

Control Flow Statements

Bok, Jong Soon
javaexpert@nate.com
<https://github.com/swacademy/JavaSE>

Simple Programming Constructs

- Conditions - Decide at runtime whether to perform certain statements.
- Loops - Decide at runtime how many times to perform certain statements.
- Branches

The **if** construct

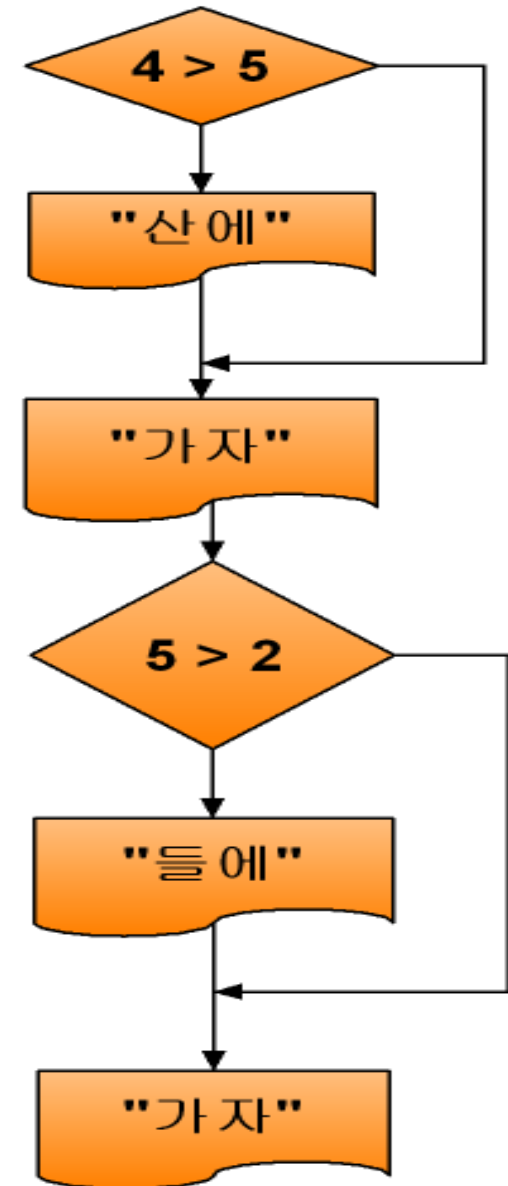
- Allows your program to make simple decisions based on stored values.
- JVM checks whether the **Boolean** expression is **true** or **false**.
- Can decide between two different statements with one condition.
- Can extend the **if** clause with the **else** clause.
- Can use **if** with code blocks.

Syntax - if

```
if (boolean expression) {  
    statement ;  
}
```

```
2 public class IfDemo {  
3     public static void main(String[] args) {  
4         if( 4 > 5 ) System.out.println("산에");  
5         System.out.println("가자");  
6         if( 5 > 2 ) System.out.println("들에");  
7         System.out.println("가자");  
8     }  
9 }
```

