# COMP-248
## Object Oriented Programming I

## Week 4: Control Flow 1

By Emad Shihab, PhD, Fall 2015,
Parts of the slides are taken from Prof. L. Kosseim
Adapted for Section EE by S. Ghaderpanah, Fall 2015

# In this chapter, we will see:

1. The `if` statement
2. The `if-else` statement
3. Relations Operators
4. Logical operators
5. Compound statements
6. Nested `if` statements
7. The `switch` statement
8. The conditional operator
9. **The `while` loop**
10. The `do-while` loop
11. The `for` loop
12. Nested loops
13. break, continue & exit

## Repetition statements (loops)

allow us to execute a statement several times

like conditional statements, they are controlled by boolean expressions

Java has 3 kinds of loops:

- the `while` loop
- the `do-while` loop
- the `for` loop

# 9- The while loop

syntax:

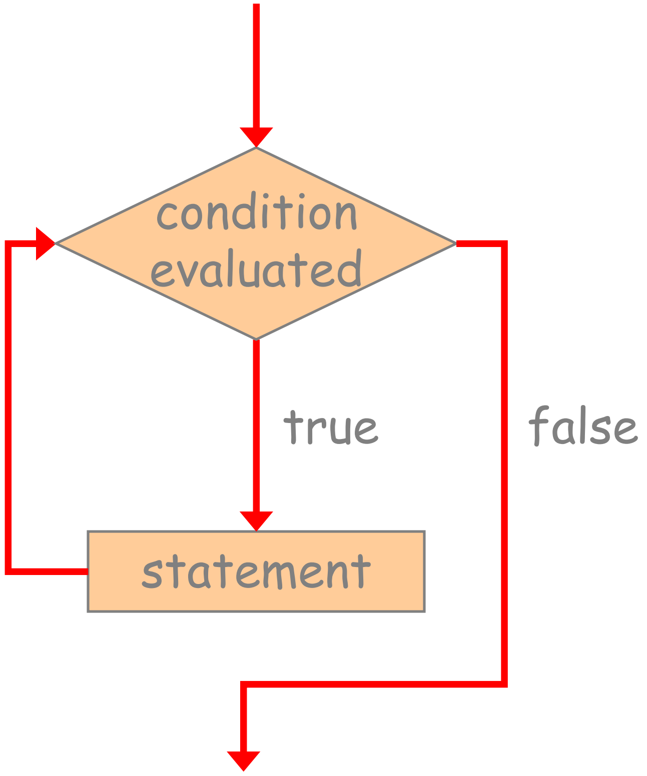
while is a reserved word

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

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

note that if the condition of a **`while`** statement is false initially, the statement is never executed

so, the body of a **`while`** loop will execute zero or more times

```
final int LIMIT = 5;
int count = 1;

while (count <= LIMIT)
{
   System.out.println(count);
   count = count + 1;
}
System.out.println("Done");
```

| LIMIT | count |
|---|---|
| | |

Trace

```
1
2
3
4
5
Done
```

Output

## Example 1

```
int remainingStars = 5;

while (remainingStars > 0)
{
    System.out.println("*");
    remainingStars--;
}
```

```
int remainingStars = 5;

while (remainingStars > 0)

    System.out.println("*");
    remainingStars--;
```

| remainingStars |
|---|
| |

Trace

| remainingStars |
|---|
| |

Trace

```
*
*
*
*
*
```

Output

```
*
*
*
*
*
......
```

Output

# Example 2

```
public class Forever
{
   public static void main(String[] args)
   {
      int count = 1;

      while (count <= 25)
      {
         System.out.println(count);
         count = count - 1;
      }

      System.out.println("Done");
   }
}
```

| count |
|---|
|  |

Trace

Output

## What will the following output?

```
boolean finished = false;
int firstInt = 3;
int secondInt = 20;
while (firstInt <= secondInt && !finished)
  if (secondInt / firstInt <= 2)
      finished = true;
  else
      firstInt++;
  System.out.println(firstInt);
```

A. 3
B. 5
C. 7
D. 8
E. 9

# Example 3: Compute average

```
Enter a series of marks (negative number to quit):
80.5 70  67  53.8 -1
The average is: 67.825
```

Output

Data needed:

Algorithm:

