



# Java Foundations

6-2

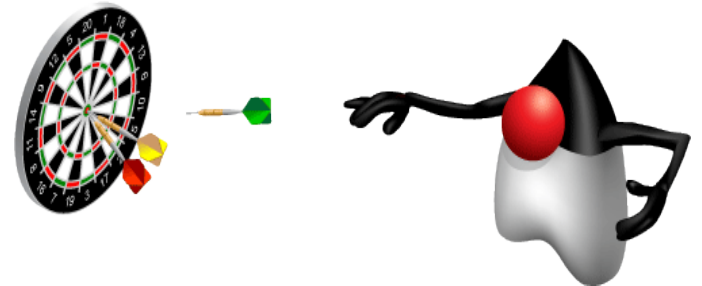
`while` and `do-while` loops



# 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



# Topics

- Using while Loops
- Using do-while Loops
- Which Loop Do I Use?



for loop

while and  
do-while  
Loops

break and  
continue  
Statements

Section 6

# 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

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

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



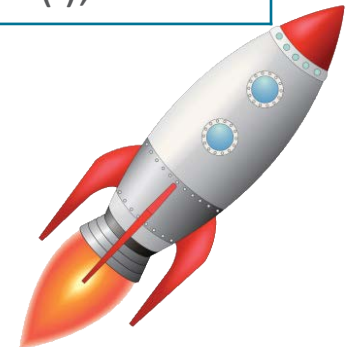
# Pre-Test Loop

- 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	<code>int i = 10;</code>
Continue looping if ...	Condition Expression	<code>i &gt;= 0;</code>
After each loop ...	Update Expression	<code>i--;</code>
Code to repeat	Code Statements	<code>System.out.println(i);</code>



# 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--;  
        }  
  
        System.out.println("Blast Off!");  
    }  
}
```

# 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);  
        }  
        System.out.println("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?

```
public static void main(String[] args) {  
    int i = 10;  
    System.out.println("Countdown to Launch!");  
    while (i >= 0) {  
        System.out.println(i);  
        i++;  
    }  
    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();  
    }  
    System.out.println("The sum is " + sum);  
}
```

# 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();  
    }  
    System.out.println("The sum is " + sum);  
}
```





# 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
```

# Topics

- Using while Loops
- Using do-while Loops
- Which Loop Do I Use?

for Loop

while and  
do-while  
Loops

break and  
continue  
Statements



Section 6

# 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.

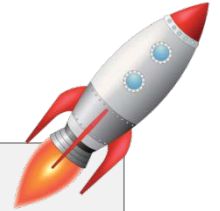
# do-while Loop

- 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.

# 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!");  
}
```

Executed once before evaluating the condition



## 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?

# Topics

- Using while Loops
- Using do/while Loops
- Which Loop Do I Use?

for Loop

while and  
do-while  
Loops

break and  
continue  
Statements



Section 6

# 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.

# Comparing the Initialization Counter

while loop

```
1. int i = 10;  
2. while (i >= 0) {  
3.     System.out.println(i);  
4.     i--;  
5. }  
6. System.out.println("Blast Off!");
```

Initialize  
counter

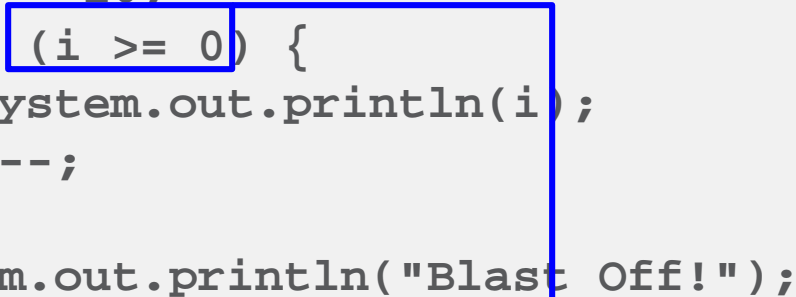
for loop

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

# Comparing the Boolean Expression

while loop

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

A blue box highlights the boolean expression `(i >= 0)` in the while loop. A blue arrow points from this box to the `i >= 0` part of the for loop's condition. Another blue arrow points from the `i >= 0` part of the for loop's condition to the text "boolean expression".

for loop


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

boolean  
expression

# Comparing the Increment Counter

while loop


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



Increment  
counter

for loop

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



# Which Loop Do I Use?

Loop Type	Definition	When to Use
<b>while</b>	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.
<b>do-while</b>	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.
<b>for</b>	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

