបត្តិការម្តងៗ គឺដំបូងតួរបស់វា រួចហើយទើបរង្វាយតំលៃកន្សោមលក្ខណ្ឌ។ បើសិនជាកន្សោមលក្ខណ្ឌនោះ ពិត នោះដំនើរវង្វិលគឺប្រតិបត្តិដំនើរការសារជាថ្មីម្តងទៀត។ ផ្ទុយទៅវិញ ដំនើរការត្រូវបញ្ចប់។ គ្រប់ដំនើរវង្វិលទាំងអស់របស់ Java, condition ត្រូវតែជាកន្សោម boolean។ កម្មវិធីខាងក្រោមនេះ ជាកម្មវិធីលើកមុនដែលអោយលទ្ធផលជា "tick" ចំនួន 10 ដង តែក្នុងពេលនេះយើងយកវាមកប្រើជាមួយឃ្លា do-while វិញម្តង។ វាអោយលទ្ធផលដូចគ្នានឹងលើកមុនដែរ:

```
//Demonstrate the do-while loop
class DoWhile{
        public static void main(String Args[]){
                int n = 10;

                do{
                        System.out.println("tick " + n);
                        n--;
                }while ( n > 0 );
        }
}
```

២.៣ ដំនើរវង្វិល for (for loop statement)

ដំនើរវង្វិល for គឺជារចនាសម្ព័ន្ធមួយមានអនុភាពខ្លាំង និង មានលក្ខណៈបត់បែន។ នេះជាទំរង់ទូទៅរបស់ for:

```
for ( initialization; condition; iteration ){
    body;
}
```

នេះជាកម្មវិធីប្រើដំនើរវង្វិល for:

```
//Demonstrate the for loop
class ForTick{
    public static void main(String Args[]){
        int n;
        for ( n =10 ; n > 0 ; n-- ){
            System.out.println("tick " + n);
        }
    }
}
```

២.៤ ការប្រកាសអញ្ញាត្តិនៅក្នុងដំនើរវង្វិល for (Declaring a variable in a for loop)

អញ្ញាត្តិដែលផ្ទៀងផ្ទាត់ដំនើរវង្វិល for គឺត្រូវការសំរាប់តែគោលបំណងនៃឃ្លា for ប៉ុណ្ណោះ ហើយមិនអាចប្រើនៅកន្លែងផ្សេងទៀតបានឡើយ។ នៅក្នុងករណីនេះ វាអាចអោយយើងប្រកាសអញ្ញាត្តិនៅខាងក្នុង for ផ្តែកកំនត់តំលៃដំបូងរបស់ for។ ឧទាហរណ៍ខាងក្រោមនេះ ជាកម្មវិធីលើកមុនដែលបានលើកឡើង ដូច្នេះអញ្ញាត្តិផ្ទៀងផ្ទាត់ដំនើរវង្វិល n នឹងត្រូវប្រកាសជាប្រភេទ int ដូចនៅក្នុងឃ្លា for ដែរ។

```
//Demonstrate a loop control variable inside  for loop
class VarLoop{
    public static void main(String Args[]){

        for ( int n =10 ; n > 0 ; n-- ){
            System.out.println("tick " + n);
        }
    }
}
```

កាលណាយើងប្រកាសអញ្ញាត្តិមួយនៅក្នុងដំនើរវង្វិល for យើងត្រូវដឹងថា វាមានចំនុចសំខាន់មួយ គឺ ទំហំនៃអញ្ញាត្តិនោះបញ្ចប់ នៅពេលឃ្លា for បញ្ចប់ដែរ។ នេះមានន័យថា ទំហំរបស់អញ្ញាត្តិ ត្រូវបានកំរិតដោយឃ្លា for។

កាលណាអញ្ញាត្តិផ្ទៀងផ្ទាត់ដំនើរវង្វិល មិនត្រូវការនៅកន្លែងផ្សេងទៀតនោះ អ្នកសរសេរកម្មវិធីប្រកាសវា នៅក្នុង for។ ឧទាហរណ៍ខាងក្រោមនេះ ជាកម្មវិធីតូចមួយ ដែលផ្ទៀងផ្ទាត់ចំនួនបឋម។ ចូរកត់សំគាល់ថា អញ្ញាត្តិផ្ទៀងផ្ទាត់ដំនើរវង្វិល i ត្រូវប្រកាសនៅក្នុង for ព្រោះវាមិនត្រូវការប្រើនៅកន្លែងផ្សេងទៀត។

```
//Test for primes.
class FindPrime{
        public static void main(String Args[]){
                int num;
                boolean isPrime = true;
                num = 14;
                for ( int i =2; i < num / 2; i++ ){
                        if ( num % i == 0 ){
                                isPrime = false;
                                break;
                        }
                }
                if ( isPrime )
                        System.out.println("Prime");
                else
                        System.out.println("Not Prime");
        }
}
```

