

Computer Language



Conditions & Loop



Agenda

- Condition
- Loop



Condition

Loop

Program's Flow

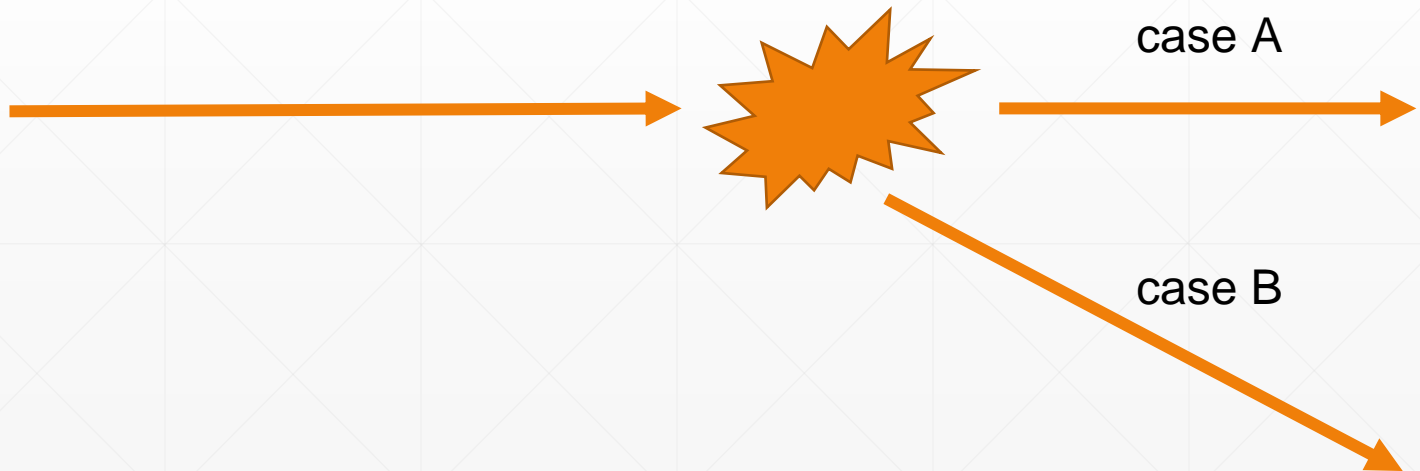
■ Our simple program so far

- Take input from the user
- Perform some operations
- Print the results

Normal, sequential flow



■ What if we want to handle various cases?



Program's Flow (cont'd)

■ How can we define a condition?

- Relational operators (==, !=, >, <, <=, >=)
- Conditional operators (||, &&, !)

■ How can we make a branch based on the condition?

- IF statements
- Switch statements
- Conditional statements (logic) execute a certain section of code only if a particular test (condition) evaluates to *true*

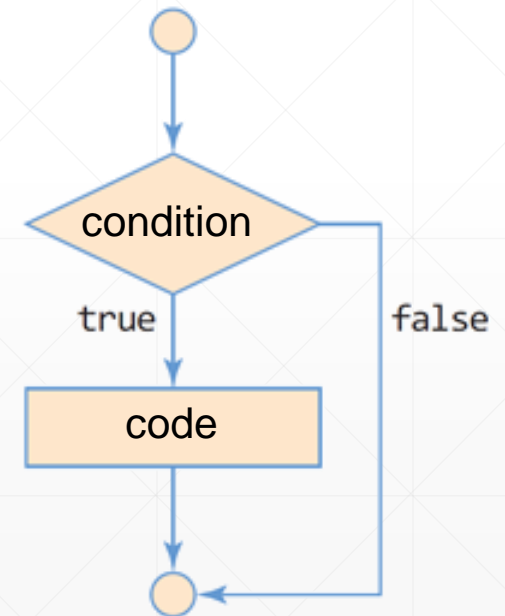
Condition: IF

■ Simple IF statement

- If an evaluation of the condition is true, then execute a code section
- Brackets can be omitted, if a code to be executed is a single line

```
if(n%2 == 0) {  
    System.out.print(n);  
    System.out.println("is an even number.");  
}  
  
if(score >= 80 && score <= 89)  
    System.out.println("Your grade is B!");
```

```
if(condition){  
  
    ... code to be executed ...  
  
}
```