# Example 3:  Averager.java

```
public class Averager
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);

        System.out.println("Enter a list of nonnegative scores.");
        System.out.println("Mark the end with a negative number.");
        System.out.println("I will compute their average.");

        double next, sum = 0; // the next mark and the cummulative sum
        int count = 0; // the number of maaks read so far

        // let's read a first mark
        next = keyboard.nextDouble( );
        while(next >= 0) // while the mark is not negative
        {
            sum = sum + next; // we add it to the cummulative sum
            count++;          // we count one more mark
            next = keyboard.nextDouble( ); // we read the next mark
        }

        if (count == 0) // if the user types in no mark
            System.out.println("No scores entered."); // display a message
        else            // otherwise
        {
            double average = sum/count; // computer average
            System.out.println(count + " scores read."); // display how many marks were read
            System.out.println("The average is " + average); // display the average
        }
    }
}
```

## Example 4: max and min

same thing… but now, determine the highest and lowest marks

Data needed:

Algorithm:

## Example 4: max and min

```
double next; // the next mark
 int count=0;
 double max = 0;
 double min = 0;

 // let's read a first mark
 next = keyboard.nextDouble( );
 while(next >= 0) // while the mark is not negative
 {

     if (next > max)
             max = next;

      if (next < min)
             min = next;

    count++;
     next = keyboard.nextDouble( ); // we read the next mark
 }

 if (count == 0) // if the user types in no mark
        System.out.println("No scores entered."); // display a message
 else
 {
         System.out.println(count + " scores read."); // display how many marks were read
         System.out.println("The max is: " + max + " and the min is:" + min); // display the average
 }
```

```
max = ((next > max) ? next : max);
min = ((next < min) ? next : min);
```

# In this chapter, we will see:

1. The `if` statement
2. The `if-else` statement
3. Relations Operators
4. Logical operators
5. Compound statements
6. Nested `if` statements
7. The `switch` statement
8. The conditional operator
9. The `while` loop
10. **The `do-while` loop**
11. The `for` loop
12. Nested loops
13. break, continue & exit

# 10- The do-while loop

syntax:

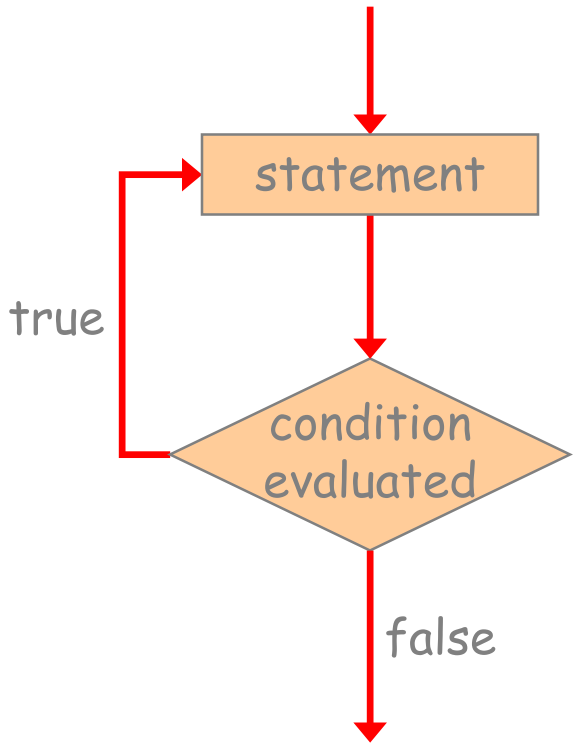
do and while are reserved words

```
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



# The do-while loop

A **do-while** 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

```
int n = 0;

