Control Statements: Loop Control Structure

By Lim Pei Geok
Lecturer of Department of Computer Science,
Southern University College

Objectives

- To understand the flow of control in loop statements
- To use Boolean expressions to control loop statements
- To use **while, do-while, and for loop** statements to control the repetition of statements
- To know the similarities and differences of three types of loops
- To implement program control with **continue**.

Loop Statements

1. The `while` Loops
2. The `do-while` Loops
3. The `for` Loops
4. `break` and `continue`

The `while` Loop

The *while statement* has the following syntax:

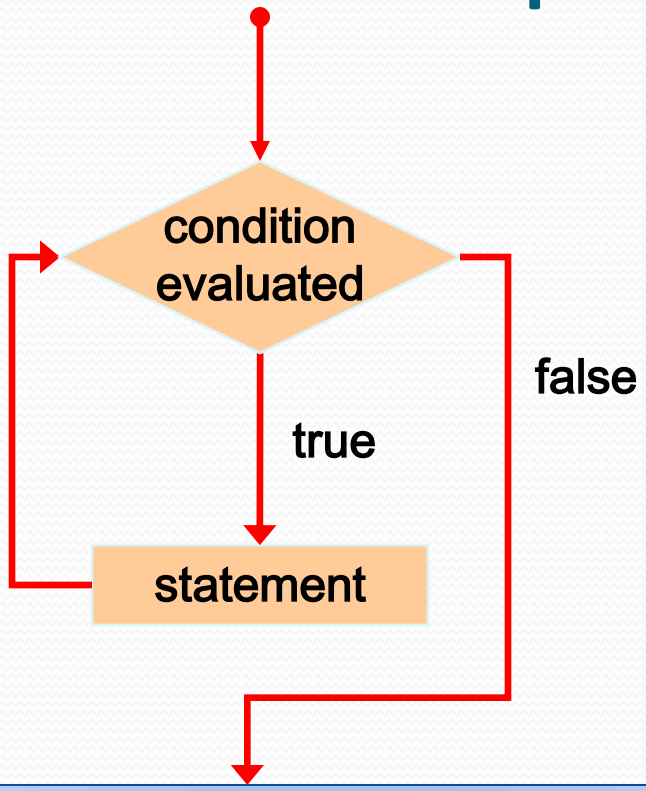
```
while ( condition )
    statement;
```

while is a reserved word

If the ***condition* is true**, the ***statement* is executed**. Then the ***condition* is evaluated again**.

The *statement* is executed repeatedly until the *condition* becomes false.

Logic of a while loop



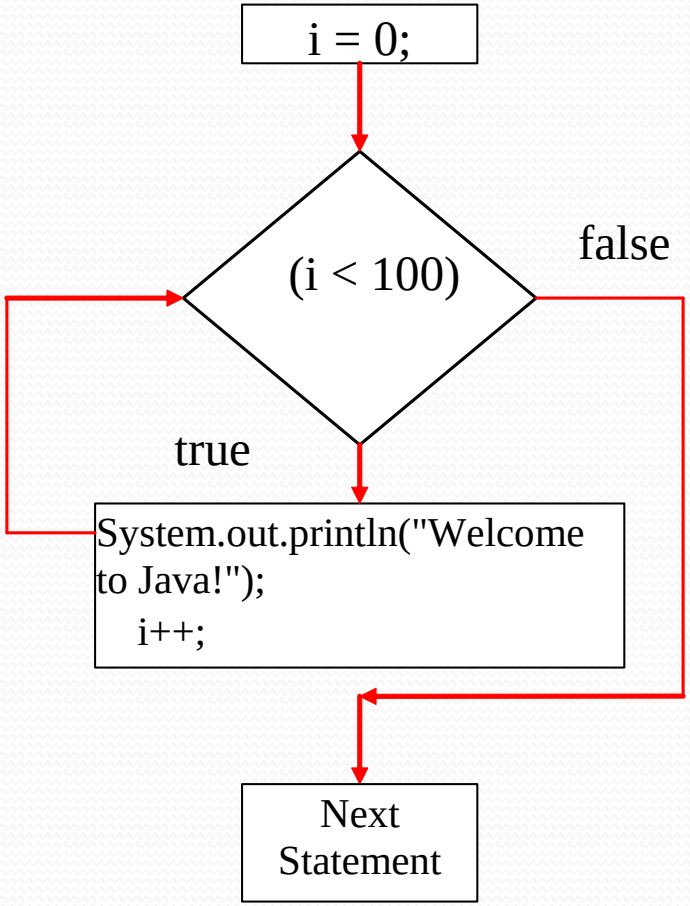
The while loop repeatedly executes the statements in the loop body /statement when the condition evaluated as true.

