Middlesex University London

Lecture 5

Loops

REVISION TASK

- Write a Java program to let a first grader practice additions.
- The program randomly generates two single-digit integers number1 and number2 and displays a question such as "What is 7 + 9?" to the student.
- After the student types the answer, the program displays a message to indicate whether the answer is true or false

TODAY:

- While loop
- Do while loop
- For loop

CONTROL FLOW STATEMENTS

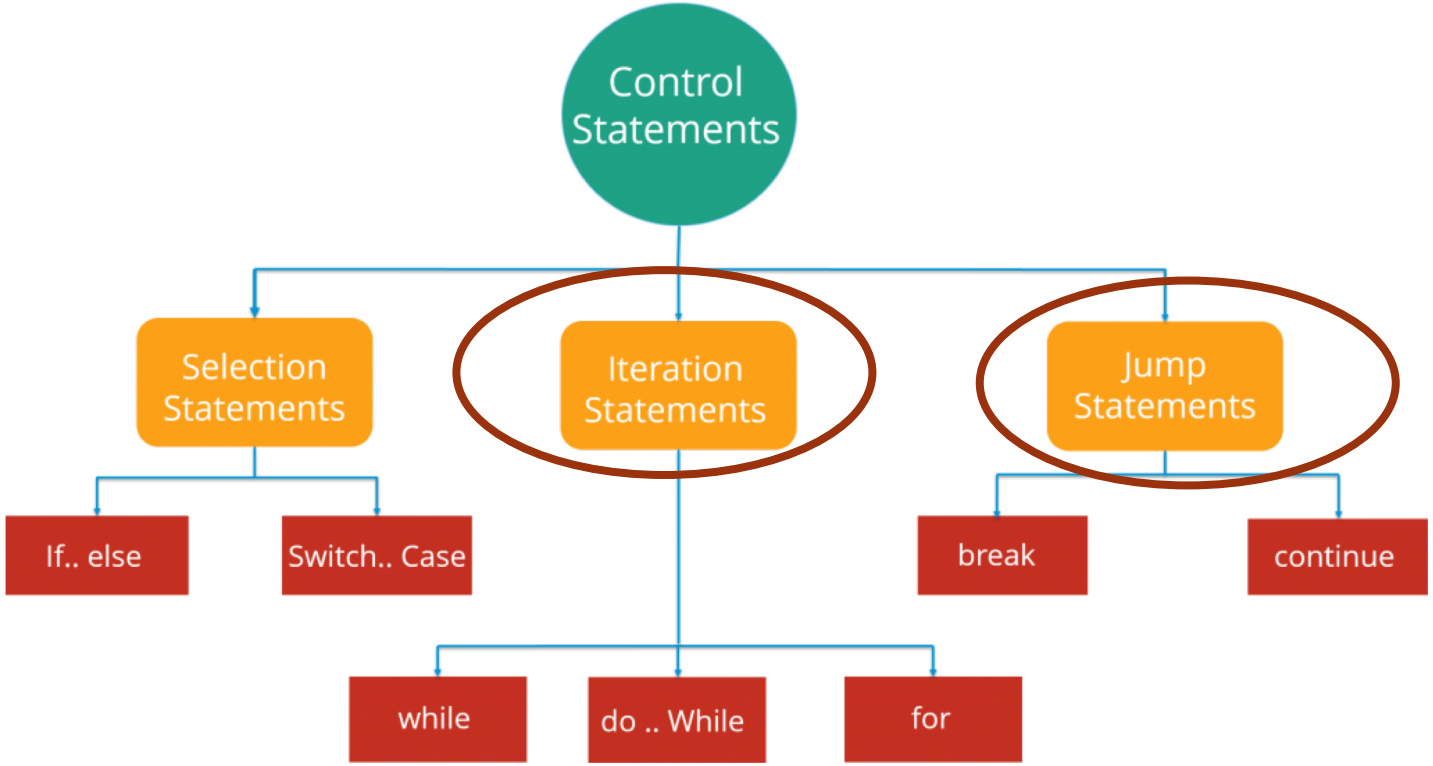


Image taken from:https://www.virginiamayhew.com/flow-chart-basics-programming/java-tutorial-java-programming-made-easy-learn-java-edureka-3/

WHAT ARE LOOPS?

