

The logo for Oracle Academy is centered on a light gray background. It features the word "ORACLE" in a bold, orange, sans-serif font. Below it, the word "Academy" is written in a smaller, dark gray, sans-serif font. The entire logo is framed by a thin black border, with dark gray horizontal bars at the top and bottom.

ORACLE

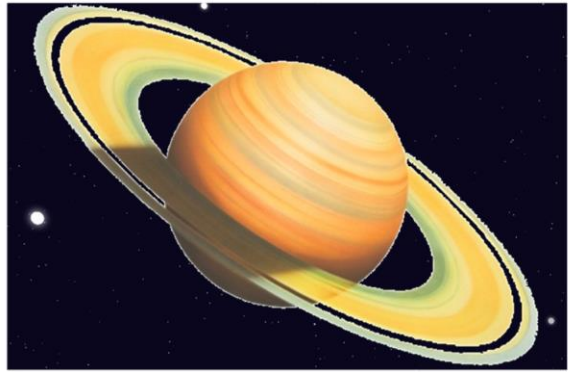
Academy

Java Foundations

6-1

Loops for

ORACLE
Academy



Copyright © 2022, Oracle and/or its affiliates. Oracle, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

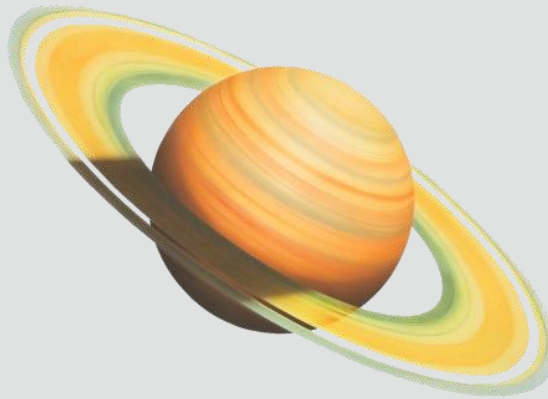
Objetivos:

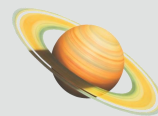
- Esta lição abrange os seguintes objetivos:
 - Entender os componentes do loop for padrão
 - Entender como criar e usar um loop for
 - Entender o escopo de uma variável
 - Entender técnicas de depuração
 - Explicar como loops infinitos ocorrem em Java



Missão para os Anéis de Saturno

- Vamos lançar um foguete
- Sua missão é estudar os anéis de Saturno
- Você faz ideia de como programar um cronômetro para contagem regressiva?





A Contagem Regressiva

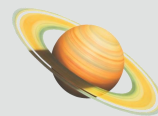
- Uma contagem regressiva a partir de 10 requer 10 linhas de código

```
System.out.println("Contagem Regressiva para Lançamento: ");  
System.out.println(10);  
System.out.println(9);  
System.out.println(8);  
System.out.println(7);  
System.out.println(6);  
System.out.println(5);  
System.out.println(4);  
System.out.println(3);  
System.out.println(2);  
System.out.println(1);  
System.out.println("Decolar!");
```



A Contagem Regressiva

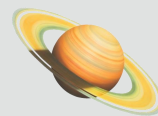
- Uma contagem regressiva a partir de 100 requer 100 linhas de código
- Isso seria muito cansativo para se programar
- Existe uma maneira mais prática de escrever este programa?
- O código pode acomodar facilmente um valor inicial?



A Contagem Regressiva

```
System.out.println("Contagem Regressiva para Lançamento: ");  
System.out.println(100);  
System.out.println(99);  
System.out.println(98);  
System.out.println(97);  
System.out.println(96);  
System.out.println(95);  
...  
...  
...  
...  
...  
...  
System.out.println(2);  
System.out.println(1);  
System.out.println("Decolar!");
```



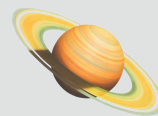


As Variáveis Podem Ajudar?