Example: a while Loop

```
int i = 0;
while (i < 100) {

        System.out.println("
        Welcome to Java!");

        i++;
}
```



Question (while loop)

(While loop) Write a java program to display 2 4 8 16 32.

Question (while loop)

- Write a java program to ask user to enter an exam score and display the grade of the entered exam score as follows:
- You program should keep asking the user to enter an valid score if the score entered is not in the range of 0-100.

Score	Grade
80-100	A
70-79	B
60-69	C
50-59	D
0-49	F

The do-while Loop

The *do statement* has the following syntax:

do and while are reserved words

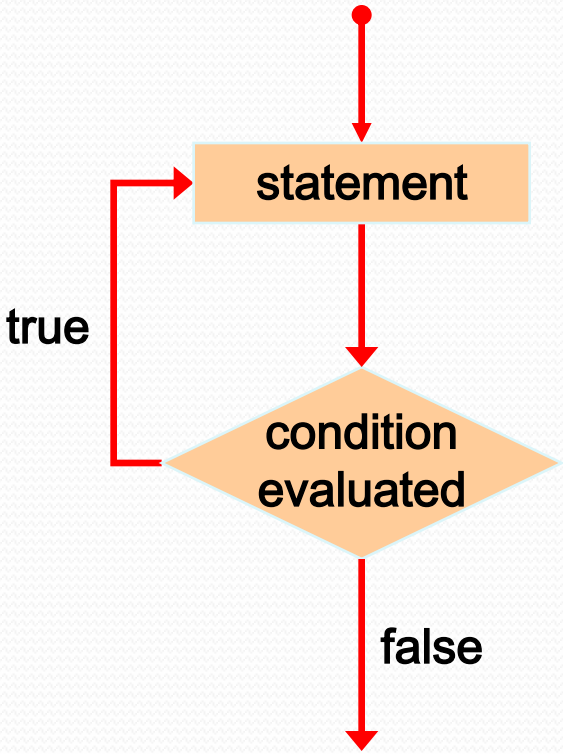
```
do
{
    statement;
}
while ( condition );
```

The ***statement* is executed once initially,** and **then** the ***condition* is evaluated**

The *statement* is executed repeatedly until the *condition* becomes false

Logic of a do-while Loop

- A do loop is similar to a while loop, except that the condition is evaluated after the body of the loop is executed
- Therefore the body of a do loop will execute at least once



Example: the do-while Loop

```
import javax.swing.JOptionPane;
public class TestDoWhile{
    public static void main(String[] args){
        int data;
        int sum=0;
        do
        {
            String dataString=JOptionPane.showInputDialog(null,
            "Enter an int value, \nthe program exits if the input is 0",
            "TestDo", JOptionPane.QUESTION_MESSAGE);

            data = Integer.parseInt(dataString);
            sum+=data;
        }while(data !=0);

        JOptionPane.showMessageDialog(null,"the sum is" + sum,
        "TestDo", JOptionPane.INFORMATION_MESSAGE);
    }
}
```

Question(do..while loop)

(do..while loop) Write a java program to display 2 4 8 16 32.

Question(do..while loop)

- Write a java program to ask user to enter an exam score and display the grade of the entered exam score as follows:
- You program should keep asking the user to enter an valid score if the score entered is not in the range of 0-100.

Score	Grade
80-100	A
70-79	B
60-69	C
50-59	D
0-49	F

The for Loop

The *for statement* has the following syntax:

Reserved word

The `initialization` is executed once before the loop begins

The `statement` is executed until the `condition` becomes false

```
for ( initialization; condition; increment )
    statement;
```