- Loops are:
  - part of the family of Iteration statements
  - used for execution of same code number of times
- There are two main types of loops:
  - while loop
  - for loop

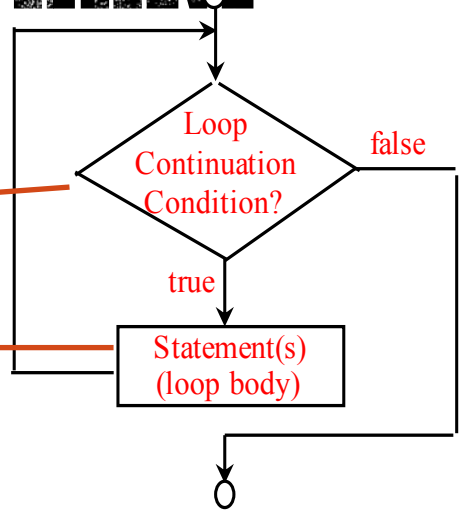
- Counter Loops VS Sentinel Loops

WHILE LOOP

- The while statement continually executes a block of statements while a particular condition is true.

WHILE LOOP SYNTAX AND FLOW CHART

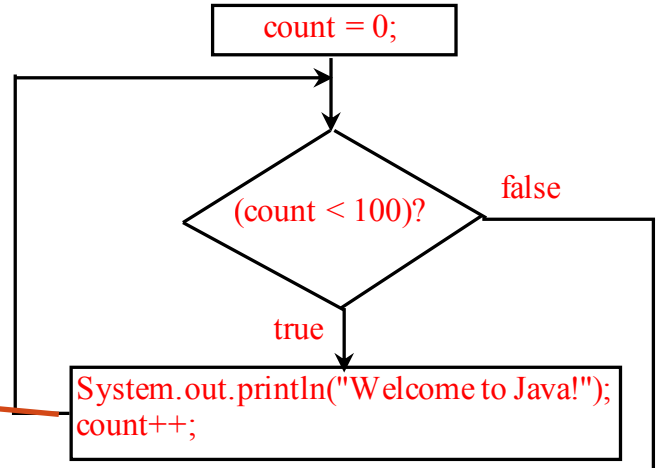
```
while (loop-continuation-condition) {
Statement(s);
}
```



(A)

```
int count = 0;

while (count < 100) {
    System.out.println("Welcome to Java!");
    count++;
}
```



(B)

TRACE WHILE LOOP

```
int count = 0;

while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

Initialize count

TRACE WHILE LOOP

```
int count = 0;

while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

(count < 2) is true

TRACE WHILE LOOP

```
int count = 0;

while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

TRACE WHILE LOOP

```
int count = 0;

while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

Increase count by 1
count is 1 now

TRACE WHILE LOOP

(count < 2) is still true since count is 1

```
int count = 0;

while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

TRACE WHILE LOOP

```
int count = 0;

while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

TRACE WHILE LOOP

```
int count = 0;

while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

Increase count by 1
count is 2 now

TRACE WHILE LOOP

(count < 2) is false since count is 2 now

```
int count = 0;

while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

TRACE WHILE LOOP

```
int count = 0;

while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

The loop exits. Execute the next statement after the loop.

Counter Loop

or Sentinel Loop?

DISCUSSION

- What will happen if you edit the previous program to (count >2)
  - A common programming error involves **infinite** loops.

TASK

- Ask the user to enter **n** the number of repetitions
- Edit your program to print "Welcome to Java" **n** times

TASK

WHAT IS BEING REPEATED?

n=100

- Edit your program to print the following
  1. Welcome to Java
  2. Welcome to Java
  3. Welcome to Java
  4. .....
  5. ....
  6. ...
  7. ...

  100. Welcome to Java

- Edit your program to print the following
  1. Welcome to Java
  3. Welcome to Java
  5. Welcome to Java

  ..

  99. Welcome to Java

WHILE LOOPS