- As variáveis são um tanto quanto úteis
- Mas, ainda assim, precisamos copiar e colar as mesmas linhas de código até 0 ser impresso

```
System.out.println("Contagem Regressiva para Lançamento: ");  
  
int i = 10;  
System.out.println(i);  
i--;  
System.out.println(i);  
i--;  
System.out.println(i);  
i--;  
...  
System.out.println("Decolar!");
```






Repetindo um Código

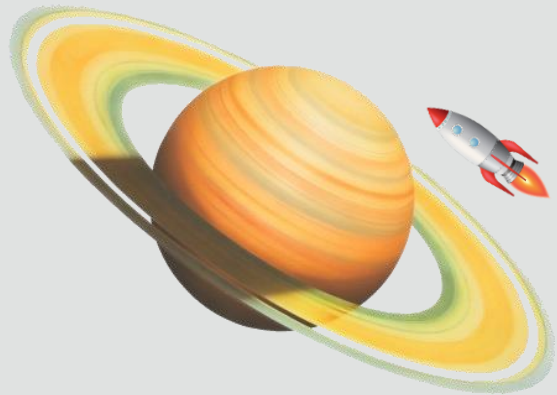
- Podemos fazer com que as mesmas linhas de código repitam uma variável um número de vezes?
- As linhas de 7 a 10 mostram o bloco de código que desejamos repetir
- Lembre-se da natureza linha por linha dos programas:
 - Quando o programa atinge a linha 10...
 - Queremos retornar à linha 7

```
5  int i = 10;  
6  
7  {  
8      System.out.println(i);  
9      i--;  
10 }
```



Instruções de Loop

- As instruções de loop são usadas para repetir linhas de código
- O Java fornece três tipos de loops:
 - `for`
 - `while`
 - `do-while`



Repetindo um Comportamento



```
while (!areWeThereYet) {  
    read book;  
    argue with sibling;  
    ask, "Are we there yet?";  
}  
  
Woohoo!;  
Get out of car;
```

Um requisito comum em um programa é repetir algumas instruções. Em geral, o código continua a repetir as instruções até algo mudar. Então, o código quebra o loop e continua com a instrução seguinte.

O exemplo do pseudocódigo mostra um loop while que permanece em loop até o booleano `areWeThereYet` ser verdadeiro.

Loops

- Os loops são usados em programas para uma execução repetida de uma ou mais instruções até uma condição de término ser atingida
 - Até uma expressão ser falsa
 - ou
 - Por um número específico de vezes:
 - Quero imprimir os números de 1 a 10
 - Quero calcular a soma dos números em determinado intervalo
- Um loop for é executado determinado número de vezes
 - Os loops for denominam-se loops definidos

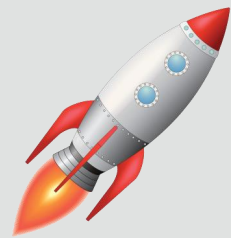
Em programação, às vezes você quer trabalhar com várias entradas, mas deseja executar a mesma lógica para cada item de entrada.

Os loops começam no início de um trecho de código, executam a lógica e retornam para o início do loop com a nova entrada, prontos para executarem o código novamente.

O que Sabemos

- No cenário da Contagem Regressiva, é isso que sabemos:

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 >= 0;</code>
Depois de cada loop...	Expressão Update	<code>i--;</code>
Código para repetição	Instruções de Código	<code>System.out.println(i);</code>



Visão Geral do Loop for

- Sintaxe:

Cabeçalho

```
for(initialization; condition; update){  
    Instrução(ões) de código  
    Instrução(ões) de Código }  
} //fim for
```

Corpo

- A expressão initialization inicializa o loop, ela é executada uma única vez quando o loop começa
- Quando a expressão condition é avaliada como falsa, o loop termina
- A expressão update é chamada depois de cada iteração por meio do loop, essa expressão pode aumentar ou diminuir um valor
- Cada expressão deve ser separada por um ponto e vírgula (;)

Expressão Initialization

- Executada uma vez quando o loop começa
- Informa ao compilador qual variável (denominada contador de loops) é usada no loop
- Pode começar em qualquer valor, e não apenas em 10