The *increment* portion is executed at the end of each iteration

The *condition-statement-increment* cycle is executed repeatedly

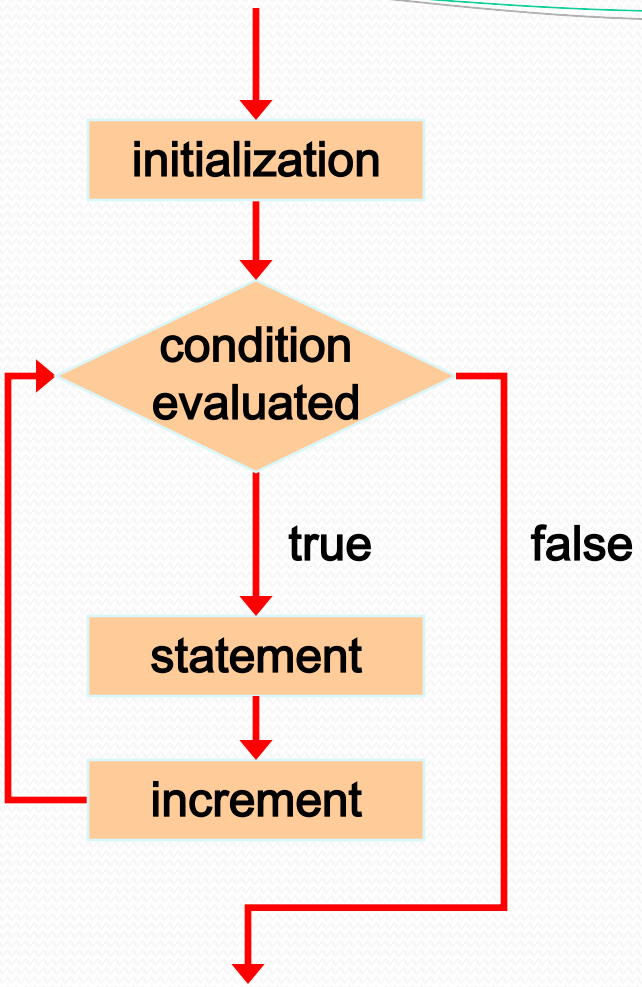
The for Loop

Syntax:

```
while ( condition )
{
    statement;
    increment;
}
```

```
for ( initialization; condition; increment )
    statement;
```

Logic of a for loop



A for loop performs an initial action once, then repeatedly executes the statements in the loop body, and performs an action after an increment when the condition evaluated as true

The for loop

- Like a `while` loop, the condition of a `for` statement is tested prior to executing the loop body
- Therefore, the body of a `for` loop will execute zero or more times
- It is well suited for executing a loop a specific number of times that can be determined in advance

Example: a while loop & a for loops

Example: A while loop

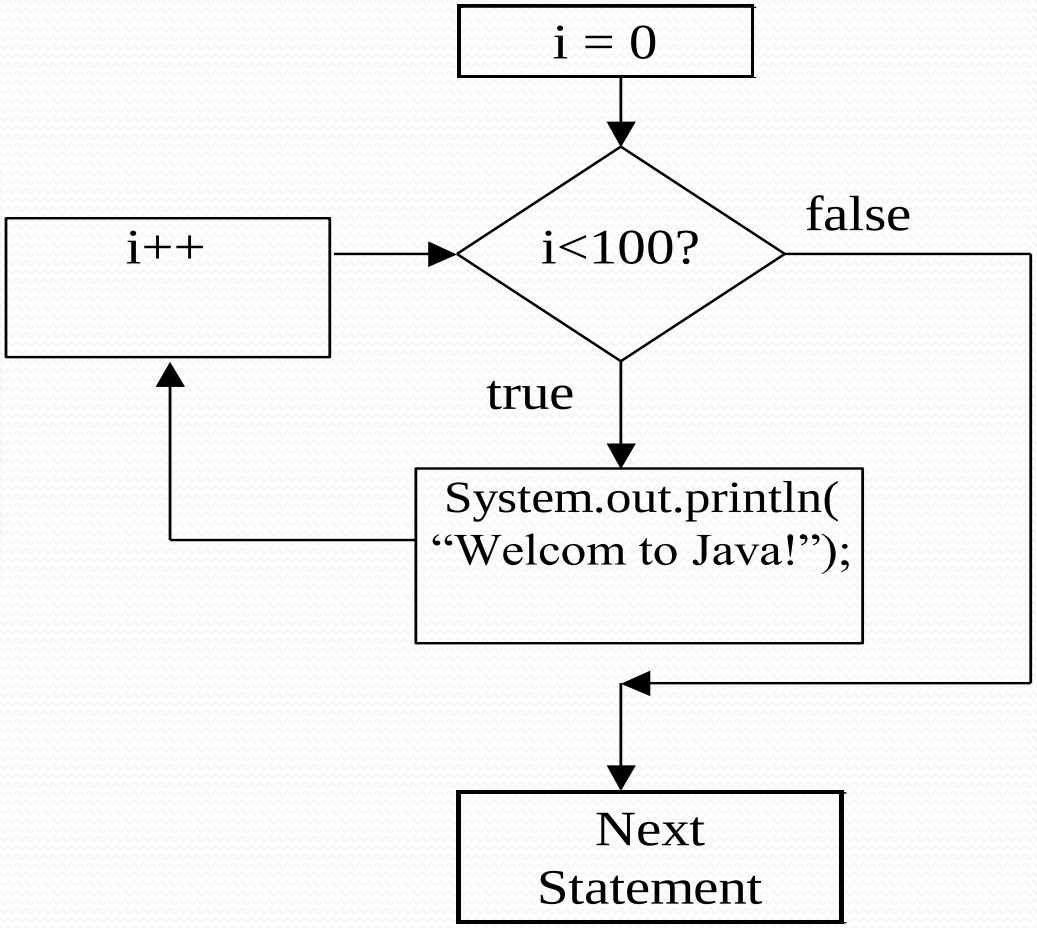
```
int i = 0;
while (i < 100) {
   System.out.println("Welcome to Java! ”
  + i);
   i++;
}
```