```
ans = 'y';
Scanner s = new Scanner (System.in);
while (ans=='Y'  || ans=='y')
{
        System.out.println("Welcome to Java");
        System.out.println ("Do you want to repeat? (Y|N)");
        ans =s.next().charAt(0);
}
```

Sentinel Loop

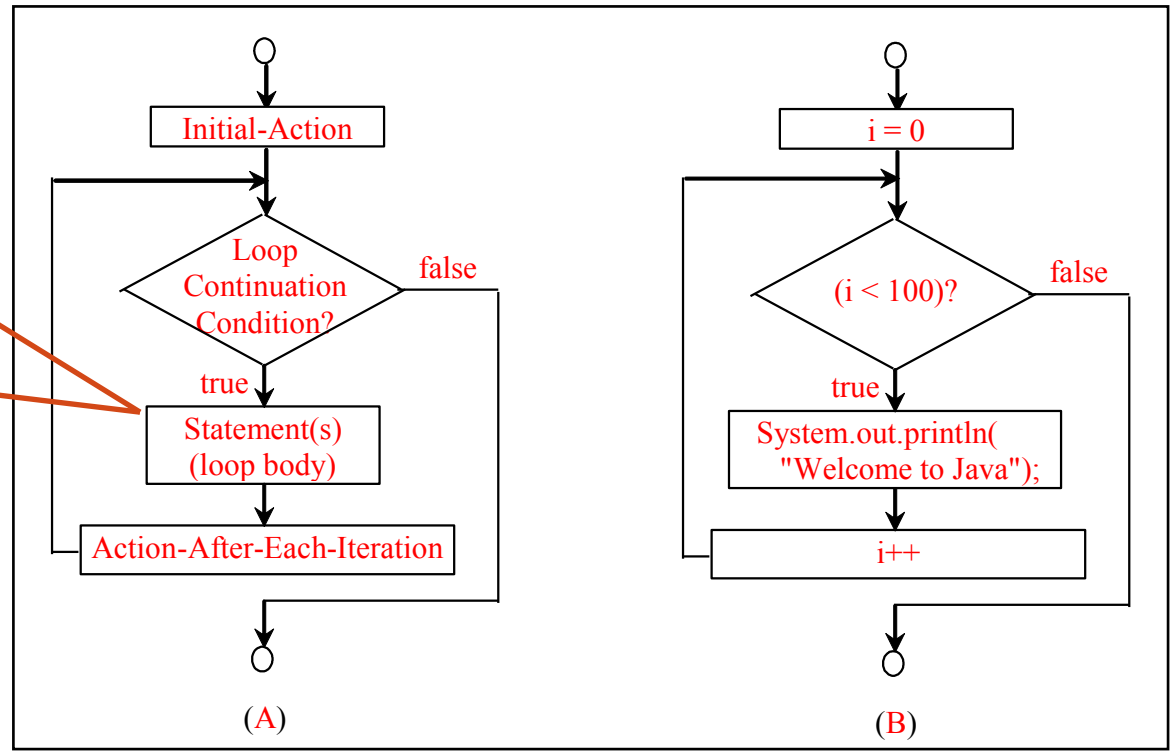
BACK TO REVISION TASK

- Write a Java program to let a first grader practice additions.
- The program randomly generates two single-digit integers number1 and number2 and displays a question such as "What is 7 + 9?" to the student.
- The program keeps on displaying a message to prompt the user for an answer until his answer is correct and will only display well done once the answer is correct

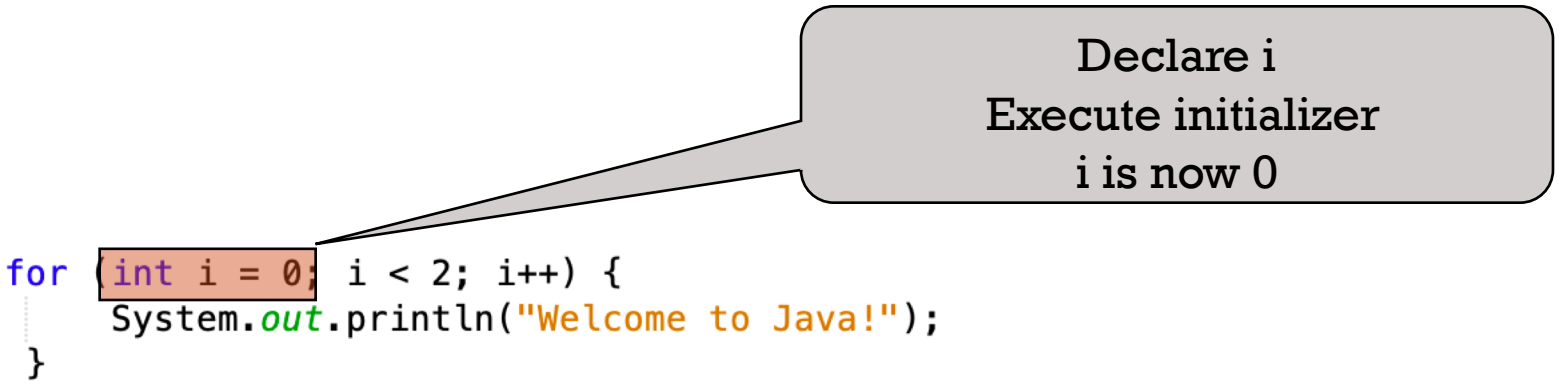
FOR LOOPS