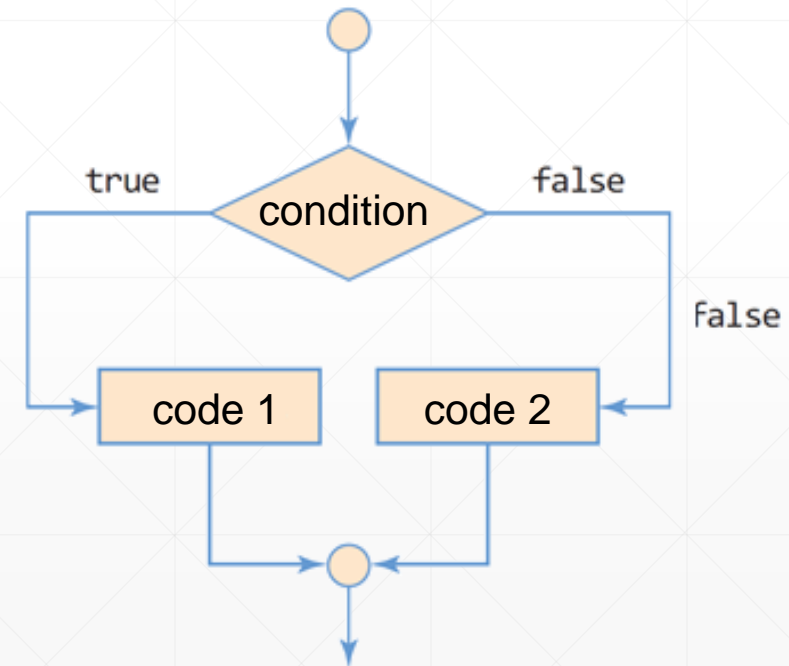
Condition: IF (cont'd)

■ IF-else statement

- If an evaluation of the condition is true, then execute a code section 1
- If false, then execute a code section 2

```
int score = 55;  
  
if(score >= 90)  
    System.out.println("A+!");  
else  
    System.out.println("F!");
```

```
if(condition){  
    code section 1  
}  
else {  
    code section 2  
}
```



Condition: IF (cont'd)

■ Ternary statement

- Condition ? opr2 : opr3
- If an evaluation of the condition is true, then the result will be opr2
- If false, then result will be opr3
- Alternative of IF-else statement

```
int x = 5;  
int y = 3;
```

```
int s;  
if(x>y)  
    s = 1;  
else  
    s = -1;
```



```
int s = (x>y) ? 1 : -1;
```


Condition: IF (cont'd)

■ Ternary statement

➤ Example)

```
int a = 3, b = 5;
```

```
System.out.println("The diff between two numbers is " + ((a>b)?(a-b):(b-a)));
```

➤ How to implement this statement using if-else statement?

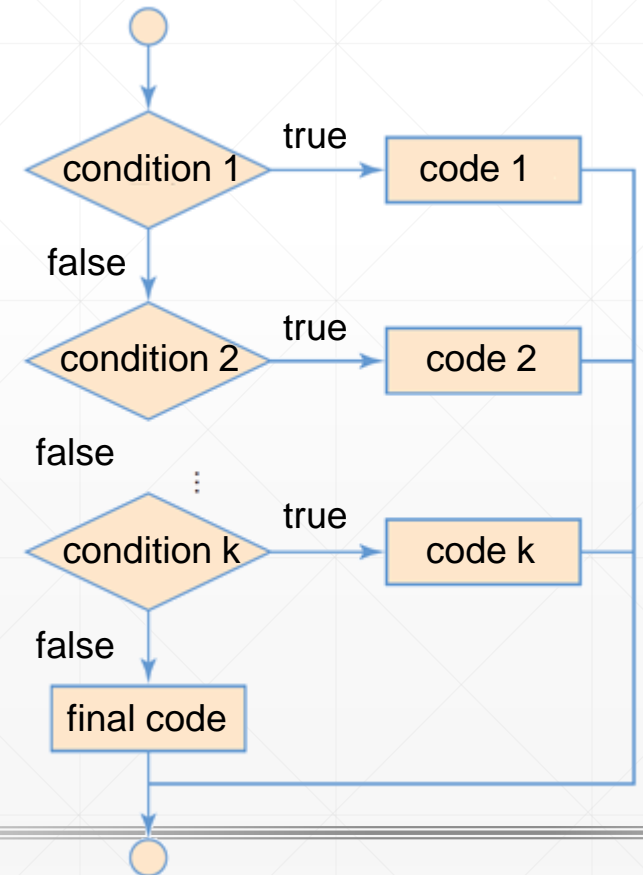
Condition: IF (cont'd)

■ Multiple if-else statement

- In case we need to have multiple branches
- The conditions are mutually exclusive (only one section will be executed)
- If – (else if)* - else

```
if(score >= 90) { // 90 <= score
    grade = 'A';
}
else if(score >= 80) { // 80 <= score < 90
    grade = 'B';
}
else if(score >= 70) { // 70 <= score < 80
    grade = 'C';
}
else if(score >= 60) { // 60 <= score < 70
    grade = 'D';
}
else { // < 60
    grade = 'F';
}
```

```
if(condition){
    code section 1
}
else if(condition2){
    code section 2
}
else if(condition3){
    code section 3
}
...
else {
    final section
}
```



Condition: IF (cont'd)

■ Multiple if-else statement

➤ Example) what happens we just use multiple if statements?

```
if(score >= 90) { // 90 <= score
    grade = 'A';
}
if(score >= 80) { // 80 <= score < 90
    grade = 'B';
}
if(score >= 70) { // 70 <= score < 80
    grade = 'C';
}
if(score >= 60) { // 60 <= score < 70
    grade = 'D';
}
else { // < 60
    grade = 'F';
}
```