가자
들에
가자

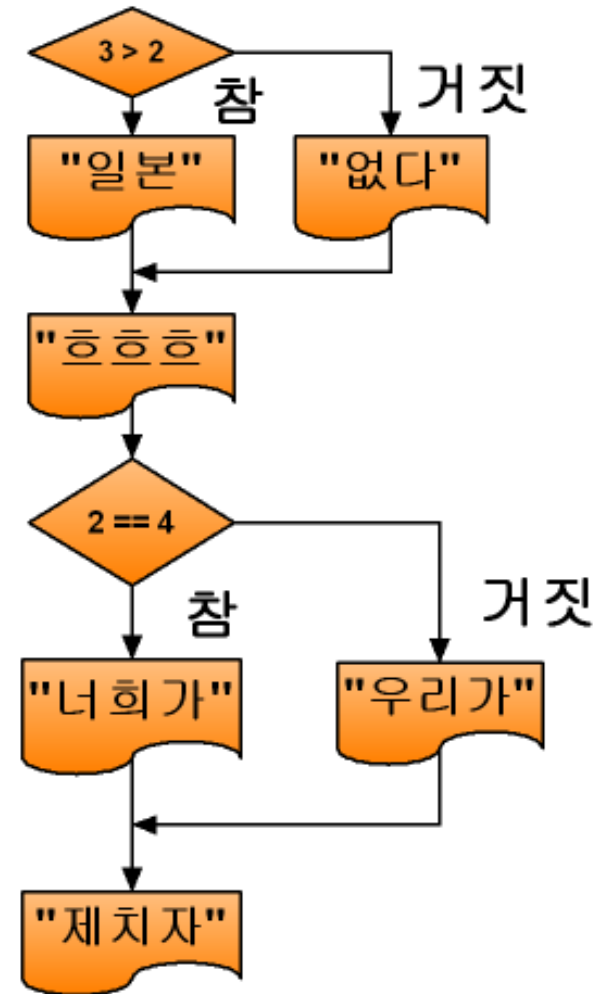


Syntax – if (Cont.)

```
if (boolean expression) {  
    statement ;  
}  
else {  
    statement ;  
}
```

```
2 public class IfDemo1 {  
3     public static void main(String[] args) {  
4         if (3 > 2) System.out.println("일본");  
5         else System.out.println("없다");  
6         System.out.println("ㅋㅋㅋㅋ");  
7         if (2 == 4) System.out.println("너희가");  
8         else System.out.println("우리가");  
9         System.out.println("제치자");  
10    }  
11 }
```

일본
우리가
제치자



Warning – block processing

```
2 public class IfDemo2 {  
3     public static void main(String[] args) {  
4         int a = 5, b = 3;  
5         if( b > a ) System.out.print("하나");  
6             System.out.print("두울");  
7             System.out.print("셋");  
8     }  
9 }
```

두울셋

```
2 public class IfDemo2 {  
3     public static void main(String[] args) {  
4         int a = 5, b = 3;  
5         if( b > a ) {  
6             System.out.print("하나");  
7             System.out.print("두울");  
8         }  
9         System.out.print("셋");  
10    }  
11 }
```

셋

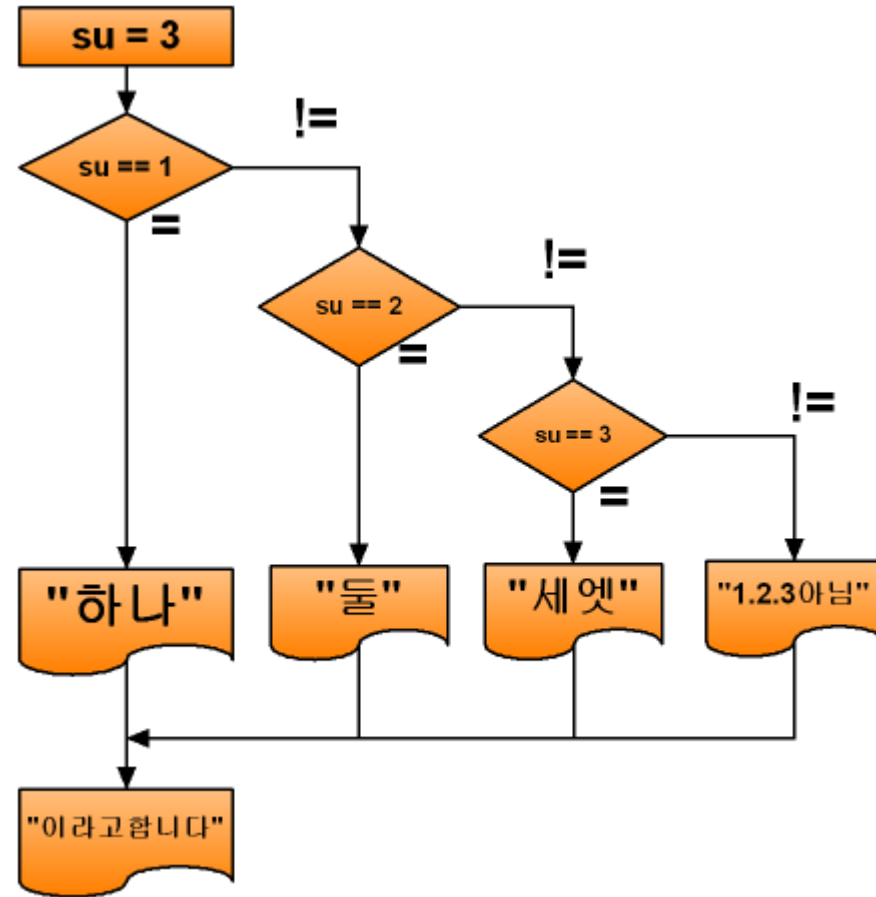
Syntax – **if** (Cont.)

```
if (boolean expression) {  
    statement ;  
} else if (boolean expression) {  
    statement ;  
} else if (boolean expression) {  
    statement ;  
} else {  
    statement ;  
}
```

