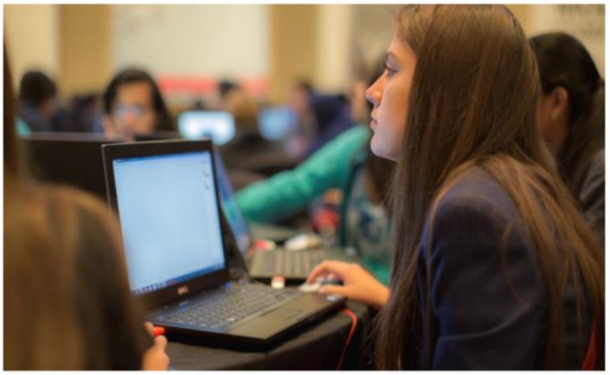
ORACLE

Academy

Java Foundations

6-2
while and do-while loops



ORACLE
Academy

Objectives

- This lesson covers the following objectives:
  - Use a while loop in a Java program (pre-test)
  - Use a do-while loop in a Java program (post-test)
  - Understand when one loop type may be more beneficial than another

How Many Times to Repeat?

- In some situations, you don't know how many times to repeat something
- That is, you may need to repeat some code until a particular condition occurs

How Many Times to Repeat?

- Let's look at an example:
  - Let's say you have to write a program to enter exam marks and find their average, but you may not know how many exams are involved
  - Instead of forcing users to count them all ahead of time, you can allow them to enter the marks one at a time and then enter -1 to indicate the completion of the entries

while Loop

- In such situations, you have to use the easier **while** loop
- It works like this:
  - The **while** loop continually executes a block of statements while a particular condition is true

while Loop Syntax

- The while statement evaluates boolean expression
- The statement(s) within the curly braces execute as long as boolean expression is true

```
while (<boolean expression>) {
    <statement(s)> ;
}//end while
```

Pre-Test Loop

- A pre-test loop evaluates the condition before the loop executes
- If the condition is false, the loop stops or may never execute
- for and while loops are pre-test loops

Countdown Scenario

- Let's write the Countdown scenario discussed in the previous lesson by using the while loop:

What we know	Technical Name	Code
When the loop starts …	Initialization Expression	`int i = 10;`
Continue looping if …	Condition Expression	`i >= 0;`
After each loop …	Update Expression	`i--;`
Code to repeat	Code Statements	`System.out.println(i);`

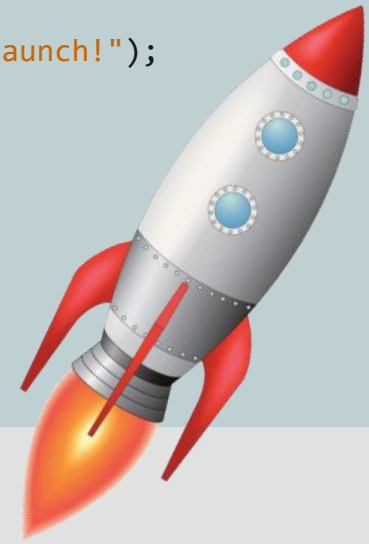
Countdown Scenario: while Loop

```
public class CountDownWhile {

    public static void main(String[] args) {
        int i = 10;
        System.out.println("Countdown to Launch!");

        while (i >= 0) {
            System.out.println(i);
            i--;
        }//end while

        System.out.println("Blast Off!");
    }//end method main
}//end class CountDownWhile
```



Some while Loops Never Run

- It's possible that the loop body will never run if the conditions are such that the boolean expression was already false, for example:

```
public class WhileLoopExample {
   public static void main(String args[]) {
      int num = 0;
      System.out.println("Let's count to 10!");
         while (num > 10) {
            num = num + 1;
            System.out.println("Number: " + num);
         }//end while
      System.out.println("We have counted to 10! Hurrah!");
   }//end method main
}//end class WhileLoopExample
```


In the slide example, the initial value of num is 0 and the boolean expression is num > 10, instead of num < 10.It's already false from the start, because 0 will never be greater than 10. The while loop evaluates the boolean expression, num > 10,finds that it's false, and prints:

Let's count to 10!
We have counted to 10! Hurrah!