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

Expressão Condition

- O loop continua até a expressão booliana ser verdadeira
- Usa operadores de comparação:
 - (==, !=, <, >, <=, >=)

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

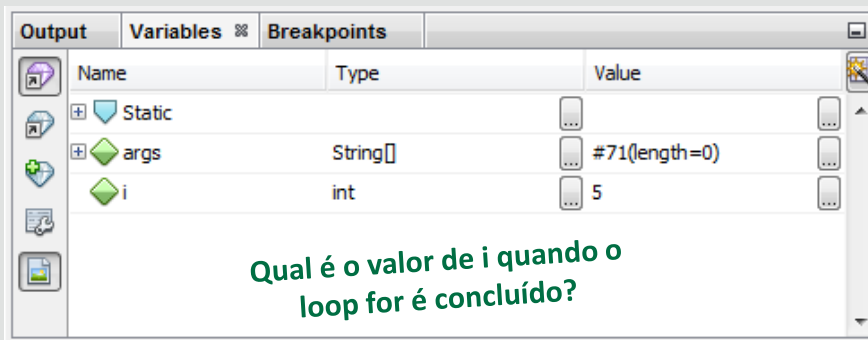

Expressão Update

- Essa expressão é executada depois de cada iteração do loop for
- Ela é usada para atualizar o contador de loops

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

Exercício 1, Parte 1

- Crie um novo projeto e adicione o arquivo `Countdown.java` a ele
- Defina um ponto de interrupção em `Countdown.java` e observe...
 - Como o loop `for` afeta a execução do código
 - Como o valor de `i` muda



Exercício 1, Parte 2

- É possível modificar o código para contar de 0 a 5?
- É possível modificar o código para contar todos os números pares de 0 a 20?

Eu Preciso da Expressão Update?

- E se eu escrevesse meu loop desta forma?

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

- Isso também funciona!
- Mas pode ser que você não queira codificar dessa forma porque seu loop poderia tornar-se muito complicado

Omitindo Expressões no Loop for

- Cada expressão no cabeçalho é opcional
- Mas há riscos quando você omite uma expressão:
 - Nenhuma inicialização:
 - Nenhuma inicialização é feita
 - Pode não haver um contador de loops
 - Nenhuma condição:
 - A condição do loop sempre é considerada como sendo verdadeira
 - O loop é um loop infinito
 - Nenhuma atualização:
 - Nenhuma operação de incremento é executada
 - O contador de loops mantém o mesmo valor

Omitindo Todas as Expressões no Loop for

- Examine o código a seguir:
 - Todas as três expressões no loop for podem ser omitidas
 - O loop é repetido infinitamente

```
for(;;){  
    System.out.println("Bem-vindo ao Java");  
}//fim for
```

Exercício 2


- Adicione o arquivo `InfiniteLoop.java` ao projeto que você criou para o exercício 1
- Execute `InfiniteLoop.java` e observe a saída
- Modifique o loop `for` em `InfiniteLoop.java` para imprimir “**Olá**” cinco vezes

Várias instruções dentro do corpo de um loop

- Para executar várias instruções em um corpo...
- Inclua as instruções em um par de chaves
- Caso contrário, só a primeira instrução no corpo será executada

```
for(int i = 1; i <= 5; i++)  
System.out.println(i);  
System.out.println("segunda linha");
```

- Saída:



```
1  
2  
3  
4  
5  
segunda linha
```


Um Uso do Loop for

- O loop for oferece uma maneira complicada de iteração em um intervalo de valores
- Repetição sem o loop for:

```
//Imprime o quadrado de 1 a 5
System.out.println("1 ao quadrado = " + 1 * 1);
System.out.println("2 ao quadrado = " + 2 * 2);
System.out.println("3 ao quadrado = " + 3 * 3);
System.out.println("4 ao quadrado = " + 4 * 4);
System.out.println("5 ao quadrado = " + 5 * 5);
```