Syntax – **if** (Cont.)

```
2 public class IfDemo3 {  
3     public static void main(String[] args) {  
4         int su = 3;  
5         if( su == 1 ) System.out.println("하나");  
6         else if ( su == 2 ) System.out.println("둘");  
7         else if ( su == 3 ) System.out.println("세엣");  
8         else System.out.println("1.2.3. 아님");  
9         System.out.println("이라고 합니다");  
10    }  
11 }
```

세엣
이라고 합니다



*Exercise : Using the **if** Construct*

- Generate a random number between 1 and 10 and print a message (“Bananas”) if the number is greater than 5.
- Use to create a random integer number in the range of 1 and a variable called *max*.

```
int i = (int) ( ( Math.random() * max ) + 1) ;
```

*Exercise : Using the **if** Construct (Cont.)*

- Generates two different random numbers in the range 1 to 10 called rand1 and rand2, and prints out the following messages.
- If $\text{rand1} \leq 3$ print the message “Bananas”.
- If $\text{rand1} > 3$ and $\text{rand2} \leq 5$ print the message “Oranges”.
- If $\text{rand1} > 3$, $\text{rand2} > 5$ print the message “Pears”

The **switch** Construct

- Is used if all of the conditions are equality tests against a single variable.
- The type of **i** can be only **char**, **byte**, **short**, or **int**, and **enum** (*JDK 1.5 higher*).
- The **case** labels must be literals.
- The **default** case is the same as the **else** in an **if** construct.
- The **break** statement is used to exit out of a **switch** statement.
- If a **case** statement does not contain a **break**, the line of code after the completion of the **case** will be executed.

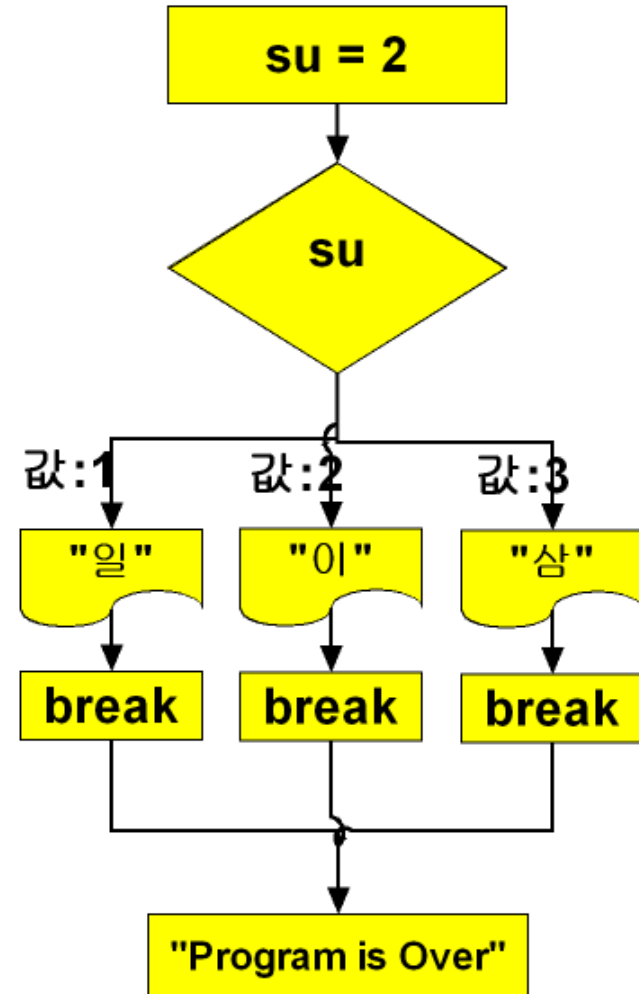
Syntax – switch

```
switch ( variable ) {  
    case constant1 :  
        statement ;  
        break ;  
    case constant2 :  
        statement ;  
        break ;  
    default :  
        statement ;  
}
```