Getting Stuck in an Infinite Loop

- You'll get stuck in a while loop if you write a boolean condition that will never evaluate to false
- We call this an **infinite loop** because it never stops executing
- If this happens, your loop will execute forever or until you send an interrupt command
- You should avoid writing infinite loops and always verify the boolean expression to ensure that the loops terminate normally

Let's Return to the Countdown Scenario

- What if we had accidentally written **`i++`** instead of **`i--`** within the while loop?

```
int i = 10;
System.out.println("Countdown to Launch!");
while (i >= 0) {
   System.out.println(i);
   i++;
}//end while
System.out.println("Blast Off!");
```

- It would continue adding 1 to **`i`**, keeping its value more than 10 forever
- This is an infinite loop because the boolean condition always remains true, and this program continues to execute

Using while Loop and Scanner Class

- while loops are often used with input by using the Scanner class

```
public static void main(String[] args) {
   Scanner console = new Scanner(System.in);
   int sum = 0;

   System.out.println("Enter a number (-1 to quit): ");
   int num = console.nextInt();
   while (num != -1) {
       sum = sum + num;
       System.out.println("Enter a number (-1 to quit): ");
       num = console.nextInt();
   }//end while

   System.out.println("The sum is " + sum);
}//end method main
```


See next slide for results from this code.

Using while Loop and Scanner Class

- Example:
  - A program that prompts users for numbers until they type -1, and then outputs their sum

```
public static void main(String[] args) {
   Scanner console = new Scanner(System.in);
   int sum = 0;

   System.out.println("Enter a number (-1 to quit): ");
   int num = console.nextInt();
   while (num != -1) {
       sum = sum + num;
       System.out.println("Enter a number (-1 to quit): ");
       num = console.nextInt();
   }//end while

   System.out.println("The sum is " + sum);
}//end method main
```


The slide example produces the following output:

```
Enter a number (-1 to quit):
20
Enter a number (-1 to quit):
40
Enter a number (-1 to quit):
-1
The sum is 60
```

Exercise 1

- Import and open the `WhileLoopEx` project
- Modify `SquareRootWhile.java` to use a while loop to repeatedly prompt users to type a number until they type a non-negative number, and then computes the square root

- Expected output:

```
Type a non-negative integer: -5
Invalid number, try again: -1
Invalid number, try again: 11
The square root of 11 is 3.166
```

Post-Test Loop

- A post-test loop evaluates its condition at the bottom of the loop instead of the top
- The do-while loop is a post-test loop

do-while Loop

- The do-while loop is a modified while loop that allows you to execute the loop once, before testing the boolean condition
- Syntax:

```
do{
 <statement(s)>
}while(<condition>);
```

The do-while loop requires a semicolon after the condition at the end of the loop

If the condition is false, the loop still executes at least once, but it stops at the end of the loop. Therefore, the statements within the do block are always executed at least once.

Countdown Scenario: do-while Loop

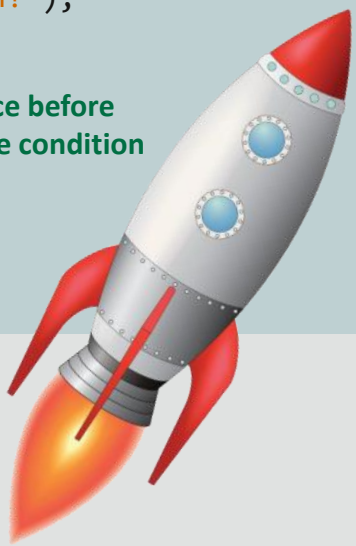
```
public static void main(String[] args) {

    int i = 10;
    System.out.println("Countdown to Launch!");

    do {
        System.out.println(i);
        i--;
    }while (i >= 0);

    System.out.println("Blast Off!");
}//end method main
```

Executed once before evaluating the condition



ORACLE Academy

JFo 6-2
while and do-while loops

Output:
Countdown to Launch!
10
9
8
7
6
5
4
3
2
1
0
Blast Off!