២.៥ ការប្រើសញ្ញាក្បៀស (Using comma)

ដើម្បីអោយអញ្ញាត្តិពីរ ឬ ច្រើនធ្វើងផ្តាត់ដំនើរវង្វិល for , Java អាចអោយយើង បញ្ចូលឃ្លាច្រើន នៅត្រង់ទីតាំងកំនត់តំលៃដំបូង និង ទីតាំងធ្វើសកម្មភាពដដែលៗរបស់ for។ ឃ្លាមួយនៅដាច់ពីឃ្លាមួយទៀត ដោយសញ្ញាក្បៀស។

តាមរយៈការប្រើសញ្ញាក្បៀស ដំនើរវង្វិល for លើកមុន អាចអោយយើងសរសេរ code មានប្រសិទ្ធិភាពជាង ដូចបានបង្ហាញខាងក្រោមនេះ:

```
//Using the comma
class Comma{
        public static void main(String Args[])[
                int a, b;
                for ( a =1, b =4; a < b; a++, b--){
                        System.out.println("a = " + a);
                        System.out.println("b = " + b);
                }
        }
}
```

២.៦ ដំនើរវង្វិលមួយនៅក្នុងដំនើរវង្វិលមួយទៀត (for in for)

ដូចក្នុងភាសាដ៏ទៃទៀតដែរ Java អាចអោយដំនើរវង្វិល ស្ថិតក្នុងដំនើរវង្វិលមួយទៀត។ ខាងក្រោមនេះជាកម្មវិធីមួយ ប្រើ for មួយនៅក្នុង for មួយទៀត។

```
//loop may be nested
class Nested{
        public static void main(String Args[]){
                int i, j;

                for ( i = 0 ; i < 10 ; i++ ){
                        for ( j = i ; j < 10 ; j++ )
                                System.out.println(".");
                        System.out.println();
                }
        }
}
```

៣. ឃ្លាប្រភេទ Jump(*Jump statement*):

Java ផ្តល់នូវឃ្លាប្រភេទ Jump ចំនួនបីគឺ: break, continue និង return។

៣.១ ការប្រើ break (Using break)

នៅក្នុង Java ឃ្លា break មានការប្រើ 3 យ៉ាង។ ទីមួយ ដូចយើងបានដឹងហើយថា វាប្រើសំរាប់ បញ្ចប់លំដាប់នៅក្នុងឃ្លា switch។ ទីពីរ វាអាចប្រើសំរាប់ចាកចេញពីដំនើរវង្វិល។ ទីបី វាត្រូវបានប្រើជាទំរង់ goto។ នៅពេលនេះ យើងនឹងពន្យល់ពីការប្រើប្រាស់ឃ្លាទាំងពីរចុងក្រោយនេះ។

៣.២ ការប្រើ break សំរាប់ចាកចេញពីដំនើរវង្វិល(Using break to break a loop)

តាមរយៈការប្រើឃ្លា break យើងអាចបង្ខំអោយវាបញ្ចប់ភ្លាមៗ នូវដំនើរការបង្វិលបាន ដោយរំលងកន្សោមលក្ខណ្ឌ និង code ដែលនៅសល់ក្នុងតួនៃ loo។ កាលណាឃ្លា break មួយ ជួបប្រទះនៅក្នុង loop ដំនើរបស់ loop ត្រូវបានបញ្ចប់ ហើយកម្មវិធីធ្វើងផ្ទាត់នឹងចាប់ផ្តើមនៅឃ្លាបន្ទាប់។ ខាងក្រោមជាឧទាហរណ៍:

```
//Using break to exit a loop
class BreakLoop{
        public static void main(String Args[]){
                for ( int i = 0; i < 100 ; i++ ) {
                        if ( i == 10 ) break;           //terminate loop if i = 10
                        System.out.println("i: " + i);
                }
                System.out.println("Loop completed");
        }
}
```