Syntax – switch (Cont.)

```
2 public class SwitchDemo {  
3     public static void main(String[] args) {  
4         int su = 2;  
5         switch( su ) {  
6             case 1 : System.out.println("일"); break;  
7             case 2 : System.out.println("이"); break;  
8             case 3 : System.out.println("삼"); break;  
9         }  
10        System.out.println("Program is Over...");  
11    }  
12 }
```

```
01  
Program is Over...
```



*Exercise: Using the **switch** Statement*

- Generates a random number in the range of 1 to 10 and prints the following messages based on the value:

1 : “Bananas”

2 : ”Oranges”

3 : “Peach”

3 or 4 : “Apples”

3 or 4 or 5 : “Plums”

6 : “Pineapples”

7 : No message – ignore this case

Any other value : “Nuts”

The **for** Loop

- Provides a compact way to iterate over a range of values.
- Initialize - Is the section that is processed *once*, before any other part of the loop.
- Condition - Is the section that is processed just before each iteration of the loop.
- Statement - Is the statement or code block which is processed with every loop iteration.
- Update - Is the section that is processed after the body but before each subsequent retest of the condition.

Syntax - **for**

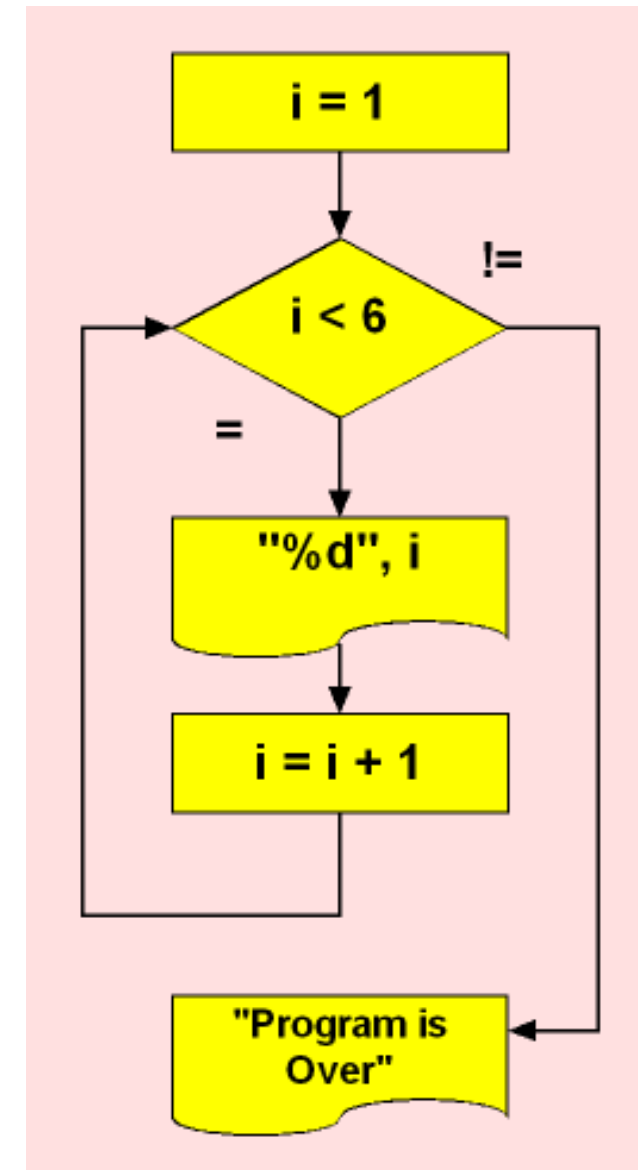
```
for ( initialize;condition;update ) {  
    statement ;  
}
```

- Multiple initializations must be separated with commas(,) not semi-colons(;).
- Condition must be a *boolean* expression.

