
ORACLE
Academy

Java Fundamentals

5-2

Declarações de Controle

ORACLE
Academy

Objetivos

- Esta aula abrange os seguintes tópicos:
  - Criar um loop while
  - Criar um loop do-while
  - Criar um loop for

O que é um Loop?

- Muitas tarefas diárias envolvem executar uma ação e, em seguida, repetir o mesmo procedimento ou ação em um objeto diferente
- Ao dobrar roupas limpas, há três etapas básicas:
  - Pegar a peça de roupa
  - Dobrá-la
  - Separá-la

- Para cada peça de roupa, estas ações são repetidas
- Cada vez que você executa a ação, somente sua entrada (a peça de roupa específica) é diferente

Loops

- Na programação, há momentos em que você quer trabalhar com várias entradas, mas quer executar a mesma lógica para cada item de entrada
- Um loop permite que você tenha uma série de entradas com o mesmo código
- Os loops começarão no início de uma parte do código, executarão a lógica que você deseja e, em seguida, retornarão para o início do loop com a nova entrada, pronta para executar o código mais uma vez

Por que os Loops são Úteis?

- Suponha que você tenha uma lista de dez números e deseja localizar a soma desses números
- Você pode criar uma instrução como esta:

```
sum = num1 + num2 + num3 + num4 + … + num10;
```

- Embora este código seja bem simples, usar um loop simplificará ainda mais o código

```
loop (loop condition){
   input currentNumber
   sum = sum + currentNumber;
}end loop
```

Quando o loop é executado pela primeira vez,
a entrada aceitará um número e cada vez que o loop for executado, a soma aumentará conforme esse número.

Controle de Loop: Interrompendo o Loop

- Para o código entrar em um loop e executar o código nele, a condição do loop deve ser verdadeira
- Para finalizar o loop, a condição do loop deve ser falsa
- Ao criar loops no Java, uma condição deve ser alterada de verdadeiro para falso, para que o código saia do loop

Existe uma exceção a essa regra (a instrução break), mas isso será abordado posteriormente nesta seção.

Interrompendo Condições

- Um loop precisa de uma condição de interrupção, que pode ser especificada como:
  - Um conjunto de vezes para executar o código
  - Uma condição booliana que é alterada no código para fazer o loop interromper a execução

Tipos de Loops

- O Java tem três tipos básicos de loops que trabalham com esses dois tipos de condições de interrupção:
  - while loops
  - for loops
  - do-while loop

Loops de pré-teste:
A condição é testada antes da execução do loop. Se a condição for falsa, o loop será interrompido ou nunca poderá ser executado.

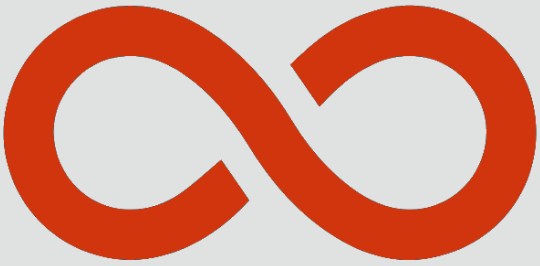
Loop de pós-teste:
A condição é testada após cada execução do loop. Se a condição for falsa, o loop ainda será executado uma vez, mas será interrompido no fim do loop.

O Loop while

- O loop while é projetado para fazer loop enquanto algo permanece verdadeiro
- Condição do exemplo:
  - Enquanto houver mais números para informar
- Pense no exemplo como uma condição verdadeira/falsa ou uma condição booliana:
  - Se a condição "Há mais números para informar" for verdadeira, continuar aceitando a entrada
  - Quando a condição "Há mais números para informar" for verdadeira, interromper a entrada

Loops infinitos

- Se não permitir a alteração da condição, o loop será executado para sempre como um loop infinito



Para sair de um programa preso em um loop infinito, basta fechá-lo.
Na maioria dos casos, um loop infinito é algo que devemos evitar. No entanto, existem situações em que o código precisa ser executado continuamente. Pense nos motores dos helicópteros, nos monitores cardíacos, etc. Você não gostaria que eles parassem.