```
for (initial-action; loop-continuation-condition; action-after-each-iteration) {
    Statement(s);
}
```

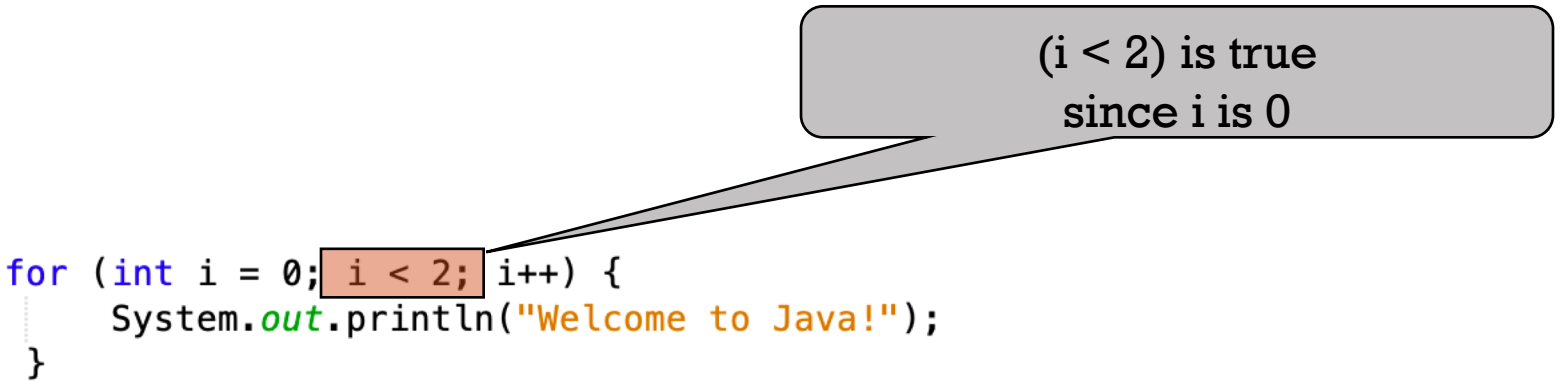
```
for (int i = 0; i < 100; i++) {
    System.out.println("Welcome to Java!");
}
```



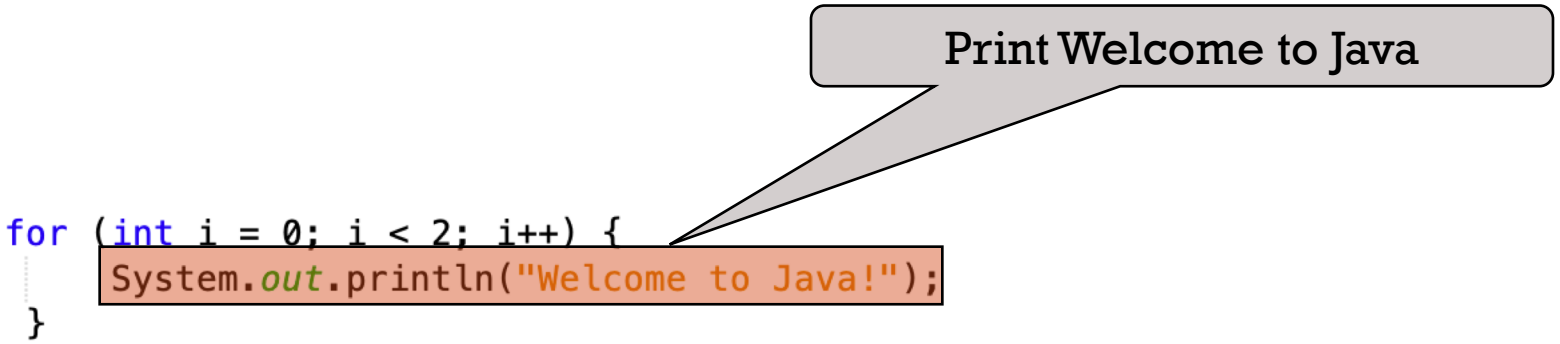
TRACE FOR LOOP



TRACE FOR LOOP



TRACE FOR LOOP

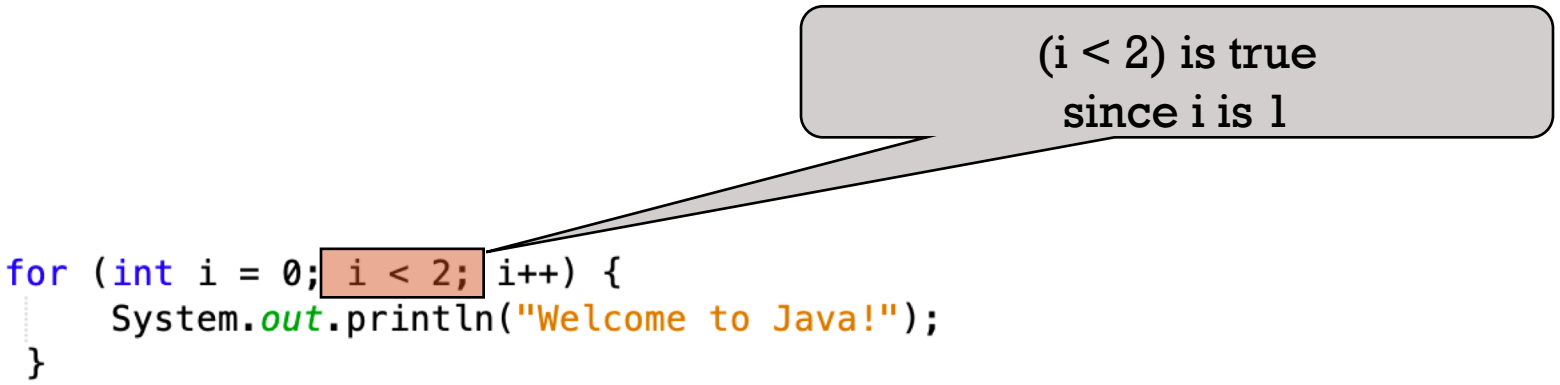


TRACE FOR LOOP

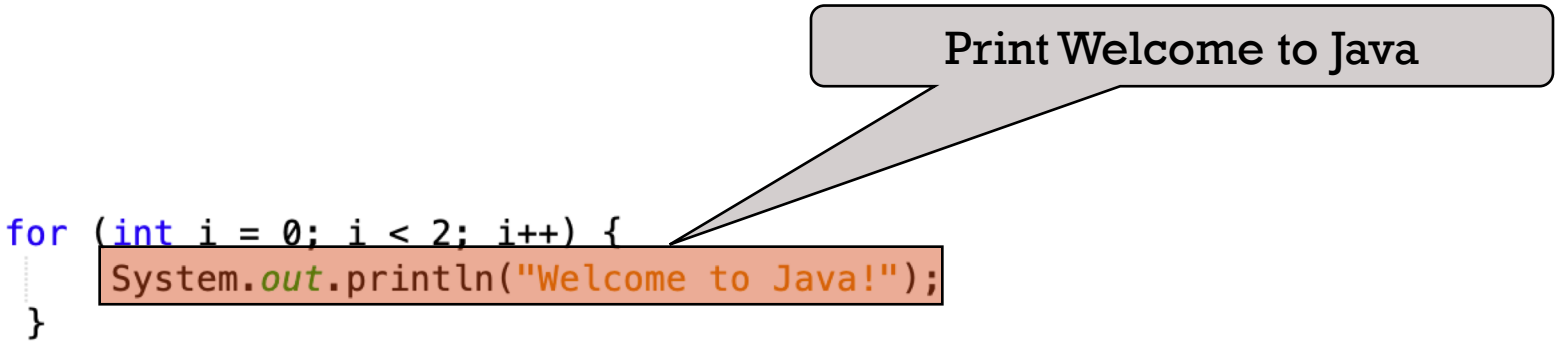
Execute adjustment statement
i now is 1