Syntax – for (Cont.)

```
2 public class ForDemo {  
3     public static void main(String[] args) {  
4         for (int i = 1 ; i < 6 ; i++){  
5             System.out.printf("%d\t", i);  
6         }  
7     }  
8 }
```

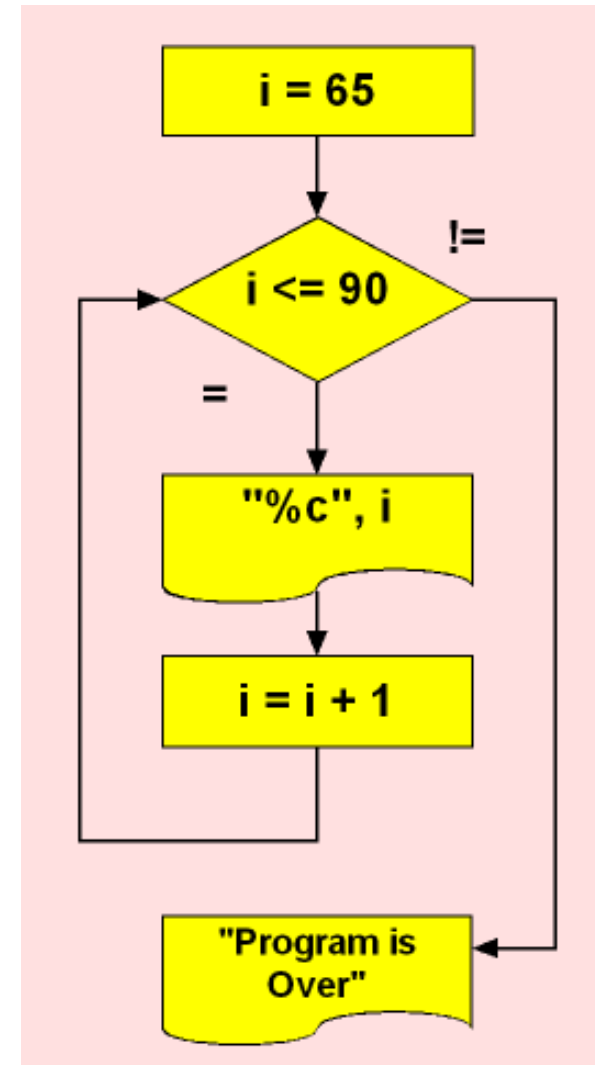
1	2	3	4	5
---	---	---	---	---



Syntax – for (Cont.)

```
2 public class ForDemo {
3     public static void main(String[] args) {
4         System.out.printf("%40s\n", "*** 영문 대문자 ***");
5         int i = 65 ;
6         for( ; i <= 90 ; ){
7             System.out.printf("%2c", i);
8             i++;
9         }
10        System.out.println("\nProgram is Over...");
11    }
12 }
```

```
                ** 영문 대문자 **
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Program is Over...
```



Syntax – for (Cont.)

```
2 public class ForDemo {  
3     public static void main(String[] args) {  
4         int a, b;  
5         for ( a = 1, b = 200 ; a <=3 ; b -= 50, a++) {  
6             System.out.printf("a = %d    b = %d\n", a, b );  
7         }  
8         System.out.println("\nProgram is Over...");  
9     }  
10 }
```

a = 1	b = 200
a = 2	b = 150
a = 3	b = 100

Program is Over...

Syntax – **for** (Cont.)

```
2 public class MultiForDemo {  
3     public static void main(String[] args) {  
4         int a, b, c = 100;  
5         System.out.println("<<<다중 for 문>>>");  
6         for ( a = 1 ; a <= 2 ; a++) {  
7             for ( b = 1 ; b <= 3 ; b++) {  
8                 c += 10 ;  
9                 System.out.printf("%5d\n", c);  
10            }  
11        }  
12        System.out.println("Program is Over...");  
13    }  
14 }
```

```
<<<다중 for 문>>>  
 110  
 120  
 130  
 140  
 150  
 160  
Program is Over...
```

Enhanced **for** Loop from Java 1.5

- a.k.a. the *for in* loop and *for each* loop
- Is used to iterate through an array or collection of any object that implements Iterable.
- The loop is executed once for each element of the array or collection
- Does not use a counter, as the number of iterations is already determined.
- See also
<http://java.sun.com/j2se/1.5.0/docs/guide/language/foreach.html>
- Syntax :
for (Type Identifier : Expression)

Enhanced **for** Loop in Java 1.5 (Cont.)

```
2 public class NewForDemo {  
3     public static void main(String[] args) {  
4         // TODO Auto-generated method stub  
5         int [] array = {5,6,7,8,9};  
6         /*old version's for loop  
7         for(int i = 0 ; i< array.length ; i++){  
8             System.out.println(array[i]);  
9         }  
10        */  
11        //new version's for loop  
12        for(int su : array){  
13            System.out.println(su);  
14        }  
15    }  
16 }
```

The **while** loop

- Continually execute a block of statements while a condition remains *true*.
- Can perform more than one statement by using a code block.

