



# Java Foundations

6-2

Loops while e do/while

**ORACLE**  
Academy



Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

# Objetivos

- Esta lição abrange os seguintes objetivos:
  - Usar um loop while em um programa Java (pré-teste)
  - Usar um loop do-while em um programa Java (pós-teste)
  - Entender quando um tipo de loop pode ser mais benéfico do que outro



# Tópicos

- Usando Loops while
- Usando Loops do-while
- Que Loop Eu Uso?



**ORACLE**  
Academy

JFo 6-2  
while e do/while

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

4

# Quantas Vezes É Preciso Repetir o Código?

- Em algumas situações, você não sabe quantas vezes é necessário repetir algo
- Ou seja, pode ser que você precise repetir um código até que determinada condição ocorra

# Quantas Vezes É Preciso Repetir o Código?

- Vamos analisar um exemplo:
  - Suponha que você precise escrever um programa para inserir notas de exame e calcular a respectiva média, mas não saiba quantos exames estão envolvidos
  - Em vez de forçar os usuários a contarem todos os exames antes da hora, você pode permitir que eles insiram as notas uma de cada vez e, em seguida, inserir -1 para indicar a conclusão das entradas

# Loop while

- Nessas situações, você precisa usar o loop while mais fácil
- É assim que ele funciona:
  - o **loop while** executa continuamente um bloco de instruções, enquanto uma condição específica é verdadeira

## Sintaxe do Loop while

- A instrução while avalia a expressão booliana
- A(s) instrução(ções) entre chaves é(são) executada(s) enquanto a expressão booliana é verdadeira

```
while (<expressão booliana>) {  
    <instrução(ções)> ;  
} //fim while
```



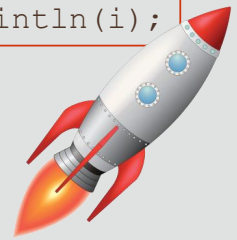
# Loop Pré-teste

- Um loop pré-teste avalia uma condição antes de o loop ser executado
- Se a condição for falsa, o loop parará ou nunca será executado
- Os loops for e while são loops pré-teste

# Cenário de Contagem Regressiva

- Vamos escrever o cenário de Contagem Regressiva discutido na lição anterior usando o loop while:

O que sabemos	Nome Técnico	Código
O que o loop inicia...	Expressão Initialization	<code>int i = 10;</code>
Continuar fazendo loop se...	Expressão Condition	<code>i &gt;= 0;</code>
Depois de cada loop...	Expressão Update	<code>i--;</code>
Código para repetição	Instruções de Código	<code>System.out.println(i);</code>





# Cenário de Contagem Regressiva: Loop while

```
public class CountdownWhile {  
  
    public static void main(String[] args) {  
        int i = 10;  
        System.out.println("Contagem Regressiva para "  
                           + "Lançamento!");  
  
        while (i >= 0) {  
            System.out.println(i);  
            i--;  
        } //fim while  
  
        System.out.println("Decolar!");  
    } //fim do método main  
} //fim da classe CountdownWhile
```



## Alguns Loops while Nunca São Executados

- É possível que o corpo do loop nunca seja executado
- Se as condições forem tais que a expressão booliana já era falsa, Por exemplo:

```
public class WhileLoopExample {  
    public static void main(String args[]) {  
        int num = 0;  
        System.out.println("Vamos contar até 10!");  
        while (num > 10) {  
            num = num + 1;  
            System.out.println("Número: " + num);  
        }  
        System.out.println("Contamos até 10! Viva!");  
    }  
}
```

**ORACLE**  
Academy

JFo 6-2  
while e do/while

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

12

No exemplo do slide, o valor inicial de `num` é 0 e a expressão booliana é `num > 10`, em vez de `num < 10`. Ela já é falsa desde o início porque 0 nunca será maior que 10. O loop `while` avalia a expressão booliana, `num > 10`, percebe que ela é falsa e imprime o seguinte:

```
Vamos contar até 10!  
Contamos até 10! Viva!
```