Condition: IF (cont'd)

■ Nested if statement

➤ If statement can be used inside another if statement

➤ Example)

Nested

```
Scanner scanner = new Scanner(System.in);

System.out.print("Input your score (0~100): ");
int score = scanner.nextInt();

System.out.print("Input your grade (1~4): ");
int year = scanner.nextInt();

if (score >= 60) { // greater than or equal to 60
    if (year != 4)
        System.out.println("PASS!"); // if not a senior, pass!
    else if (score >= 70)
        System.out.println("PASS!"); // if senior && greater than or equal to 70, pass!
    else
        System.out.println("FAIL!"); // if senior && less than 70, fail!
} else // less than 60, fail!
    System.out.println("FAIL!");

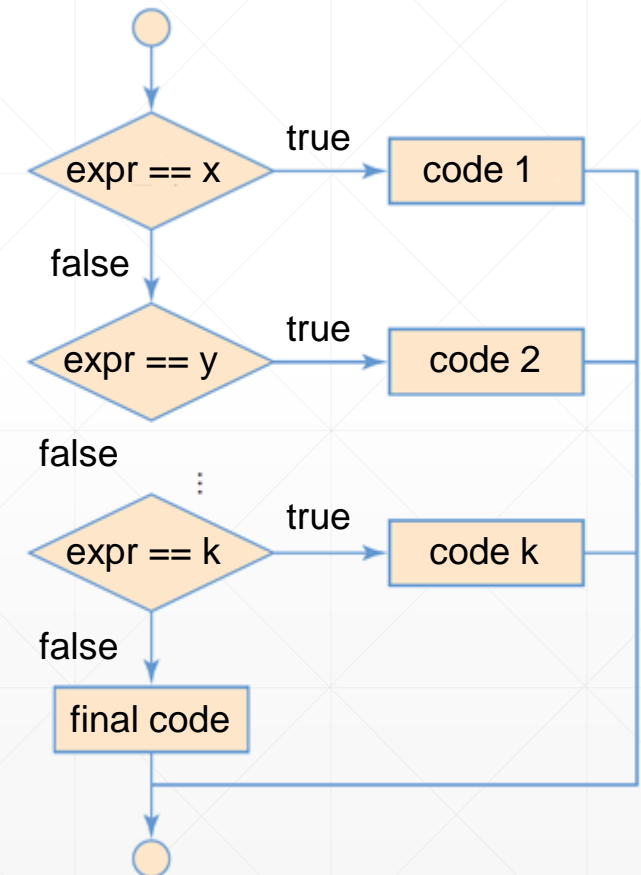
scanner.close();
```

Condition: Switch

■ Switch-case statement

- Evaluates if a given expression matches each case value
- Case value
 - Only char, integer, String literals are allowed
 - Floating-point literal is not allowed
- If matched, execute its code block
 - **Break** literally breaks the switch statement
- If nothing matched, execute a code block of **the default section**

```
switch (expression){  
    case x:  
        code block 1  
        break;  
  
    case y:  
        code block 2  
        break;  
  
    ...  
    default:  
        final code block  
}  
}
```



Condition: Switch (cont'd)

■ Example)

```
int b;
switch(b%2) {
    case 1 : ...; break; // integer literal is allowed
    case 2 : ...; break;
}

char c;
switch(c) {
    case '+' : ...; break; // char literal is allowed
    case '-' : ...; break;
}

String s = "Yes";
switch(s) {
    case "Yes" : ...; break; // String literal is allowed
    case "No" : ...; break;
}
```

```
switch(a) {
    case a :           // Error!
    case a > 3 :        // Error!
    case a == 1 :       // Error!
}
```

Condition: Switch (cont'd)

■ Example)

```
Scanner scanner = new Scanner(System.in);


char grade;
System.out.print("Input your score (0~100): ");
int score = scanner.nextInt();
switch (score / 10) {
    case 10: // score = 100
    case 9: // score 90~99
        grade = 'A';
        break;
    case 8: // score 80~89
        grade = 'B';
        break;
    case 7: // score 70~79
        grade = 'C';
        break;
    case 6: // score 60~69
        grade = 'D';
        break;
    default: // score < 60
        grade = 'F';
}
System.out.println("Your grade is " + grade);
```

Condition: Switch (cont'd)

■ Example) What happens if we remove break?

```
Scanner scanner = new Scanner(System.in);

char grade;
System.out.print("Input your score (0~100): ");
int score = scanner.nextInt();
switch (score / 10) {
    case 10: // score = 100
    case 9: // score 90~99
        grade = 'A';
    case 8: // score 80~89
        grade = 'B';
    case 7: // score 70~79
        grade = 'C';
    case 6: // score 60~69
        grade = 'D';
    default: // score < 60
        grade = 'F';
}
System.out.println("Your grade is " + grade);
```

Conditions **Loop**

Why We Need a Loop?

- Suddenly, wanna make a multiplication table!

➤ Let's compute 1×1 , 1×2 , ..., 1×9 !