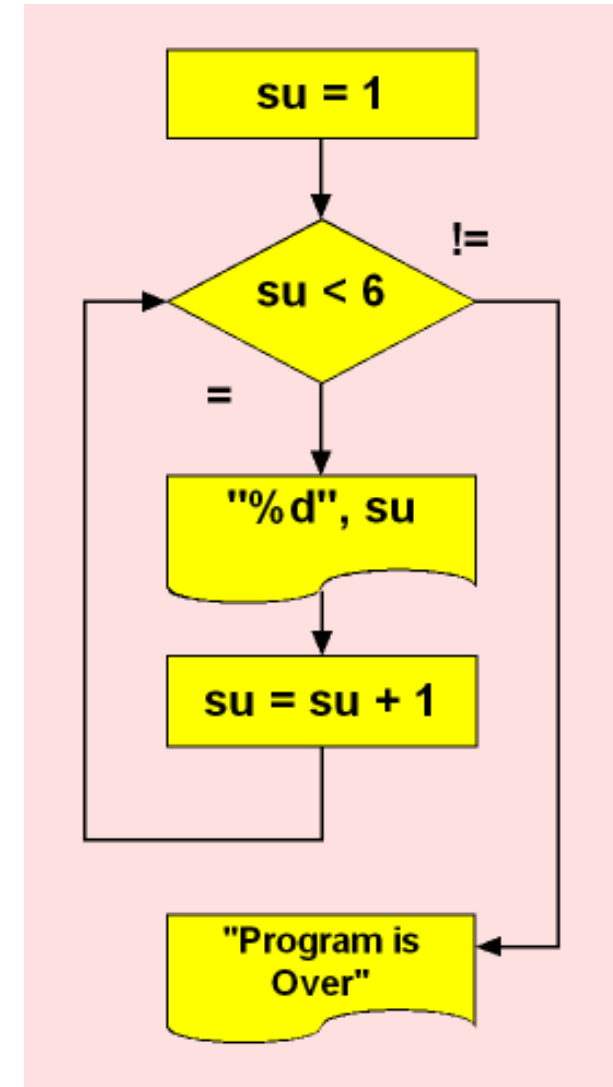
Syntax – while

```
initialization ;  
while (boolean expression) {  
    statement ;  
    update ;  
}
```


Syntax – while (Cont.)

```
2 public class WhileDemo {  
3     public static void main(String[] args) {  
4         int su = 1;  
5         while ( su < 6 ) {  
6             System.out.printf("%d\t", su);  
7             su++;  
8         }  
9         System.out.println("\nProgram is Over...");  
10    }  
11 }
```

```
1      2      3      4      5  
Program is Over...
```



Syntax – while (Cont.)

```
2 public class MultiWhileDemo {  
3     public static void main(String[] args) {  
4         int a, b;  
5         a = 1;  
6         while ( a < 11) {  
7             b = 1;  
8             while ( b <= a ){  
9                 System.out.printf("%d", b);  
10                b++;  
11            }  
12            System.out.println();  
13            a++;  
14        }  
15        System.out.println("\nProgram Is Over...");  
16    }  
17 }
```

```
1  
12  
123  
1234  
12345  
123456  
1234567  
12345678  
123456789  
12345678910
```

Program Is Over...

The **do** Loop

- **while** and **for** loops are used for *zero/many* iterative loops.
- **do** is used for *one/many* iterative loops.
- Condition at the bottom of the loop is processed after the body.
- Body of loop is processed at least *once*.