ឃ្លា break អាចប្រើជាមួយឃ្លា loop របស់ Java។ ឧទាហរណ៍ខាងក្រោមនេះជាកម្មវិធីលើកមុនដែលបានសរសេរបង្ហាញពីការប្រើដំនើរវង្វិល while។ លទ្ធផលនៃកម្មវិធីនេះ ក៏ដូចទៅនឹងកម្មវិធីលើកមុនដែរ:

```
//Using break to exit a while loop
class BreakLoop2{
        public static void main(String Args[]){
                int i = 0;
                while ( i < 100 ) {
                        if ( i == 100 ) break;
                        System.ou.println("i: " + i );
                        i++;
                }
                System.out.println("Loop complete");
        }
}
```

កាលណាប្រើនៅក្នុងដំនើរវង្វិលច្រើនជាន់នោះ ឃ្លា break នឹងអាចចាកចេញតែដំនើរវង្វិលណាដែលនៅក្នុងបំផុត។ ឧទាហរណ៍:

```
//Using break with nested loop
class BreakLoop3{
        public static void main(String Args[]){

                for ( int i = 0; i < 3 ; i++ ){
                        System.out.println("Pass i " + i + ":" );
                        for ( int j = 0; j < 100 ; j++ ){
                                if ( j == 10 ) break;
                                System.out.println( j + " " );
                        }
                        System.out.println();
                }
                System.out.println("Loop complete");
        }
}
```

៣.៣ **ការប្រើ break ជួចនឹងទំរង់ goto**(Using break as a goto)

ឃ្លា break អាចប្រើដោយខ្លួនវា ដើម្បីផ្តល់នូវទំរង់ goto។ ទំរង់ទូទៅនៃឃ្លា break បានបង្ហាញដូចខាងក្រោម:

***break** label;*

ក្នុងនោះ label គឺជាឈ្មោះនៃសញ្ញាសំគាល់ ដែលប្រើសំរាប់សំគាល់ block នៃ code។ នៅពេលទំរង់នៃ break នេះប្រតិបត្តិការ នោះការផ្ទៀងផ្ទាត់គឺសំដៅទៅរកឈ្មោះ block នៃ code។ ឃ្លា break ដែលមានឈ្មោះត្រូវដាកនៅក្នុង block នៃ code។ នេះមានន័យថា យើងអាចប្រើឃ្លា break ដែលមានឈ្មោះសំរាប់ចាកចេញពីសំនុំ block។ ប៉ុន្តែយើងមិនអាចប្រើឃ្លា break ដើម្បីរក block នៃ code ដែលមិនមានអមឃ្លា break នោះឡើយ។

ដើម្បីដាក់ឈ្មោះអោយ block មួយ យើងដាក់ឈ្មោះសញ្ញាសំគាល់នៅខាងដើម block និង សញ្ញា : នៅពីក្រោយ។ កម្មវិធីខាងក្រោមនេះ បង្ហាញពី block បីជាន់ ដែល block និមួយៗ មានឈ្មោះរៀងៗខ្លួន។ ឃ្លា break ធ្វើអោយការប្រតិបត្តិលោតទៅមុខ ហួសទីបញ្ចប់នៃ block ដែលមានឈ្មោះ second ដោយរំលងឃ្លា println() ពីរ។

```
//Using a break as a civilized form of goto
class Break{
        public static void main(String Args[]){
                boolean t = true;
                first: {
                        second: {
                                third: {
                                        System.out.println("Before the break.");
                                        if ( t )
                                                break second;
                                                System.out.println("This wont execute");
                                }
                                System.out.println("This wont execute");
                        }
                        System.out.println("This is after second block");
                }
        }
}
```

ការប្រើឃ្លា break ជាមួយឈ្មោះសញ្ញាសំគាល់ច្រើនជាងគេនោះ គឺ ដើម្បីចាកចេញពីជំនើរវង្វិលមួយ នៅក្នុងជំនើរវង្វិលមួយទៀត។ ចូរសង្កេតកម្មវិធីខាងក្រោមនេះ:

```
//Using break to exit from nested loop
class BreakLoop4{
        public static void main(String Ags[]){
                outer: for ( int i = 0 ; i < 3 ; i++ ){
                                System.out.println("Pass " + i );
                                for ( int j = 0 ; j < 100 ; j++ ){
                                        if ( j == 10 ) break outer;   \\exit both loops
                                        System.out.println( j + " ");
                                }
                                System.out.pirntln("This will not print");
                        }
                        System.out.println("Loops complete");
        }
}
```