## Ficando Preso em um Loop Infinito

- Você ficará preso em um loop while se escrever uma condição booliana que nunca será avaliada como falsa
- Esse tipo de loop denomina-se **loop infinito** porque ele nunca para de ser executado
- Se isso acontecer, seu loop será executado para sempre ou até você enviar um comando de interrupção
- Evite escrever loops infinitos e sempre verifique a expressão booliana para assegurar que os loops sejam terminados normalmente

## Vamos Retornar ao Cenário de Contagem Regressiva

- E se você escrevesse acidentalmente `i++` , em vez de `i--` dentro do loop `while`?

```
int i = 10;
System.out.println("Contagem Regressiva para Lançamento!");
while (i >= 0) {
    System.out.println(i);
    i++;
} //fim while
System.out.println("Decolar!");
```

- Ele continuaria a somar 1 a `i`, mantendo seu valor superior a 10 indefinidamente
- Esse é um loop infinito porque a condição booliana sempre permanece verdadeira, e esse programa continua a ser executado

# Usando o Loop while e a Classe Scanner

- Os loops while geralmente são usados com uma entrada utilizando a classe Scanner

```
public static void main(String[] args) {  
    Scanner console = new Scanner(System.in);  
    int sum = 0;  
  
    System.out.println("Informe um número (-1 para encerrar): ");  
    int num = console.nextInt();  
    while (num != -1) {  
        sum = sum + num;  
        System.out.println("Informe um número (-1 para encerrar): ");  
        num = console.nextInt();  
    } //fim while  
  
    System.out.println("A soma é " + sum);  
} //fim do método main
```

**ORACLE**  
Academy

JFo 6-2  
while e do/while

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

15

O exemplo do slide produz a saída a seguir:

```
Informe um número (-1 para encerrar):  
20  
Informe um número (-1 para encerrar):  
40  
Informe um número (-1 para encerrar):  
-1  
A soma é 60
```

# Usando o Loop while e a Classe Scanner

- Exemplo:

- um programa que solicita aos usuários números até eles digitarem -1 e, então, exibe a soma

```
public static void main(String[] args) {  
    Scanner console = new Scanner(System.in);  
    int sum = 0;  
  
    System.out.println("Informe um número (-1 para encerrar): ");  
    int num = console.nextInt();  
    while (num != -1) {  
        sum = sum + num;  
        System.out.println("Informe um número (-1 para encerrar): ");  
        num = console.nextInt();  
    } // fim while  
  
    System.out.println("A soma é " + sum);  
} // fim do método main
```

**ORACLE**  
Academy

JFo 6-2  
while e do/while

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

16

O exemplo do slide produz a saída a seguir:

```
Informe um número (-1 para encerrar):  
20  
Informe um número (-1 para encerrar):  
40  
Informe um número (-1 para encerrar):  
-1  
A soma é 60
```





## Exercício 1

- Importe e abra o projeto `WhileLoopEx`
- Modifique `SquareRootWhile.java` para usar um loop `while` que solicita repetidamente aos usuários para digitarem um número até eles digitarem um número não negativo e, em seguida, calcula a raiz quadrada
- Saída esperada:

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

# Tópicos

- Usando Loops while
- **Usando Loops do-while**
- Que Loop Eu Uso?



**ORACLE**  
Academy

JFo 6-2  
while e do/while

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

18

# Loop Pós-teste

- Um loop pós-teste avalia sua condição no fim do loop, e não no início
- O loop do-while é um loop pós-teste

## Loop do-while

- O loop do-while é um loop while modificado que permite a você executar o loop uma vez antes de testar a condição booliana
- Sintaxe:

```
do{  
  <instrução(ções)>  
}while(<condição>);
```

O loop do-while requer um ponto e vírgula depois da condição no fim do loop

Se a condição for falsa, o loop continuará sendo executado pelo menos uma vez, mas parará no fim. Portanto, as instruções dentro do bloco do sempre são executadas pelo menos uma vez.



# Cenário de Contagem Regressiva: Loop do-while

```
public static void main(String[] args) {  
  
    int i = 10;  
    System.out.println("Contagem Regressiva para "  
                        + "Lançamento!");  
  
    do {  
        System.out.println(i);  
        i--;  
    }while (i >= 0);  
  
    System.out.println("Decolar!");  
} //fim do método main
```

Executado uma  
vez antes de a  
condição ser  
avaliada



Saída:

Contagem Regressiva para o Lançamento!