- How to do this?

```
System.out.println("1x1=" + 1*1);  
System.out.println("1x2=" + 1*2);  
System.out.println("1x3=" + 1*3);  
System.out.println("1x4=" + 1*4);  
System.out.println("1x5=" + 1*5);  
System.out.println("1x6=" + 1*6);  
System.out.println("1x7=" + 1*7);  
System.out.println("1x8=" + 1*8);  
System.out.println("1x9=" + 1*9);
```

Ok, I can do this.

Why We Need a Loop? (cont'd)

■ But, a multiplication table consists of one, two, ..., nine times table!

■ How to do this?

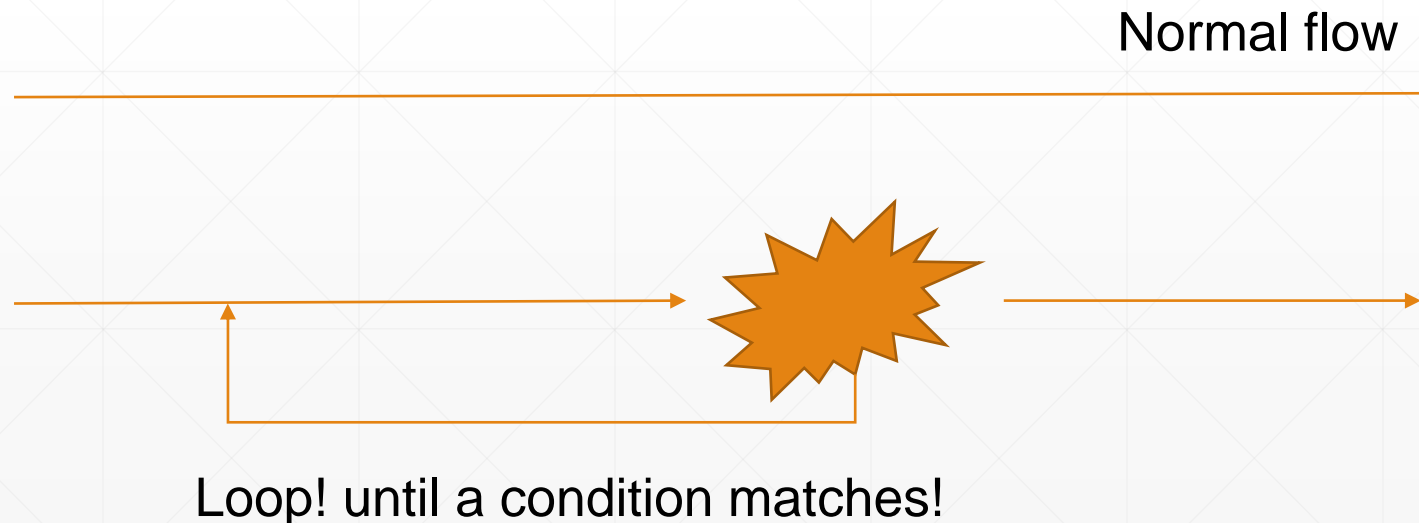
```
System.out.println("1x1=" + 1*1);  
System.out.println("1x2=" + 1*2);  
System.out.println("1x3=" + 1*3);  
System.out.println("1x4=" + 1*4);  
System.out.println("1x5=" + 1*5);  
System.out.println("1x6=" + 1*6);  
System.out.println("1x7=" + 1*7);  
System.out.println("1x8=" + 1*8);  
System.out.println("1x9=" + 1*9);
```

```
System.out.println("2x1=" + 2*1);  
System.out.println("2x2=" + 2*2);  
System.out.println("2x3=" + 2*3);  
System.out.println("2x4=" + 2*4);  
System.out.println("2x5=" + 2*5);  
System.out.println("2x6=" + 2*6);  
System.out.println("2x7=" + 2*7);
```

Umm... am I doing correct programming?

Why We Need a Loop? (cont'd)

- How can we do the same/similar tasks iteratively?
- Use Loop statements
 - For statement
 - While statement
 - Do-while statement