```
for (int i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
 }
```

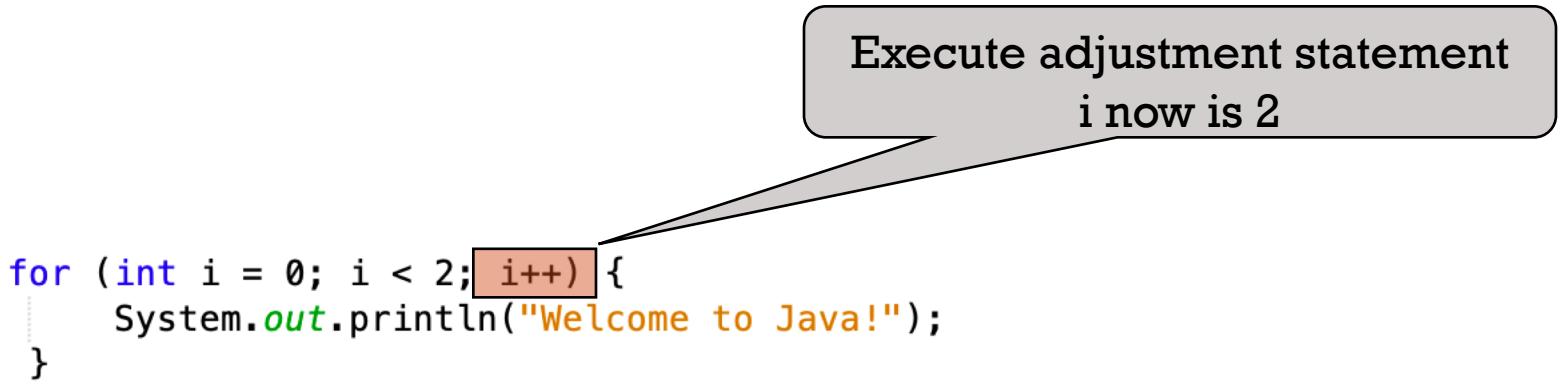
TRACE FOR LOOP



TRACE FOR LOOP



TRACE FOR LOOP



TRACE FOR LOOP

(i < 2) is false
since i is 2