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

```
0
```

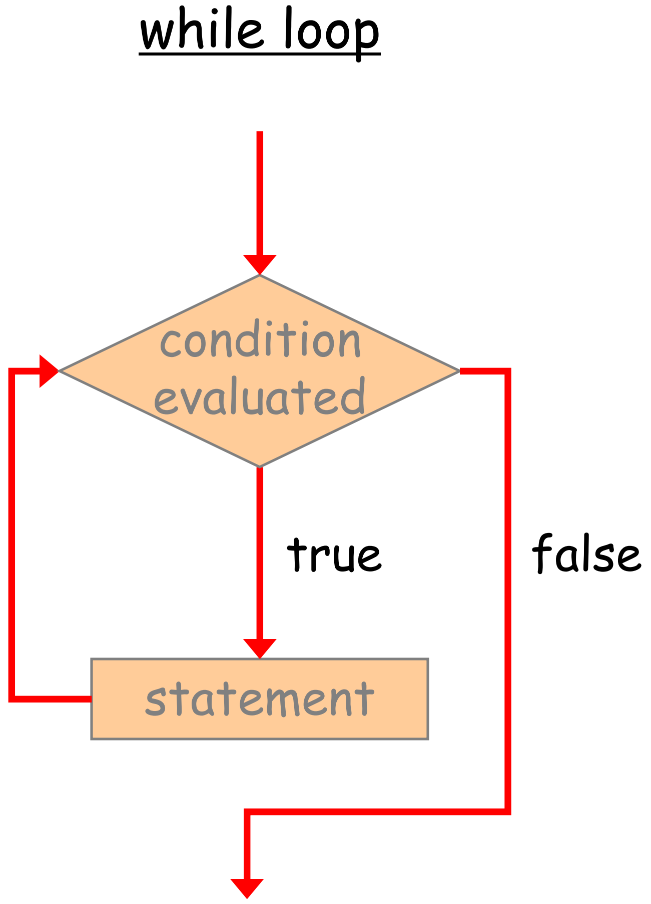
Output

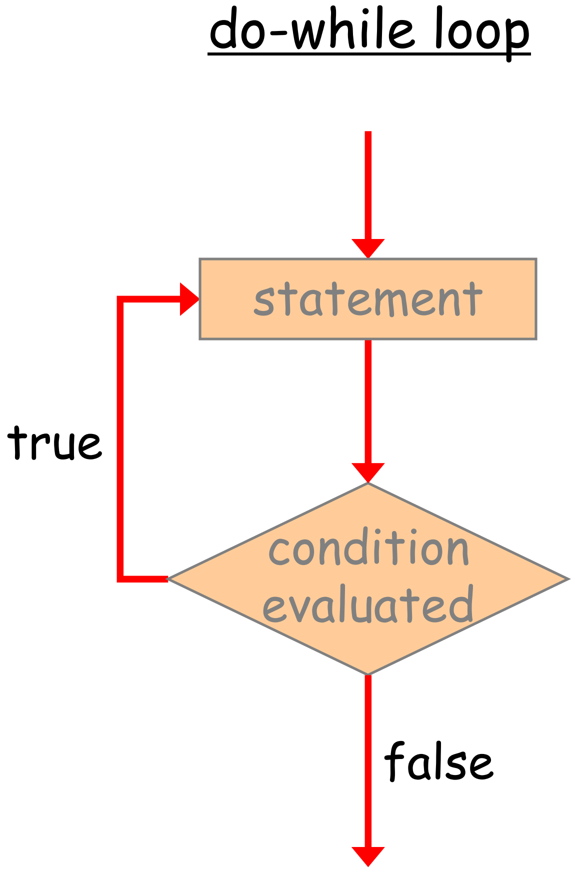
```
int n = 0;
do
{
    System.out.println("*");
    n--;
}
while (n > 0);
System.out.println(n);
```

```
*
-1
```

Output

# Comparing while and do-while





# Typical applications
## - user-controlled loop

```
String answer;
do
{
    // do the computation
    // …
    System.out.println("Do you wish to continue(yes/no)?");
    answer = myKeyboard.next();
}
while ((answer.toUpperCase()).compareTo("YES") == 0);
```

## Typical applications

## - verify user input

```
int age;
boolean valid;
do
{
   System.out.println("How old are you?");
   age = myKeyboard.nextInt();
   valid = (age > 0) && (age < 125);
   if (!valid)
      System.out.println("error! try again!");
}
while (!valid);
```

# What will the following output?

```
int beta = 5;
do
{
  switch (beta)
  {
     case 1 : System.out.print('R');
              break;
     case 2 :
     case 4 : System.out.print('O');
               break;
     case 5 : System.out.print('L');
   }
     beta--;
 }
 while (beta > 1);
System.out.print('X');
```

A. X
B. ROOLX
C. LOORX
D. LOOX
E. ROOX

## Next Topic:

1. The `if` statement
2. The `if-else` statement
3. Relations Operators
4. Logical operators
5. Compound statements
6. Nested `if` statements
7. The `switch` statement
8. The conditional operator
9. The `while` loop
10. The `do-while` loop
11. **The `for` loop**
12. Nested loops
13. break, continue & exit