Example: A for loop

```
int i;
for (i = 0; i < 100; i++) {
   System.out.println("Welcome to Java! ”
  + i);
}
```

`for` Loop Example

```
int i;
for (i = 0; i<100; i++) {
  System.out.println(
     "Welcome to Java");
}
```



Question (for loop)

(for loop) Write a java program to display 2 4 8 16 32.

Which Loop to Use?

- A for loop may be used if the number of repetitions is known, as, for example, when you need to print a message 100 times.
- A while loop may be used if the number of repetitions is not known, as in the case of reading the numbers until the input is 0.
- A do-while loop can be used to replace a while loop if the loop body has to be executed before testing the continuation condition.

Caution

Adding a semicolon at the end of the <u>for</u> clause before the loop body is a common mistake, as shown below:

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

Logic error

Caution, cont.

Similarly, the following loop is also wrong:

```
int i=0;
while (i<10);
{ System.out.println("i is " + i);
  i++;
}
```

Logic error

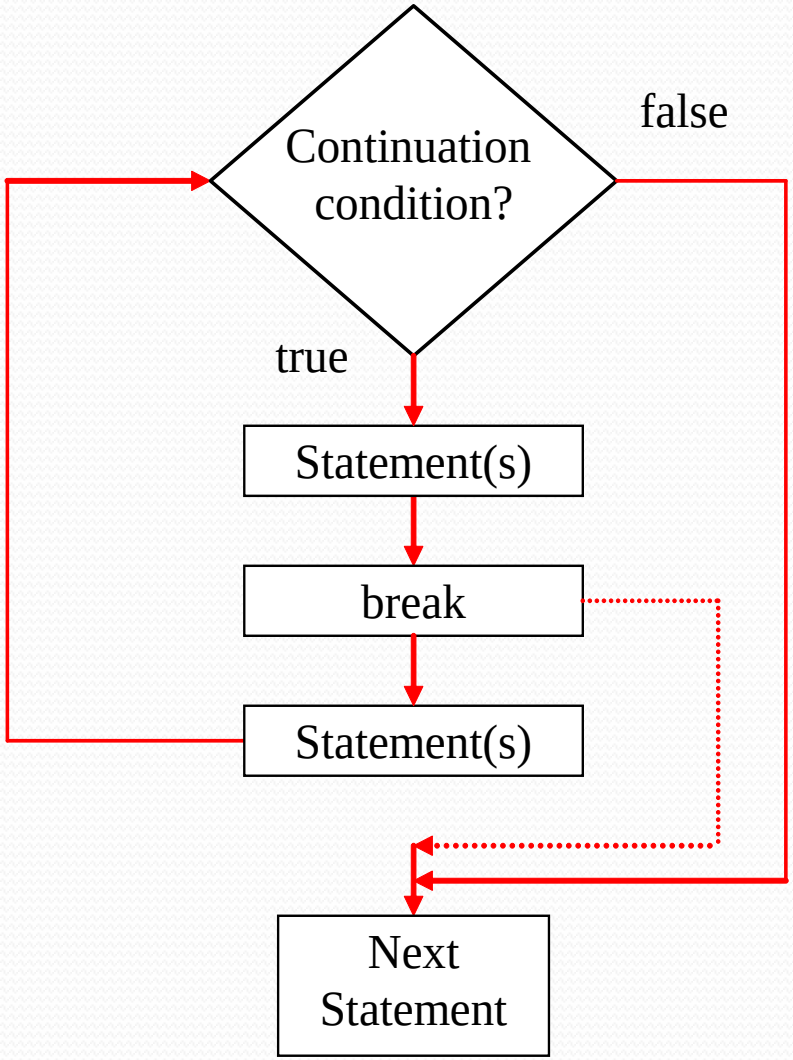
In the case of the do loop, the following semicolon is needed to end the loop.

```
int i=0;
do
{
  System.out.println("i is " + i);
  i++;
}while (i<10);
```