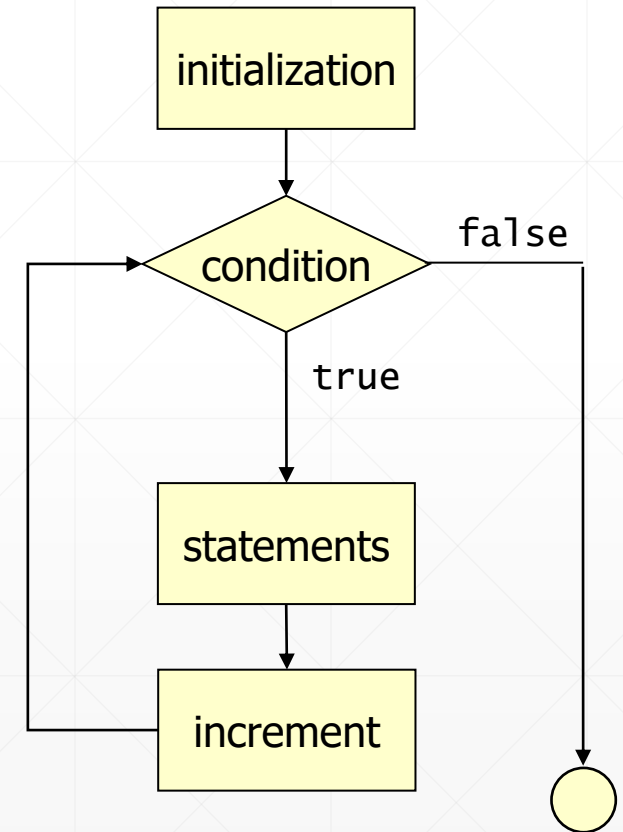


Loop: For

■ Most frequently used loop statement

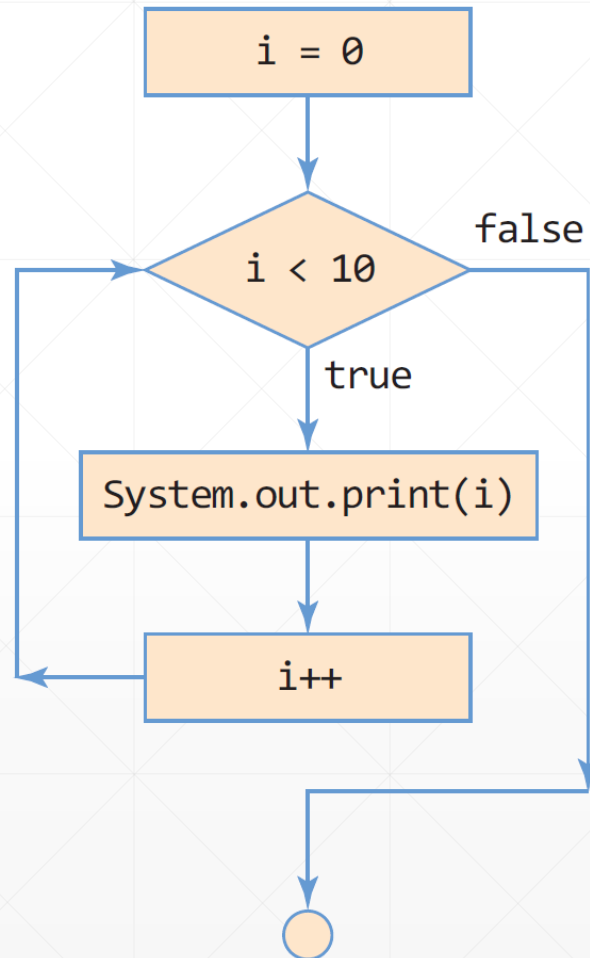
- The initialization expression initializes the loop
- When the condition evaluates to false, the loop terminates
 - Generally, used with counting
- The increment is invoked after each iteration through the loop
 - It is acceptable for this expression to increment or decrement a value

```
for (initialization; condition; increment) {  
    ... statement(s) ...  
}
```



Loop: For (cont'd)

■ Example)



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

0123456789

Loop: For (cont'd)

■ Example)

```
for(initialization; true; increment) { // if a condition is true, then this is an infinite loop  
.....  
}
```

```
for(initialization; ; increment) { // if a condition is empty, recognize it as true, so this is also an infinite loop  
.....  
}
```

```
// initialization and increment can have multiple statements, separated by ','  
for(i=0; i<10; i++, System.out.println(i)) {  
.....  
}
```

```
// It is possible to declare a local variable inside the for loop  
for(int i=0; i<10; i++) { // variable i can be only used inside this loop  
.....  
}
```

Loop: For (cont'd)

■ Example) print from 0 to 9

```
int i;  
for(i = 0; i < 10; i++) {  
    System.out.print(i);  
}
```

```
int i;  
for(i = 0; i < 10; i++)  
    System.out.print(i);
```

■ Example) summate from 0 to 100

```
int sum = 0;  
for(int i = 0; i <= 100; i++)  
    sum += i;
```

```
int i, sum;  
for(i = 0, sum=0; i <= 100; i++)  
    sum += i;
```

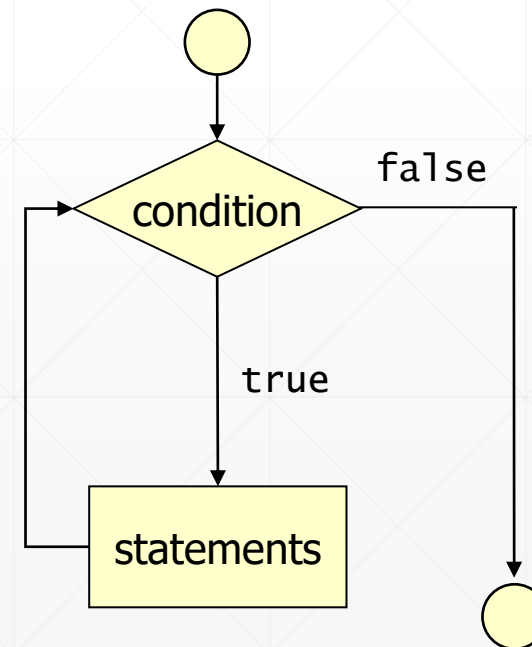
```
int sum = 0;  
for(int i = 100; i >= 0; i--)  
    sum += i;
```