Syntax - do

```
initialization ;
```

```
do {
```

```
    statement ;
```

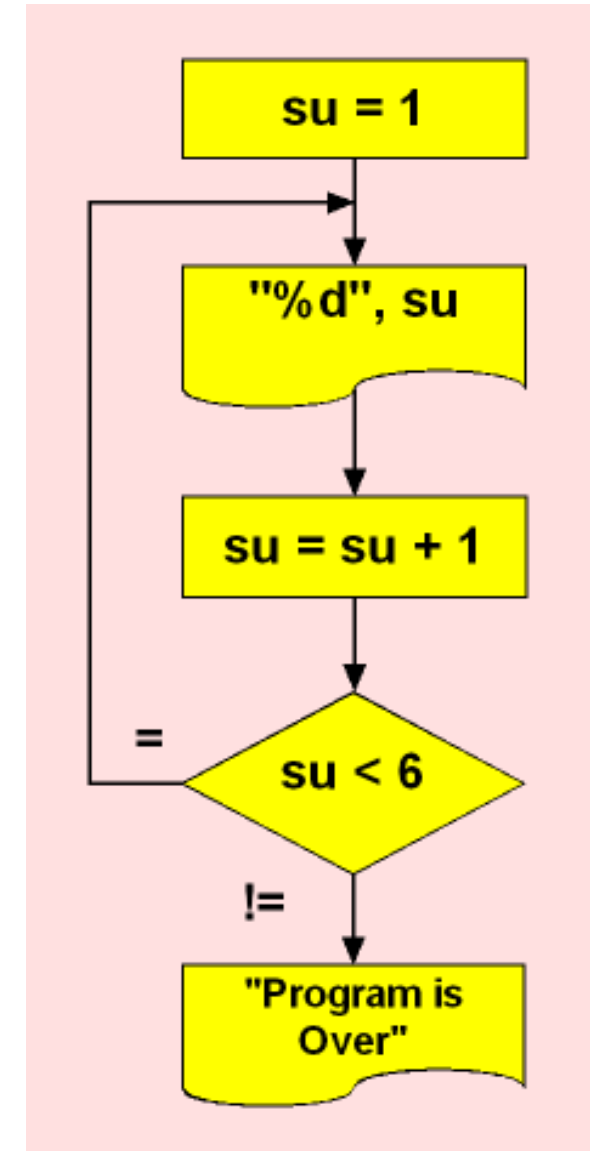
```
    update;
```

```
} while ( boolean expression) ;
```

Syntax – do (Cont.)

```
2 public class DoDemo {  
3     public static void main(String[] args) {  
4         int su = 1;  
5         do {  
6             System.out.printf("%d\t", su);  
7             su++;  
8         }while( su < 6);  
9         System.out.println("\nProgram Is Over...");  
10    }  
11 }
```

1	2	3	4	5
Program Is Over...				



*Exercise : Using the **while** Loop*

1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Comparing Loop Constructs

- **while** – Iterates indefinitely through statements and performs the statements zero or more times.
- **do** – Iterates indefinitely through statements and performs the statements *one* or more times.
- **for** – Steps through statements a predefined number of times

Transfer of Control

- `break [label] ;`
- `continue [label] ;`
- `label : statement ;` //Where statement be
// any legal statement

The **break** Statement

- The **break** statement permits the controlled and immediate termination of a loop or **switch** statement.
- It can be used to prevent fall thru.
- The **break** statement is valid inside **while**, **for**, **do**, and **switch** constructs only.

The **break** Statement (Cont.)

```
1 public class Test{
2     public static void main(String[] args) {
3         for(int i = 0 ; i < 3 ; i++){
4             for(int j = 0 ; j < 5 ; j++){
5                 System.out.printf("(%d, %d)\n", i, j);
6                 if(i == 1 && j == 3) break;
7             }
8         }
9         System.out.println("End");
10    }
11 }
```

```
(0, 0)
(0, 1)
(0, 2)
(0, 3)
(0, 4)
(1, 0)
(1, 1)
(1, 2)
(1, 3)
(2, 0)
(2, 1)
(2, 2)
(2, 3)
(2, 4)
```

End

The **break** [**label**] Statement

- Forces a break of the loop statement immediately following the label.
- Labels are typically used with **for** and **while** loops, when there are nested loops and there is a need to identify which loop to break.
- To label a loop or a statement, place the label statement immediately before the loop or statement being labeled, as follows:

```
scanScoreTable:
```

```
for (int r = 0; r < size ; r++) { //Labeled  
    for (int c = 1; c <= 18; c++) {  
        System.out.println("R:\" + r + " C:\" + c);  
        break scanScoreTable; //Exit loops  
    }  
}
```