```
for (int i = 0; i < 2; i++) {
     System.out.println("Welcome to Java!");
 }
```

TRACE FOR LOOP

```
for (int i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
 }
```

Exit the loop. Execute the next statement after the loop

FOR **LOOP**

- The for statement provides a compact way to iterate over a range of values.
- Programmers often refer to it as the "for loop" because of the way in which it repeatedly loops until a particular condition is satisfied.

TASK

SUMGIVINGNUMBERS

- Write a program that adds all the numbers given by user until -1 is entered

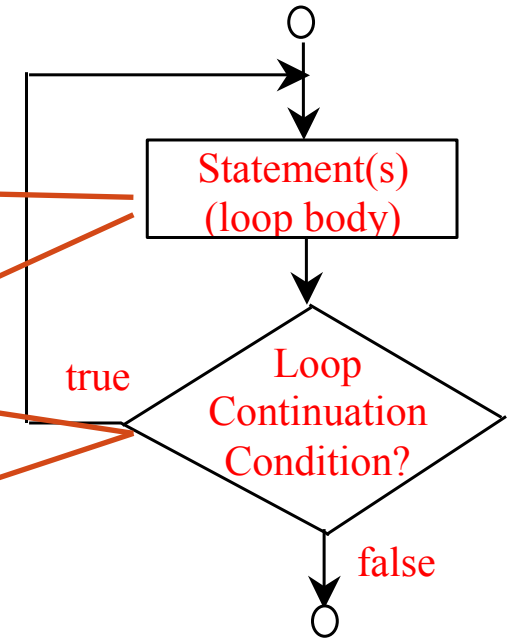
Example for input	Program Output
5 2 7 8 -1	Sum is 22
2 10 -1	Sum is 12
-1	Sum is 0
7 3 1 3 4 5 -1	Sum is 23

DO-WHILE LOOP

- The difference between do-while and while is that do-while evaluates its expression at the bottom of the loop instead of the top.
- Therefore, the statements within the do block are always executed at least once.

DO-WHILE LOOP

```
do{
        Statement(s);
}while(loop-continuation-condition);
```




```java
int i=0;

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

DISCUSSION

- What would be a benefit of using a do-while loop, rather than a while loop?