Sintaxe Java para Loops while

- Com um loop while, o Java usa a sintaxe:

```
while(condition is true){
   //logic
}//fim while
```

- Semelhante às instruções if, os parâmetros do loop while podem ser dos tipos boolianos ou podem ser iguais a um valor booliano
- As instruções condicionais (<, >, <=, >=, !=, ==) são iguais aos valores boolianos
  - Exemplos:

```
while (num1 < num2)
while (isTrue)
while (n !=0)
```

ORACLE Academy

Exemplo de Loop While

- A implementação que usa um loop while é mostrada abaixo
- Este exemplo adiciona uma sequência de dez inteiros que são inseridos pelo usuário

Exemplo de Loop While

```
import java.util.Scanner;
public class LoopPractice{
   public static void main(String[] args){
      Scanner in = new Scanner(System.in);
      int numInputs = 0;
      int sum = 0;
      while(numInputs < 10){  //condition to be tested each time
                              //loop is executed
         input = in.nextInt();     //user inputs a number
          sum+=input;              //add user input to sum
               numInputs++;        //statement that will change the
                                   //loop condition
        }//fim while
        System.out.println("The sum of those ten numbers is: "
                           + sum);
   }//fim do método main
}//fim da classe LoopPractice
```

Observe que a condição do loop é testada antes de cada execução do loop. Este é um loop de pré-teste.

Usando um Loop while com Métodos de String

- Um palídromo é uma palavra pronunciada da mesma forma da esquerda para a direita ou ao contrário
- Exemplos:
  - arara, esse e asa
- Grava um código Java que solicita uma palavra e retorna verdadeiro se for um palíndromo, e falso se não for
- Este código deve:
  - Calcular o tamanho da palavra
  - Comparar a primeira e última letras, se forem correspondentes
  - Comparar as letras até o meio da palavra ser atingido

Usando um Loop while com Exemplos de Métodos de String

- Código do método de palíndromo:

```
public class PalindromeTester{
    public static void main(String[] args){
       Scanner in = new Scanner(System.in);
       System.out.println("Enter a word:");
       String word = in.next();
       int firstPosition = 0;
       int lastPosition = word.length() - 1;
       boolean isAPalindrome = true;

       while(firstPosition < lastPosition){
          if(word.charAt(firstPosition)!=word.charAt(lastPosition))
             isAPalindrome = false;
          firstPosition++;
          lastPosition--;
       }//fim while

       if(isAPalindrome)
          System.out.println("The word is a Palindrome");
       else
          System.out.println("The word is not a Palindrome");
    }//fim do método main
}//fim da classe PalindromeTester
```

O Loop do-while

- O loop do-while:
  - É um loop pós-teste
  - É um loop while modificado que permite que o programa execute o loop uma vez antes de testar a condição booliana
  - Continua até a condição se tornar falsa
- Se não permitir a alteração da condição, o loop será executado para sempre como um loop infinito

O loop do-while é bom para reunir entradas do usuário, testá-las e, se forem inválidas, reuni-las novamente.

O Exemplo do Inseto do Loop do-while

- Considere o movimento de um inseto voando que pousa em cada flor que ele vê
- Para recriar este movimento do inseto no código, você pode usar um loop do-while

Insect Action Pseudocode

- Pseudocódigo do movimento de um inseto voando que pousa em cada flor que ele vê:
  - O inseto está voando inicialmente antes de qualquer condição ser testada
  - O inseto continua voando até localizar uma flor
  - A condição que será testada é noFlowerSpotted
  - Quando noFlowerSpotted for verdadeira, o inseto continuará voando
  - Quando for falsa, o inseto interromperá o loop e pousará na flor

Sintaxe Java para Loops do-while

- O loop do-while usa a mesma lógica booliana que é usada para um loop while regular
- O bloco do código do é executado uma vez e a condição while é testada no fim de cada execução do código
  - Primeiro, o bloco do código do é executado
  - Em seguida, a condição é testada
  - Isso é repetido até a condição se tornar falsa
- sintaxe do loop do-while:

```
do{
   //statements to repeat go here
} while(condition);
```

Não esqueça o ponto e vírgula ou o código não será compilado.