The **break** [label] Statement(Cont.)

```
1 public class Test{
2     public static void main(String[] args) {
3         outer :
4         for(int i = 0 ; i < 3 ; i++){
5             for(int j = 0 ; j < 5 ; j++){
6                 System.out.printf("(%d, %d)\n", i, j);
7                 if(i == 1 && j == 3) break outer;
8             }
9         }
10        System.out.println("End");
11    }
12 }
```

(0, 0)
(0, 1)
(0, 2)
(0, 3)
(0, 4)
(1, 0)
(1, 1)
(1, 2)
(1, 3)
End

The **break** [label] Statement(Cont.)

```
2 public class LabelDemo {  
3     public static void main(String[] args) {  
4         outer :  
5         for ( int i = 0 ; i < 3 ; i++ ) {  
6             System.out.print("Line : " + i + " -> ");  
7             for ( int j = 0 ; j < 10 ; j++ ) {  
8                 if ( j == 5 ) {  
9                     break outer;  
10                }  
11                System.out.print(j + " ");  
12            }  
13        }  
14        System.out.println("\nProgram is Over...");  
15    }  
16 }
```

Line : 0 -> 0 1 2 3 4
Program is Over...

The `continue` Statement

- Permits to end a loop iteration.
- Used inside `while`, `for`, and `do` loops only.
- Should be used only when the alternative code is much more complex.

```
for (int i = 0; i < Player.getNumPlayers(); i++) {  
    Player p = players[i];  
    if (!p.getUnderParFlag( ))  
        continue;  
    else  
        p.displayUnderPar( );  
}
```

The `continue` Statement (Cont.)

```
1 public class Test{
2     public static void main(String[] args) {
3         for(int i = 0 ; i < 10 ; i++){
4             if(i == 5) continue;
5             System.out.print(i + "\t");
6         }
7         System.out.println("End");
8     }
9 }
```

0 1 2 3 4 6 7 8 9 End

The `continue [label]` Statement

```
1 public class Test{
2     public static void main(String[] args) {
3         outer :
4         for(int i = 0 ; i < 3 ; i++){
5             for(int j = 0 ; j < 5 ; j++){
6                 if(i == 1 && j == 3) continue outer;
7                 System.out.printf("(%d, %d)\n", i, j);
8             }
9         }
10        System.out.println("End");
11    }
12 }
```

(0, 0)
(0, 1)
(0, 2)
(0, 3)
(0, 4)
(1, 0)
(1, 1)
(1, 2)
(2, 0)
(2, 1)
(2, 2)
(2, 3)
(2, 4)
End

The **break**, **continue** label Statement

```
2  /*
3   Environment : Windows XP Service Pack 3, EditPlus 3.31
4   Reference : 남궁성, 『Java의 정석 2nd Edition』 (서울:도우출판, 2009), p.105
5  */
6
7  public class BreakContinueLabelDemo{
8      public static void main(String[] args) {
9          Loop1 :
10             for(int i = 2 ; i <= 9 ; i++){
11                 for(int j = 1 ; j <= 9 ; j++){
12                     if(j == 5)
13                         break Loop1;
14                     //break;
15                     //continue Loop1;
16                     //continue;
17                     System.out.println(i + " * " + j + " = " + i * j);
18                 } //end of for j
19             } //end of for i, end of Loop1
20     }
21 }
```

break, continue

```
1 import java.io.*;
2 public class BreakContinueDemo {
3     public static void main(String[] args) throws IOException{
4         char [] pass = {'A', 'B', 'C', 'D'};
5         System.out.print("Enter your Password : ");
6         BufferedReader br = null;
7         br = new BufferedReader(new InputStreamReader(System.in));
8         String userValue = br.readLine().trim(); //사용자가 입력한 값
9         char [] userArray = userValue.toCharArray();
10        int i;
11        for ( i = 0 ; i < userArray.length; i++ ) {
12            if ( pass[i] == userArray[i] ) continue; // 사용자가 입력한 값과 원값과 하나씩 비교
13            else break; //한개라도 틀리면 바로 break;
14        }
15        if ( i == 4) System.out.println("Success");
16        else System.out.println("Failure");
17    }
18 }
```

Enter your Password : ABCd
Failure