Loop: While

■ Another loop statement

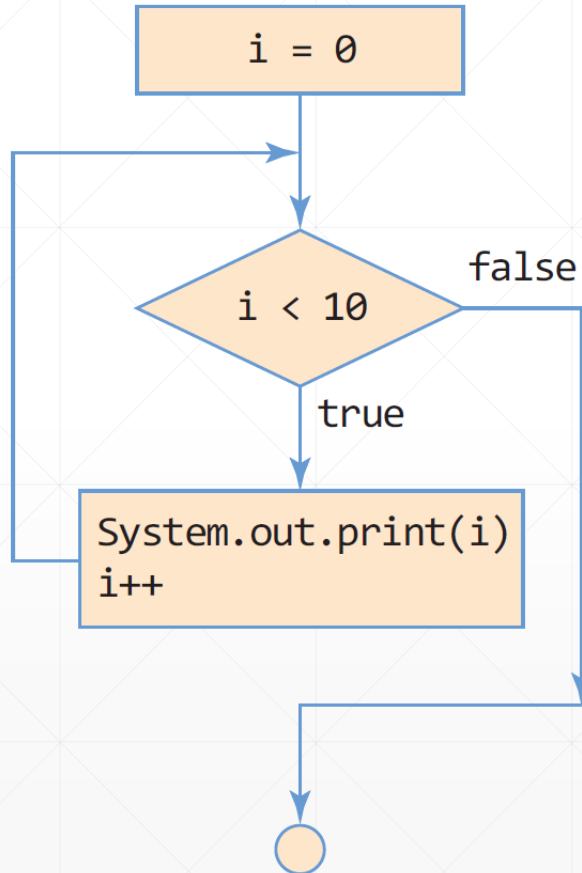
- While statement evaluates expression, which must return a boolean value
- If the expression evaluates to true, the while statement executes the statement(s) in the while block
- While statement continues testing the expression and executing its block until the expression evaluates to false

```
while (condition) {  
    ... statement(s) ...  
}
```



Loop: While (cont'd)

■ Example)



```
i = 0;  
while(i<10) {  
    System.out.print(i);  
    i++;  
}
```

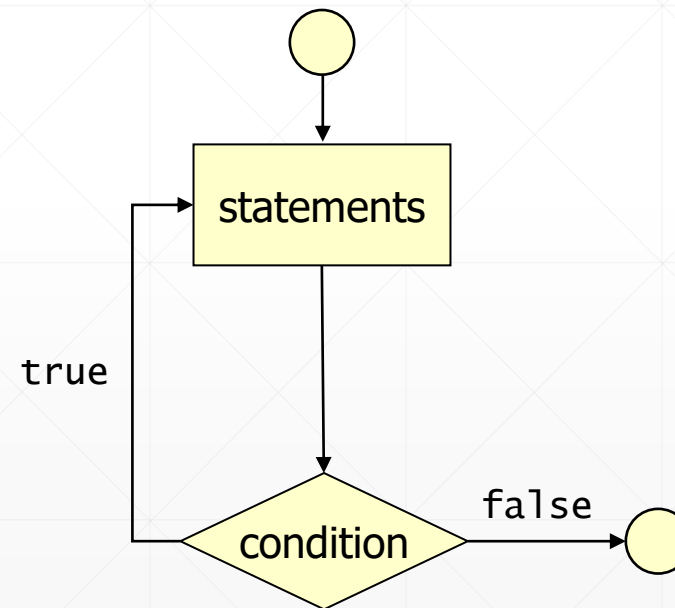
0123456789

Loop: Do-While

■ Another while statement

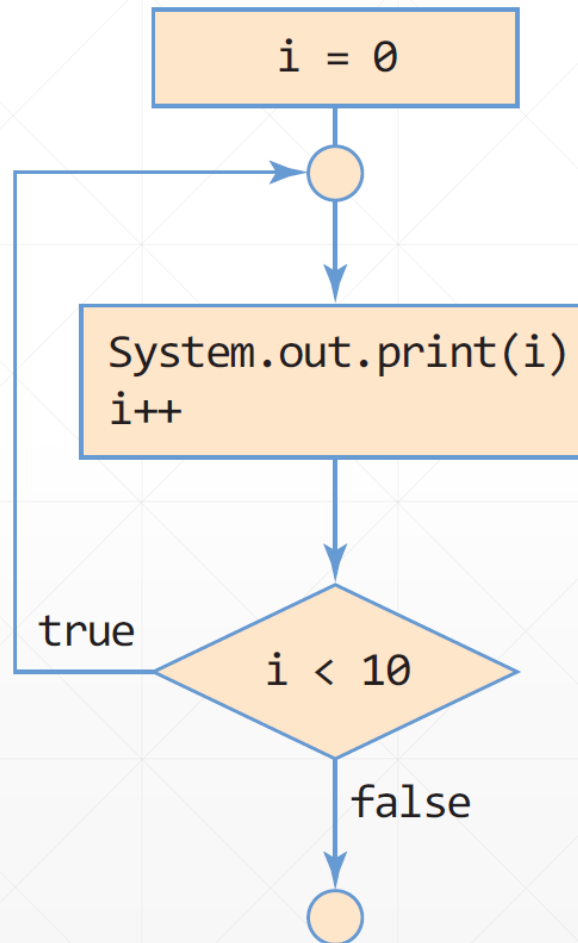
- Do-while evaluates its expression at the bottom of the loop instead of the top!
- The statements within the do block are always executed **at least once**

```
Do {  
    ... statement(s) ...  
} while (condition);
```