While Loop

```
ans = 'y';
Scanner s = new Scanner (System.in);
while (ans=='Y'  || ans=='y')
{
        System.out.println("Welcome to Java");
        System.out.println ("Do you want to repeat? (Y|N)");
       ans =s.next().charAt(0);
}
```

Do While Loop

```
Scanner s = new Scanner (System.in);
do
{
        System.out.println("Welcome to Java");
        System.out.println ("Do you want to repeat? (Y|N)");
       ans =s.next().charAt(0);
} while (ans=='Y'  || ans=='y');
```

NOTE

- If the loop-continuation-condition in a for loop is omitted, it is implicitly true. Thus the statement given below in (a), which is an infinite loop, is correct. Nevertheless, it is better to use the equivalent loop in (b) to avoid confusion:

```
for ( ; ; ) {
  // Do something
}
```

(a)

Equivalent

```
while (true) {
  // Do something
}
```

(b)

CAUTION

- Adding a semicolon at the end of the for 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);
}
```

- 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

Correct

```
int i=0;

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

WHICH LOOP TO USE?

- The three forms of loop statements, while, do-while, and for, are expressively equivalent; that is, you can write a loop in any of these three forms.
- For example, a while loop in (a) in the following figure can always be converted into the following for loop in (b):

```
while (loop-continuation-condition) {
  // Loop body
}
```

(a)

Equivalent

```
for ( ; loop-continuation-condition; )
  // Loop body
}
```

(b)

- A for loop in (a) in the following figure can generally be converted into the following while loop in (b) except in certain special cases.

```
for (initial-action;
     loop-continuation-condition;
     action-after-each-iteration) {
  // Loop body;
}
```

(a)

Equivalent

```
initial-action;
while (loop-continuation-condition) {
  // Loop body;
  action-after-each-iteration;
}
```

(b)

RECOMMENDATIONS

- Use the one that is most intuitive and comfortable for you.
- In general, 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.

USING BREAK AND CONTINUE

- Both "break" and "continue" are the **'jump' statements**, that transfer control of the program to another part of the program. The main difference between break and continue is that break is used for immediate termination of loop. On the other hand, 'continue' terminate the current iteration and resumes the control to the next iteration of the loop.

```
int sum =0;
int number =0;

while (number < 20){
    number++;
    sum += number;
    System.out.println("the sum vale is: "+sum);
    if (sum >= 10)
        break;
}

System.out.println("The number of loops is "+ number+ " times");
```

```
int sum =0;
int number =0;

while (number < 20){
    number++;


    if (number == 10 || number == 11)
        continue;
        System.out.println("number="+ number);
        sum += number;

}

System.out.println("the sum vale is: "+sum);
```

KEY THINGS YOU SHOULD UNDERSTAND AFTER WEEK 5

- Iteration statements
  - ***for*** loop
  - ***while*** loop
  - ***do while*** loop
- Jump statements
  - break
  - continue

MASTERING YOUR SKILLS

- Read Chapter 5 and do all the exercises in the end of this chapter
- Alternative source of learning about loops may be found in:
  - The *for* statement - tutorial
    http://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html
  - The *while*, and *do while* statements- tutorial
    http://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html

MORE EXERCISES

- Write a program that calculates the sum of numbers between 1 and n where **n** is entered by the user.
- Write a program that calculates the product of numbers between 1 and n where **n** is entered by the user.
- Write a program that determines the even numbers between 1 and n where **n** is entered by the user.
- Write a program that determines the odd numbers between 1 and n where **n** is entered by the user.
- Write a program that determines the prime numbers between 1 and n where **n** is entered by the user.

MORE EXERCISES

- Write a program that prints the following rectangular shape, where **n** is entered by the user.

```
n=5

***
***
***
***
***
```

```
n=2

***
***
```

```
n=7

***
***
***
***
***
***
***
```

MORE EXERCISES

- Write a program that prints the following XMAS Tree shape, where **n** is entered by the user.

```
n=4

   *
  ***
 *****
*******
```

```
n=3

  *
 ***
*****
```

```
n=7

      *
     ***
    *****
   *******
  *********
***********
```

MORE EXERCISES

- Write a program that reads and calculate the sum of an unspecified number of integers. The input 0 signifies the end of input.