O Código do Exemplo do Inseto do Loop do-while

- O inseto voa até encontrar uma flor, depois pousa
- Etapa 1:
  - O inseto começa a voar antes de testar se pode ver uma flor
- Etapa 2: Teste se o inseto vê uma flor
  - Se o inseto não vir uma flor, repita as etapas 1 e 2
  - Se o inseto vir uma flor, vá para a etapa final

```
do{
   insect.fly();                //Step 1
}while(noFlowerFound)           //Step 2
insect.land();                  //Final Step
```

Exemplo de loop do-while

- A implementação que usa um loop while é mostrada abaixo
- Este exemplo adiciona uma sequência de dez inteiros que são inseridos pelo usuário
- Você pode ver e explicar as diferenças deste exemplo e do loop pré-teste?

```
import java.util.Scanner;
public class LoopPractice{
  public static void main(String[] args){
     Scanner in = new Scanner(System.in);
     int numInputs = 0;
     int sum = 0, input = 0;
     do{
        System.out.println("Enter a number");
        input = in.nextInt();  //user inputs a number
        sum+=input;            //add user input to sum
        numInputs++;  //statement to change the loop condition
     }while(numInputs <  10);
     System.out.println("The sum of those ten numbers is: " + sum);
   }//fim do método main
}//end class LoopPractice
```

Observe que a condição do loop é testada após cada execução do loop. Este é um loop de pós-teste.

ORACLE Academy

Esse programa é funcionalmente o mesmo do slide 13.

Exemplo de loop do-while

```
import java.util.Scanner;
public class LoopPractice2{
   public static void main(String[] args){
      Scanner in = new Scanner(System.in);
      int repeat = 0;
      do{
         System.out.println("Enter a number");
         input = in.nextInt();    //user inputs a number
         sum+=input;          //add user input to sum
         System.out.println("Do you want to enter another number?");
         System.out.println("Enter 1 for yes or 2 for no:");
         repeat = in.nextInt();
      } while(repeat==1);
      System.out.println("The sum of the numbers is: " + sum);
   }//fim do método main
}//fim da classe LoopPractice2
```

O Loop For

- Recordando o exemplo de dobrar roupas
  - Se soubermos quantas peças de roupas temos, sabemos exatamente quando a condição "Há mais roupas" será falsa
  - para os loops informarem ao loop quando parar, dizendo explicitamente "Interromper quando o loop for executado uma vez para cada peça de roupa"
  - Por exemplo, se tivermos dez peças de roupas, podemos dizer ao loop, "Executar 10 vezes", desde que saibamos que após a décima execução, não haverá mais roupas

Sintaxe de Java do Loop For

- a sintaxe do loop For contém três partes:
  - Inicializando a lcv (variável de controle de loop)
  - Instrução condicional ou condição de interrupção
  - Atualizando o contador (indo para o próximo valor)
    - Pense em i como contador, iniciando em 0 e aumentando até i=timesToRun
- para a sintaxe do loop:

para convenções de nomeação do loop, geralmente usamos i como o nome da variável lcv.

```
for(int i=0; i < timesToRun; i++){
   //logic
}//fim for
```

Loop for Explicado

- O loop for tem os seguintes componentes:
- Inicializando a variável (i) do contador
  - Geralmente, i é definido como 0 (zero), com isso iniciando a contagem em 0
- O loop coInstrução condicional ou condição de interrupção
  - ntinuará sendo executado enquanto i < timesToRun, que significa que ele será executado timesToRun vezes
- Atualizando o contador (indo para o próximo valor)
  - Cada vez que o loop é executado, i é incrementado por um (o operador ++ adiciona um a i)

```
for(int i=0; i < timesToRun; i++){
    //statements to repeat
}//fim for
```

Exemplo 1 do Loop For

- Revise o código de exemplo abaixo para dobrar dez peças de roupa
- Você pode identificar três parte do loop For neste exemplo?

```
for(int i = 0; i < numFolded; i++)
{
   fold();
}//fim for
System.out.println("All Done!");
```

Exemplo 2 do Loop for

- Este exemplo é uma sequência de dez inteiros que são inseridos pelo usuário
  - Você pode ver e explicar as diferenças deste exemplo e do loop de pós-teste?

```
import java.util.Scanner;
public class LoopPractice{
   public static void main(String[] args){
         Scanner in = new Scanner(System.in);
         int numInputs = 10;
         int sum = 0;
         for(int i = 0; i < numInputs; i++){
            input = in.nextInt();    //user inputs a number
            sum+=input;  //add user input to sum
         }//fim for
         System.out.println("The sum of those ten numbers is: " + sum);
   }//fim do método main
}//fim da classe LoopPractice
```

A condição do loop é testada antes de cada execução do loop. Este é um loop de pré-teste.

Esse programa é funcionalmente o mesmo do slide 13.

Que Loop Devo Usar?