- Repetição com o loop for:

```
for(int i = 1; i <= 5; i++){
    System.out.println("i ao quadrado = " + i * i);
} //fim for
```

i É o Contador de Loops

- Todo exemplo que vimos baseia-se no contador de loops

```
for(int i = 1; i <= 5; i++){  
    System.out.println("i ao quadrado = " + i * i);  
}//fim for
```

- i pode:
 - Ser impresso
 - Ter seu valor alterado
 - Ser usado em cálculos
- Isso é ótimo para:
 - Contar
 - Calcular valores rapidamente

Entendendo o Escopo da Variável

- Mas `i` só existe dentro do loop `for`
 - Isso é conhecido como escopo de `i`
 - `i` não existe quando o loop `for` termina
 - Se `i` for usado para calcular valores, nunca obteremos esses valores fora do loop `for`
- Você percebeu que `i` desapareceu quando `Countdown.java` foi depurado?

```
for(int i = 1; i <= 5; i++){  
    System.out.println("i ao quadrado = " + i * i);  
}//fim for
```

Escopo da Variável: Exemplo

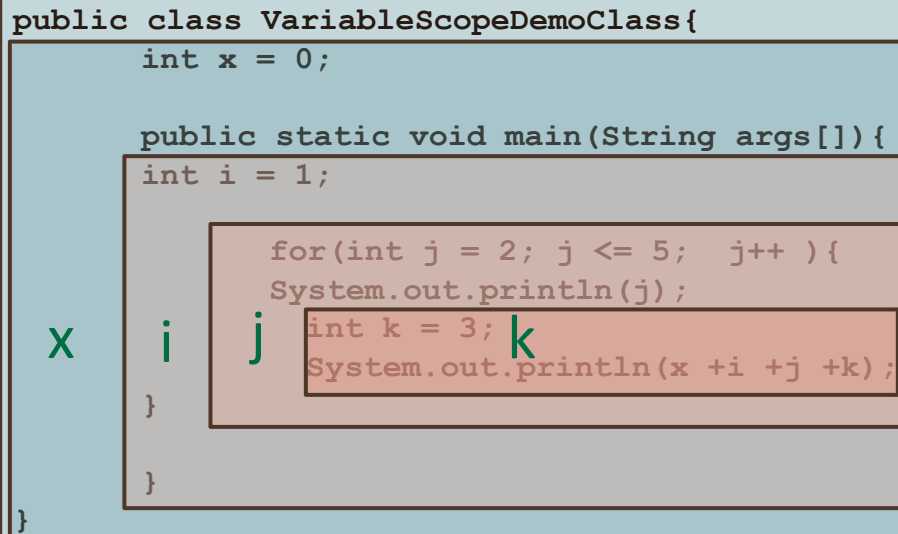
- A variável `i` declarada no loop `for` é uma variável local e não pode ser acessada fora do loop
- O erro do compilador é gerado na linha 8

```
1 public class VariableScopeDemo {
2
3     public static void main(String args[]){
4
5         for(int i = 0; i <= 5; i++ ){
6             System.out.println("i: " +i);
7         }//fim for
8     → System.out.println("i: " +i);
9     }//fim do método main
10 }//fim da classe VariableScopeDemo
```

Escopo da Variável

- As variáveis não podem existir antes nem fora de seu bloco de código

```
public class VariableScopeDemoClass{  
    int x = 0;  
  
    public static void main(String args[]){  
        int i = 1;  
  
        for(int j = 2; j <= 5; j++){  
            int k = 3;  
            System.out.println(x + i + j + k);  
        }  
    }  
}
```



Outro Uso dos Loops

- Suponha que você precise encontrar a soma de quatro membros

```
import java.util.Scanner;
public class Add4Integers {
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        System.out.println("This program adds four numbers.");
        System.out.println("Type each number, followed by Enter.");
        int n1 = in.nextInt();
        int n2 = in.nextInt();
        int n3 = in.nextInt();
        int n4 = in.nextInt();
        int total = n1 + n2 + n3 + n4;
        System.out.println(" O total é " + total + ".");
    } // fim do método main
} // fim da classe Add4Integers
```