## 11- The for loop

syntax:

Reserved word

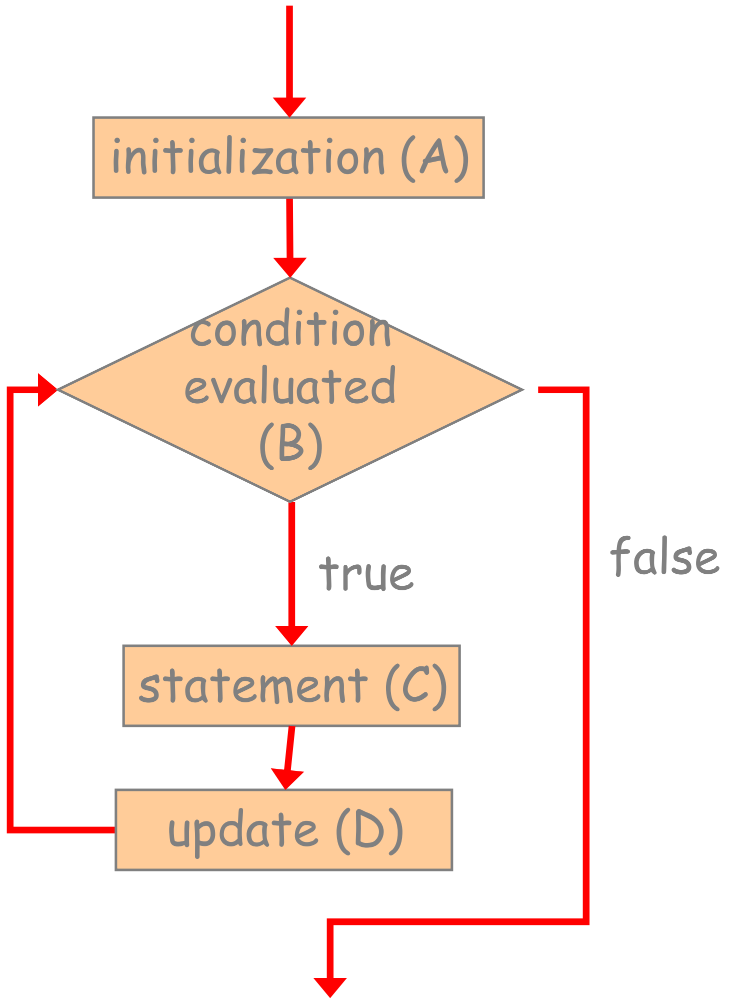
The *initialization* is executed once before the loop begins

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

```
for ( initialization (A) ; condition (B); update (D))
    statement (C);
```

The *update* portion is executed at the end of each iteration
The *condition*-*statement*-*update* cycle is executed repeatedly

## Logic of a for loop



## Example

```
int i;
for (i=1; i<=5; i++)
    System.out.print(i);
System.out.print(i);
```

Output

Trace

| i |
|---|
| |

# for versus while

A **`for`** loop is equivalent to the following **`while`**:

```
initialization;
while ( condition )
{
    statement;
    update;
}
```

## More examples

```
for (int i=0; i<0; i--)

    System.out.print("hello");
```

Output

```
for (int i=0; i<=0; i--)
    System.out.print("hello");
```

Output

## Example: Display multiples

```
Enter a positive value:  10
Enter an upper limit:  95
Multiples of 10 between 10 and 95:
10 20 30 40 50
60 70 80 90
```

Output

Data needed:

Algorithm:

## Example: Display multiples Multiples.java

```
final int PER_LINE = 5;
int value, limit, mult, count = 0;
Scanner myKeyboard = new Scanner(System.in);

System.out.print("Enter a positive value: ");
value = myKeyboard.nextInt();

System.out.print("Enter an upper limit: ");
limit = myKeyboard.nextInt();

System.out.println("Multiples of "+value+" between "+ value + " & " + limit);

for (mult = value; mult <= limit; mult += value) {
   System.out.print(mult + "\t");
}
...
```

## Next Topic:

1. The `if` statement
2. The `if-else` statement
3. Relations Operators
4. Logical operators
5. Compound statements
6. Nested `if` statements
7. The `switch` statement
8. The conditional operator
9. The `while` loop
10. The `do-while` loop
11. **The `for` loop**
12. Nested loops
13. break, continue & exit