10

9

8

7

6

5

4

3

2

1

0

Decolar!



## Exercício 2

- Importe e abra o projeto `WhileLoopEx`
- Examine `SumofNums.java`, que soma uma sequência de 10 números inteiros inseridos pelo usuário
- Você pode implementar o mesmo usando um loop `do-while`?

# Tópicos

- Usando Loops while
- Usando Loops do-while
- **Que Loop Eu Uso?**



**ORACLE**  
Academy

JFo 6-2  
while e do/while

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

23

## Padrão do Loop for Comparado com o Loop while

- Diferenças entre esses dois loops:
- Em um loop for:
  - As instruções de inicialização, condição e incremento (initialization, condition e increment, respectivamente) são colocadas todas juntas em uma linha, o que facilita a compreensão e implementação do loop

Nos próximos três slides, você vê o exemplo de um loop `while` na parte superior do slide. Na parte inferior, você vê a mesma lógica implementada usando um loop `for` padrão.

Os três elementos essenciais de um loop `while` também estão presentes no loop `for`, mas em diferentes locais.

1. O contador (`i`) é declarado e inicializado fora do loop `while` na linha 1.
2. O contador é incrementado no loop `while` na linha 4.
3. A expressão booleana que determina o número de iterações do loop está dentro dos parênteses do loop `while` na linha 2.

No loop `for`, todos os três elementos ocorrem dentro dos parênteses, conforme indicado no slide.

A saída de cada instrução é a mesma.



## Padrão do Loop for Comparado com o Loop while

- Diferenças entre esses dois loops:
- Em um loop while:
  - A inicialização é feita antes do início do loop
  - A instrução condicional sempre é colocada no início do loop
  - As instruções de incremento podem ser combinadas com as de condição ou incorporadas ao corpo do loop

Nos próximos três slides, você vê o exemplo de um loop `while` na parte superior do slide. Na parte inferior, você vê a mesma lógica implementada usando um loop `for` padrão.

Os três elementos essenciais de um loop `while` também estão presentes no loop `for`, mas em diferentes locais.

1. O contador (`i`) é declarado e inicializado fora do loop `while` na linha 1.
2. O contador é incrementado no loop `while` na linha 4.
3. A expressão booliana que determina o número de iterações do loop está dentro dos parênteses do loop `while` na linha 2.

No loop `for`, todos os três elementos ocorrem dentro dos parênteses, conforme indicado no slide.

A saída de cada instrução é a mesma.

# Comparando o Contador de Inicialização

## Loop while

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

Inicializar o  
contador

## Loop for

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

# Comparando a Expressão Booliana

## Loop while

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

→ Expressão  
booliana

## Loop for

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

# Comparando o Contador de Incremento

## Loop while

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

Contador de  
incremento

## Loop for

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

# Que Loop Eu Uso?

Tipo de Loop	Definição	Quando Usar
<b>while</b>	Loop pré-teste que é repetido até uma condição especificada ser falsa	Utilize-o quando não tiver certeza do número de vezes que o loop deverá ser executado ou se ele deverá ser executado
<b>do-while</b>	Loop pós-teste que executa o loop antes de testar a condição e é repetido até a condição ser falsa	Utilize-o quando você souber que o código deve ser executado pelo menos uma vez e possivelmente mais vezes, dependendo da condição
<b>for</b>	Loop que contém um contador inicializado e incrementa esse contador a cada execução pelo loop. É repetido até a condição ser falsa	Utilize-o quando precisar executar um loop um número específico de vezes ou quando precisar aumentar em um conjunto de dados.  O contador também pode ser usado como índice para acessar dados um item por vez

# Resumo

- Nesta lição, você deverá ter aprendido a:
  - Usar um loop while em um programa Java (pré-teste)
  - Usar um loop do-while em um programa Java (pós-teste)
  - Entender quando um tipo de loop pode ser mais benéfico do que outro