Correct

The `break` Keyword

- Break immediately ends the innermost loop that contains it.
- Generally used with an if statement



Example: break keyword

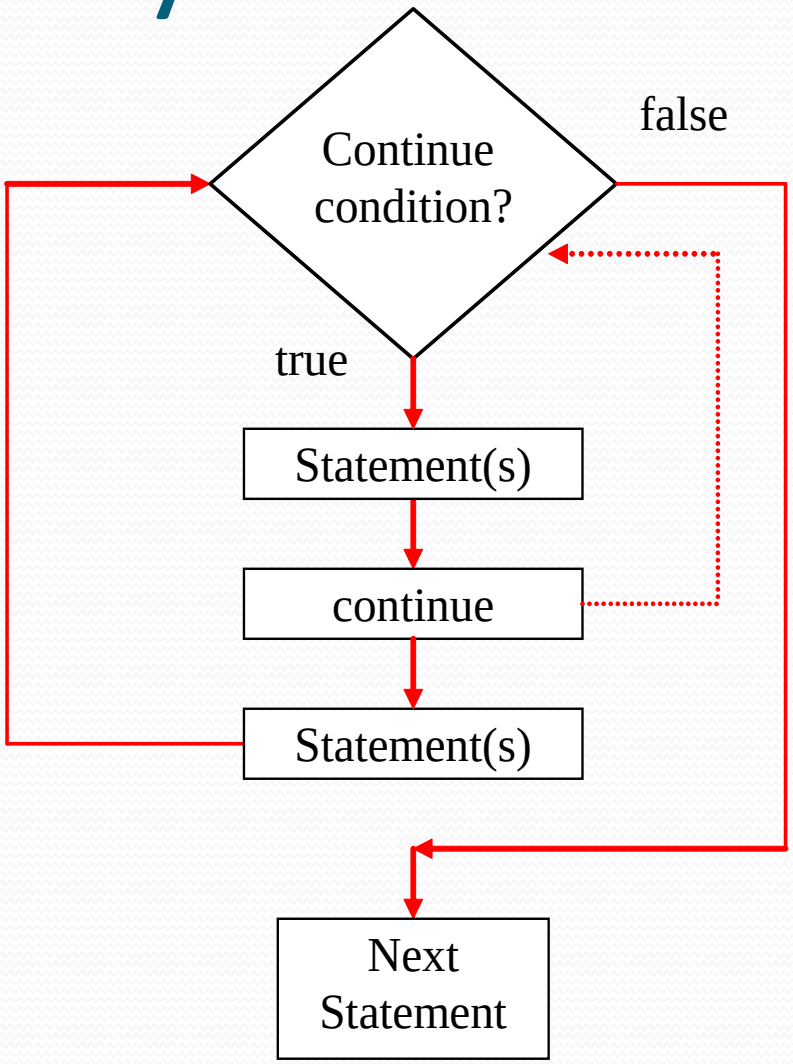
```
//add the integers from1 to 20 in this order to sum until sum is
//greater than or equal to 100
public class TestBreak{
    public static void main(String[] args)
    {
        int sum=0;
        int number=0;

        while(number < 20){
            number++;
            sum+=number;
            if(sum>=100)break;
        }
        System.out.println("The number is " + number);
        System.out.println("The sum is " + sum);
    }
}
```

The break statement in the TestBreak program forces the while loop to exit when sum is greater than or equal to 100.

The `continue` Keyword

- Continue only ends the current iteration.
- Program control goes to the end of the loop body.
- Generally used with an if statement



Example: Continue Keyword

```
//add all the integers from 1 to 20 except 10 and 11 to sum
public class TestContinue{
   public static void main(String[] args)
   {
         int sum=0;
         int number=0;

         while(number < 20){
               number++;
               if(number ==10 || number ==11)continue;
               sum+=number;
         }
         System.out.println("The sum is " + sum);
   }
}
```