Tipo de Loop	Definição	Quando Usar
while	Loop de pré-teste que repete-se até uma condição especificada ser falsa.	Use quando não tiver certeza do número de vezes que o loop deve ser executado ou mesmo se souber
do-while	O loop pós-teste que executa o loop antes de testar a condição, em seguida, repete-se até a condição ser falsa.	Use quando você souber que o código deve ser executado pelo menos uma vez e, provavelmente, mais vezes, dependendo da condição.
for	Loop que contém um contador inicializado e incrementa o contador com cada execução pelo loop. Repete-se até a condição se tornar falsa.	Use quando precisar executar um loop para um número específico de vezes, ou quando precisar incrementar pelo conjunto de dados. O contador também pode ser usado como um índice para acessar dados de um item de uma vez.

Há uma variação do loop for denominado loop for-each. Ele será abordado na Seção 6, pois trabalha com matrizes.

Usando break e continue

- break e continue são palavras-chave do Java que ajudam a controlar o fluxo de seu programa
- A palavra-chave break é útil para instâncias nas quais você quer sair de um loop em um ponto especificado que é diferente da instrução da condição
- Usar break em um loop fará com que o código saia do loop

Sempre que um loop usa um break, deve ser bem documentado no cabeçalho do loop para que os programadores que virem o código saibam que o loop poderá fechar mesmo se a condição ainda for verdadeira.

Usando break e continue

- A palavra-chave continue é útil para casos especiais nos quais você quer excluir o código de um elemento específico em uma lista
- Usar continue fará com que o código ignore o restante do código no loop e avalie a instrução da condição (em um loop for, a lcv também será incrementada)

A instrução continue pode ser considerada um loop "do over". A lcv é a variável de controle de loop.

Exemplo de break

- Este exemplo solicitará uma entrada de dez vezes
  - Se o usuário informar o valor 999, o loop será encerrado, independentemente do valor de i (a lcv)

```
import java.util.Scanner;
public class BreakExample{
   public static void main(String[] args){
      Scanner in = new Scanner(System.in);
      int numInputs = 10, input = 0, sum = 0, stopLoop = 999;
      for(int i = 0; i < numInputs; i++){
         input = in.nextInt();     //user inputs a number
         if(input == stopLoop)     //if the number is 999, exit the
            break;                 //loop without adding to the sum
          else
             sum += input;         //if number is not 999, add it to sum
      }//fim for
      System.out.println("The sum of the numbers entered is: " + sum);
   }//fim do método main
}//fim da classe BreakExample
```

Exemplo de continue

- Dada uma lista de inteiros (você deseja emitir uma mensagem para todos os números ímpares e deseja ignorar os números pares), isso pode ser feito usando o seguinte código

```
import java.util.Scanner;
public class ContinueExample{
   public static void main(String[] args){
      Scanner in = new Scanner(System.in);
      int numInputs = 10, input = 0;
      for(int i = 0; i < numInputs; i++){
         input = in.nextInt();  //user inputs a number
         if(input % 2 == 0)     //if it's even
            continue;   //skip the remaining code in the loop, increment i,
                        //re-check the condition
         System.out.println("That number was odd");  //output only if odd
      }//fim for
   }//fim do método main
}//fim da classe ContinueExample
```

Exemplo de if, do-while e switch

```
import java.util.Scanner;
public class Section5Example{
   public static void main(String[] args){
      boolean quit = false;
      int num1 = 10, num2 = 6, answer = 0;
      char operand = '';
      Scanner in = new Scanner(System.in);
      do {
         System.out.println("Please enter a mathematical operand ");
         String input = in.next();
         char operand = input.charAt(0);
```

. . .

Exemplo de if, do-while e switch

```
...
         switch(operand) {
            case '*': answer = num1 * num2;
                      break;
            case '+': answer = num1 + num2;
                      break;
            case '-': answer = num1 - num2;
            case '/': answer = num1 / num2;
            default:  System.out.println("Invalid input.");
         }
         System.out.println("Quit? Y/N");
         if(in.next().equalsIgnorecase("Y"))
            quit=true;
      } while(!quit);
   }//fim do método main
}//fim da classe Section5Example
```

Terminologia

- Os principais termos usados nesta aula foram:
  - break
  - continue
  - loop do-while
  - loop for
  - loop while

Resumo

- Nesta aula, você deverá ter aprendido a:
  - Criar um loop while
  - Criar um loop do-while
  - Criar um loop for


ORACLE
Academy