Exercise 2

- Import and open the `WhileLoopEx` project
- Examine the `SumofNums.java`, which sums up a sequence of 10 integers that are input by the user
- Can you implement the same by using a do-while loop?

Standard for Loop Compared with while Loop

- Differences between these two loops:
- In a for loop:
  - Initialization, condition, and increment statements are all put together in one line, which makes the loop easier to understand and implement

Standard for Loop Compared with while Loop

- Differences between these two loops:
- In a while loop:
  - Initialization is done before the beginning of the loop
  - The conditional statement is always put at the start of the loop
  - Increment statements can be either combined with condition or embedded into the body of the loop

In the next three slides, you see a while loop example at the top of the slide. At the bottom, you see the same logic implemented by using a standard for loop.

The three essential elements of a while loop are also present in the for loop, but in different places.

1. The counter (i) is declared and initialized outside the while loop on line 1.
2. The counter is incremented in the while loop on line 4.
3. The boolean expression that determines the number of loop iterations is within the parentheses for the while loop on line 2.

In the for loop, all three elements occur within the parentheses, as indicated in the slide. The output for each statement is the same.

Comparing the Initialization Counter

Loop `while`

```
int i = 10;
while (i >= 0) {
    System.out.println(i);
    i--;
}//end while
System.out.println("Blast Off!");
```

Initialize counter

Loop `for`

```
for (int i = 10; i >= 0; i--) {
    System.out.println(i);
}//end for
System.out.println("Blast Off!");
```

Comparing the Boolean Expression

Loop while

```
int i = 10;
while (i >= 0) {
    System.out.println(i);
    i--;
}//end while
System.out.println("Blast Off!");
```

Loop for

```
for (int i = 10; i >= 0; i--) {
    System.out.println(i);
}//end for
System.out.println("Blast Off!");
```

Comparing the Increment Counter

Loop `while`

```
int i = 10;
while (i >= 0) {
    System.out.println(i);
    i--;
}//end while
System.out.println("Blast Off!");
```

Increment counter

Loop `for`

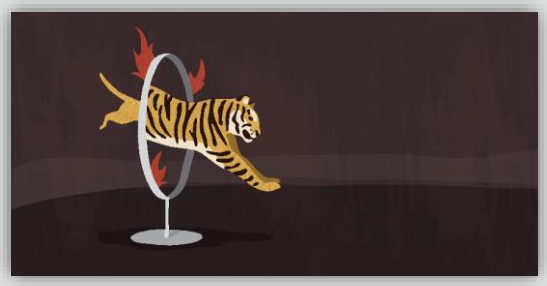
```
for (int i = 10; i >= 0; i--) {
    System.out.println(i);
}//end for
System.out.println("Blast Off!");
```

Which Loop Do I Use?

Loop Type	Definition	When to Use
while	Pre-test loop that repeats until a specified condition is false	Use when you are not certain the number of times the loop should be executed or even if it should at all
do-while	Post-test loop that executes the loop before testing the condition, then repeats until the condition is false	Use when you know that the code must be executed at least once and possibly more times depending on the condition
for	Loop that contains an initialized counter, and increments the counter with each run through the loop. Repeats until the condition is false	Use when you need to execute a loop a specific number of times, or when you need to increment through a set of data. The counter can also be used as an index for accessing data one item at a time

Summary

- In this lesson, you should have learned how to:
  - Use a while loop in a Java program (pre-test)
  - Use a do-while loop in a Java program (post-test)
  - Understand when one loop type may be more beneficial than another




ORACLE
Academy