Loop: Do-While (cont'd)

■ Example)



```
i = 0;  
do {  
    System.out.print(i);  
    i++;  
} while(i < 10);
```

0123456789

Loop: Summary

```
for(initialization; condition ; increment)
{
    statements
}
```

```
while( condition )
{
    statements
}
```

```
do
{
    statements
} while( condition );
```

Loop: Nested Loop

- Similar to the nested if-statement, loop statements can be used in a nested way

➤ i.e., Loop inside another loop

```
Get in a department store;  
while (any interesting store??){  
    Get in the store;  
    Look around the store;  
    while (any interesting toy){  
        Take a closer look at the toy;  
        if (Love it?) Buy it!;  
        Move to another toy;  
    }  
    Get out of the store;  
}  
Leave the department store;
```

Diagram illustrating nested loops:

- Outer Loop (Loop):
 - Get in a department store;
 - while (any interesting store??){
 - Inner Loop (Loop):
 - Get in the store;
 - Look around the store;
 - while (any interesting toy){
 - Take a closer look at the toy;
 - if (Love it?) Buy it!;
 - Move to another toy;
 - Get out of the store;
- Leave the department store;



Loop: Nested Loop (cont'd)

- Similar to the nested if-statement, loop statements can be used in a nested way
 - i.e., Loop inside another loop

```
for(int i=0; i<100; i++) { // sum up the scores of 100 schools
    ....
    for(int j=0; j<10000; j++) { // sum up the scores of 10,000 students, for each school
        ....
        ....
    }
    ....
}
```

Doubly nested loop to add up the scores of 100 schools,
each of which has 10,000 students.

Loop: Nested Loop (cont'd)

- Example) print a multiplication table using double nested loop


```
public class NestedLoop {  
    public static void main(String[] args) {  
        for(int i=1; i<10; i++) { // from 1 times table to 9 times table  
            for(int j=1; j<10; j++) { // for each table  
                System.out.print(i + "*" + j + "=" + i*j); // print multiplication  
                System.out.print(' '); // print tab  
            }  
            System.out.println(); // nextline  
        }  
    }  
}
```

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

Loop: Continue & Break

■ Oh...Please...

```
Get in a department store;  
while (any interesting store??){  
    Get in the store;  
    Look around the store;  
    while (any interesting toy?){  
        Take a closer look at the toy;  
        if (Love it?) Buy it!;  
        Move to another toy;  
    }  
    Get out of the store;  
}  
STOP IT!!!!!!!!!!!!  
Leave the department store;
```



Loop: Continue & Break (cont'd)

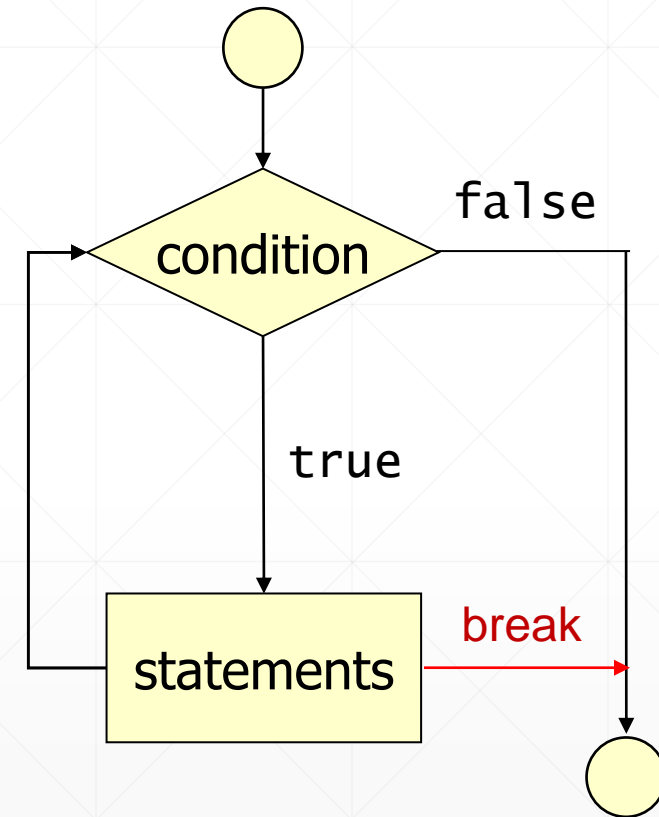
■ Break statement

- Can be used to break the loop!
- Only applied to the current loop

■ Example)

```
for(int i=0;i<5;i++){  
    if(i==3) break;  
    System.out.println(i);  
}
```

```
int j=0;  
while(j<5){  
    if(j==3) break;  
    System.out.println(j);  
    j++;  
}
```