Outro Uso dos Loops

- Esse método é inconveniente de ser programado se você deseja adicionar 100 valores

```
int n1 = in.nextInt();  
int n2 = in.nextInt();  
int n3 = in.nextInt();  
int n4 = in.nextInt();  
...  
int n100 = in.nextInt();  
int total = n1 + n2 + n3 + n4 +... + n100;
```

- Um loop for pode tornar esse programa mais curto?
- Um loop for pode ajudar a encontrar a soma de números inteiros de uma variável?

Usando o Escopo com Loops for

- Isso pode ser resolvido usando...
 - Um loop for com variáveis de escopo diferente

```
import java.util.Scanner;
public class PracticeCode {
public static void main(String[] args){
    Scanner in = new Scanner(System.in);
    int N = 100;
    int total = 0;
    System.out.println("This program adds " + N + " numbers.");
    for(int i = 0; i < N; i++){
        System.out.println("Enter your next number:");
        int value = in.nextInt();
        total += value;
    } //end for
    System.out.println("The total is " + total + ".");
} //end method main
```

Observação para os Instrutores: as caixas de código nesses dois slides devem estar na mesma posição.

Escopo

```
import java.util.Scanner;
public class PracticeCode {
public static void main(String[] args){
    Scanner in = new Scanner(System.in);

    int N = 100;
    int total = 0;
    System.out.println("This program adds " + N + " numbers.");
    for(int i = 0; i < N; i++){
        System.out.println("Enter your next number:");
        int value = in.nextInt();
        total += value;
    }//end for
    System.out.println("The total is " + total + ".");
} //end method main
```

N

total

i

value

Observação para os Instrutores: as caixas de código nesses dois slides devem estar na mesma posição

Exercício 3

- Adicione o arquivo `ScopeTest.java` ao projeto que você criou para o exercício 1
- `ScopeTest.java` está quebrado
- É possível corrigi-lo?
- Você deverá receber a seguinte saída:

```
-64 32 16 8 4 2 1
```

```
-0 1 2 3 4 5
```

```
-5 4 3 2 1 0
```

```
-2 4 8 16 32 64
```

Variável Já Definida

- i é criado antes do loop for
- Não pode haver outro i dentro do mesmo escopo
- Uma dessas variáveis precisa de um nome diferente

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

```
    int i = 0;
```

i

```
    for(int i = 64; i >0; i=i/2 ){  
        System.out.print(i +" ");  
    }
```

```
}
```

Fora do Escopo

- j não pode existir fora escopo em que foi criado
- Poderá ser criado outro j se os escopos não se sobrepuserem

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

```
    for(int j = 0; j<=5; j++){  
        j System.out.print(j + " ");  
    }
```

```
    for(int j = 5; j>=0; j--){  
        j System.out.print(j + " ");  
    }
```

```
    for(int k = 2; k<=64; k=k*2){  
        k System.out.print(j + " ");  
    }
```

```
}
```

Preciso da Expressão Initialization?

- E se eu escrevesse meu loop desta forma?

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

- Isso também funciona!
 - Mas i existe fora do escopo do loop for
 - Se i for apenas um contador de loops, a variável estará consumindo memória
 - Mantenha o escopo restrito (o mais restrito possível)
 - Variáveis perdidas complicam o código e aumentam a possibilidade de bugs

Resumo

- Nesta lição, você deverá ter aprendido a:
 - Entender os componentes do loop for padrão
 - Entender como criar e usar um loop for
 - Entender o escopo de uma variável
 - Entender técnicas de depuração
 - Explicar como loops infinitos ocorrem em Java



The logo for Oracle Academy is centered on a light gray background. It features the word "ORACLE" in a bold, orange, sans-serif font. Below it, the word "Academy" is written in a smaller, dark gray, sans-serif font. The entire logo is framed by two horizontal dark gray bars, one at the top and one at the bottom.

ORACLE

Academy