ដូចយើងបានឃើញស្រាប់ហើយថា នៅពេលជំនើរវង្វិលខាងក្នុង បំបែកចេញទៅកាន់ជំនើរវង្វិលខាងក្រៅនោះ ជំនើរទាំងពីរត្រូវបានបញ្ចប់។ ត្រូវចាំទុកថា យើងមិនអាចបំបែកចេញទៅកាន់ឈ្មោះសញ្ញាណាមួយ ដែលមិនបានកំនត់សំរាប់ block នោះឡើយ។ ខទាហរណ៍ខាងក្រោមនេះ មិនត្រឹមត្រូវ ហើយមិនធ្វើការបំបែកចេញឡើយ:

```
class BreakErr{
        public static void main(String Args[]){
                one: for ( int i = 0 ; i < 3 ; i++ ){
                                System.out.println("Pass " + i);
                        }
                for ( int j = 0 ; j < 100 ; j++ ){
                        if ( j == 10 ) break one;     //Wrong
                        System.out.println( j + " " );
                }
        }
}
```

៤. ការប្រើ Continue(*Using a Continue statement*):

ជួនកាលវាមានប្រយោជន៍ដែរ ក្នុងការបង្ខំអោយឆាប់ធ្វើសកម្មភាពដដែលៗ របស់ដំនើរវង្វិល។ បានន័យថា យើងចង់បន្តដំនើរការវង្វិល ប៉ុន្តែបញ្ឈប់ដំនើរការ code ដែលនៅសេសសល់ ក្នុងតួរបស់វា ចំពោះការធ្វើសកម្មភាពដដែលៗនេះ។ លក្ខណៈនេះគឺជា goto ដែលរំលងតួនៃដំនើរវង្វិល ទៅកាន់ចុងក្រោយនៃដំនើរវង្វិល។ ឃ្លា continue ធ្វើសកម្មភាពបែបនេះ។ នេះជាគំរូ ដែលប្រើ continue បណ្តាលអោយពីរចំនួន បោះនៅលើបន្ទាត់និមួយៗ។

```
//Demonstrate Continue
class Continue{
        public static void main(String Args[]){
                for ( int i = 0 ; i < 10 ; i++ ){
                        System.out.println( i + " " );
                        if ( i % 2 == 0 ) continue;
                        System.out.pirntln(" ");
                }
        }
}
```

ដូចគ្នានឹង break ដែរ continue អាចកំនត់នូវសញ្ញាសំគាល់មួយ ដើម្បីប្រាប់នូវដំនើរវង្វិលណាដែលត្រូវបន្តសកម្មភាព។ នេះជាកម្មវិធីមួយ ដែលប្រើ continue ដើម្បីបោះចេញនូវតារាងមួយ រាងត្រីកោណពី 0 ដល់ ៩។

```
//Using continue with label
class ContinueLabel{
        public static void main(String Args[]){
                outer: for ( int i = 0 ; i < 10 ; i++ ){
                        for ( int j = 0 ; j < 10 ; j++ ){
                                if ( j > i ){
                                        System.out.println();
                                        continue outer;
                                }
                                System.out.println( " " + (i * j) );
                        }
                        System.out.println();
                }
        }
}
```

៥. ការប្រើ return(*Using a return statement*):

ឃ្លា return ត្រូវបានប្រើសំរាប់អោយតំលៃពី method មួយដោយច្បាស់លាស់។ បានន័យថា វាផ្ទៀងផ្ទាត់ដើម្បីបន្ទេរទៅអោយ អ្នកហៅ method នោះ។ វាត្រូវបានចាត់ចូលជាឃ្លាប្រភេទ Jump។

នៅពេលណាក៏ដោយនៅក្នុង method មួយ, ឃ្លា return អាចប្រើសំរាប់ធ្វើអោយការប្រតិបត្តិ បែកចេញទៅរកអ្នកហៅ method នោះ។ ដូច្នេះ ឃ្លា return បញ្ចប់ method ភ្លាមៗ ពេលវាត្រូវបានដំនើរការនៅក្នុង method នោះ។ ឧទាហរណ៍ខាងក្រោមនេះបង្ហាញពីចំនុចទាំងនេះ។ ក្នុងនេះ return ធ្វើអោយការប្រតិបត្តិវិលទៅរកប្រព័ន្ធដំនើរការ របស់ Java ពេលវាហៅ main។

```
//Demonstrate return
class Return{
        public static void main(String Args[]){
                boolean t = true;

                System.out.println("Before the return");
                if ( t ) return;        // return to caller
                System.out.println("This wont execute");
        }
}
```