Loop: Continue & Break (cont'd)

■ Break in the nested loop

- Only applied to the current loop

■ Example)

```
while(true){  
    while(true){  
        break;  
    }  
    System.out.println("Here!");  
}  
System.out.println("There!");
```

← Infinite loop!

← Break the loop!

Where should we go?

Loop: Continue & Break (cont'd)

■ Oh...Please...

Get in a department store;

while (any interesting store??){

Get in the store;

Look around the store;

Hmm, not interesting!
Next!

while (any interesting toy){

Take a closer look at the toy;

if (Love it?) Buy it!;

Move to another toy;

}

Get out of the store;

}

Leave the department store;



Loop: Continue & Break (cont'd)

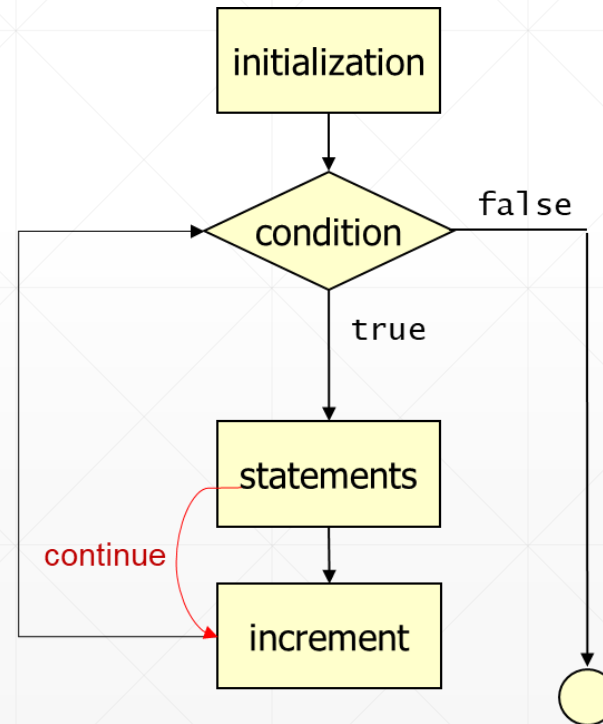
■ Continue statement

- Skip the remaining statements
- Move to check the condition of the loop
- Only applied to the current loop

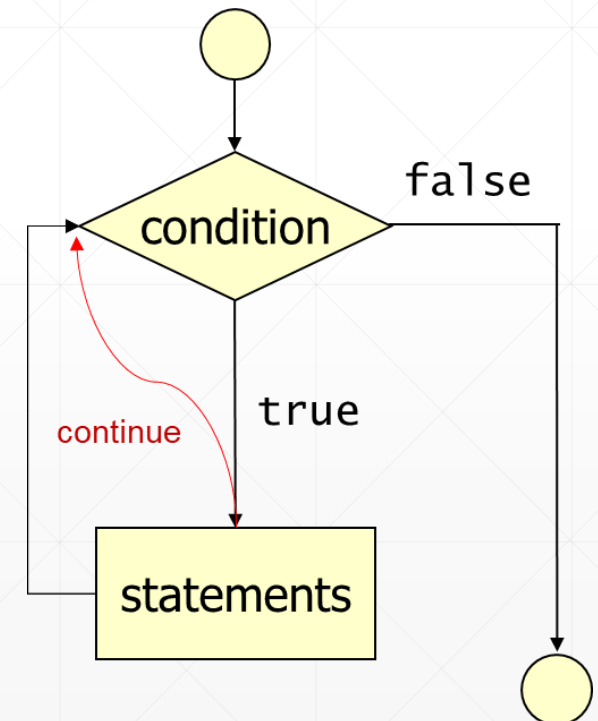
■ Example)

```
for(int i=0;i<5;i++){  
    if(i%2==0) continue;  
    System.out.println(i);  
}
```

```
int j=0;  
while(j<5){  
    if(j%2==0) { j++; continue; }  
    System.out.println(j);  
    j++;  
}
```



continue in for



continue in while

Loop: Continue & Break (cont'd)

■ Continue in the nested loop

- Only applied to the current loop

■ Example) `int i,j=0;`

```
while (j<10) {  
    while (j<5) {  
        System.out.println("first!");  
        j++;  
        if (j %2 == 1) continue;  
        System.out.println("second!");  
    }  
    System.out.println("third!");  
    j++;  
}  
System.out.println("fourth!");
```

Q&A

- Next week
 - Reference Types