

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

# ORACLE

## Academy

# Java Foundations

6-3

Usando Instruções break e continue

**ORACLE**  
Academy

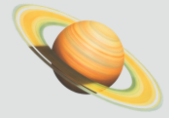


Copyright © 2020, Oracle e/ou suas afiliadas. Todos os direitos reservados.

# Objetivos

- Esta lição abrange os seguintes objetivos:
  - Usar uma instrução break para sair de um loop
  - Usar uma instrução continue para ignorar parte de um loop
  - Explicar a necessidade de comentários do loop





# Missão para os Anéis de Saturno

- Vamos considerar outro cenário para essa missão
  - Quando a nave espacial está girando ao redor de Saturno e tirando fotos, o braço robótico ou a câmera quebra
- Como você resolveria esse problema?
  - Se você tivesse que escrever um programa Java, qual construção usaria?
  - Vamos ver se o Java tem uma instrução que permita a você terminar um loop imediatamente

## Como Você Sai de um Loop Antes?

- Em geral, a única maneira de sair de um loop é quando a condição dele é avaliada como falsa
- No entanto, normalmente é conveniente terminar um loop antes quando determinadas condições são atendidas
- Nesses casos, continuar o loop seria uma perda de tempo do processador

## Como Você Sai de um Loop Antes?

- Você pode usar duas instruções Java para terminar um loop antes:
  - **break**
  - **continue**

## Usando uma Instrução break em um Loop

- Quando uma instrução break é executada dentro de um loop, a instrução do loop termina imediatamente
- O programa continua a ser executado com a instrução após a instrução do loop
- Sintaxe:

```
break;
```

# Usando break em um Loop while

```
while(condition){  
    statement1;  
    statement2;  
    break;  
    statement3;  
    statement4  
}
```

O controle passa para a  
instrução fora do loop

```
statement; ← [instrução fora do loop while]
```



# Usando break em um Loop while: Exemplo

• Saída: 0 1 2 3

- A execução do loop é terminada quando o contador de loop fica igual a 4

```
public static void main(String[] args) {  
    int i = 0;  
    while (i < 10) {  
        System.out.println(i + "\t");  
        i++;  
        if (i == 4) {  
            break;  
        }  
    }  
}
```

No exemplo de código, embora o loop seja declarado para ser executado 10 vezes, a instrução break sai do loop depois de apenas 4 iterações. O último valor de i é a saída do console após o loop ser terminado.

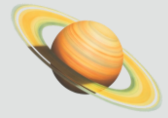
## Usando break em um Loop for

- Vamos escrever um programa para demonstrar uma instrução break em um loop for
- O programa deve ...
  - Ler 10 números do console
  - Calcular a soma dos números que o usuário informa
  - Se o usuário informar 999, termine o loop, independentemente do valor do contador de loops e sem adicionar a soma

# Usando break em um Loop while: Exemplo

```
public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int numInputs = 10, input = 0, sum = 0, stopLoop = 999;
    System.out.println("Enter 10 numbers");
    for (int i = 0; i < numInputs; i++) {
        input = in.nextInt();
        if (input == stopLoop){
            break;
        }
        else {
            sum += input;
        } //endif
    } //end for
    System.out.println("The sum of the numbers:" + sum);
} //end method main
```

# Missão para os Anéis de Saturno: Implementando as Condições



- Vamos usar um loop while e uma instrução break para implementar as condições especificadas no início da lição

```
public static void main(String[] args) {  
    long distTravelled = 0;  
    long maxDistance = 50000000;  
  
    while (distTravelled <= maxDistance) {  
        if (isCameraBroken()) {  
            break;  
        }  
        else {  
            cameraSnap();  
        }  
    }  
    shipRotate();  
}
```

**ORACLE**  
Academy

JFo 6-3  
Using break and continue Statements

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

12

Suponha que existam métodos disponíveis da seguinte forma:

isCameraBroken() – retorna verdadeiro ou falso

cameraSnap() – tira uma foto

shipRotate() – nave espacial gira ao redor de Saturno

Neste exemplo, se a câmera estiver quebrada, as instruções restantes dentro do loop while serão ignoradas e o controle passará para a instrução fora do loop while .

Ou seja, a nave espacial continua a girar ao redor de Saturno.



## Exercício 1

- Importe e abra o projeto `BreakContinueEx`
- Examine `ComputeSum.java`
- Implemente o seguinte:
  - Aceite 10 números do usuário
  - Calcule a soma dos números informados
  - Quando 0 for informado, o programa deverá sair e exibir a soma dos números

Saída esperada:

Informe 10 números.

Informe 0 para sair.

1

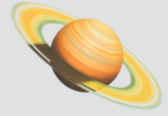
2

3

0

A soma dos números informados é 6.

# Missão para os Anéis de Saturno: Outro Cenário



- Vamos considerar outro cenário para essa missão
  - Enquanto a nave espacial está voando ao redor de Saturno e tirando fotos dos anéis de Saturno...
    - Se a visibilidade for zero, não tire fotos
    - Caso contrário, continue a tirar as fotos
- Como você resolveria esse problema?
  - Se você tivesse que escrever um programa Java, qual construção usaria?
  - Vamos ver se o Java tem uma instrução que permita a você ignorar a iteração atual do loop

## Usando continue em um Loop

- Às vezes, pode ser que você queira ignorar a iteração atual em um loop e não terminar o loop propriamente dito
- Você pode usar uma instrução continue para ignorar a iteração atual em um loop:
  - Ou seja, o restante do corpo do loop é ignorado até o fim do loop
  - Mas isso não termina o loop
  - Quando o programa chega no fim do loop, ele retorna para testar a condição de continuação do loop
- Sintaxe:

```
continue;
```

# Usando continue em um Loop while

```
while(condition){
```

```
    statement1;
```

```
    statement2;
```

```
    continue;
```

```
    statement3;
```

```
    statement4
```

```
}
```


```
statement;  [instrução fora do loop while ]
```

O controle passa para a condição do loop

Essas instruções são ignoradas na iteração atual



# Usando continue em um Loop for

```
for (i = 0; i < 10; i++) {  
    statement1;  
    statement2;  
    continue;  O controle passa para a condição do  
    statement3; } loop  
    statement4; } Essas instruções são ignoradas na iteração atual  
} //end for
```

# Usando continue em um Loop for

- Saída: `0 1 2 3 5 6 7 8 9`
  - A saída não inclui 4
  - Devido à instrução `continue`, a execução do loop é ignorada quando o contador de loops é 4

```
public static void main(String[] args) {  
    for (int i = 0; i < 10; i++) {  
        if (i == 4) {  
            continue; //control jumps to update i++  
        } //endif  
        System.out.print(i + "\t");  
    } //end for  
} //end method main
```

## Colocando Tudo Junto

- Vamos escrever um programa usando o loop while e a instrução continue
- O programa deve ...
  - Calcular a soma dos números entre 1 e 99 usando o loop while
  - Se o número for um múltiplo de 10, a iteração atual deverá ser ignorada e o número não deverá ser adicionado à soma
  - Exiba a soma no console

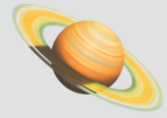
# Calculando a Soma de Números

```
public static void main(String[] args) {  
    int counter = 0;  
    int sum = 0;  
    while (counter < 100) {  
        counter++;  
        if (counter % 10 == 0) {  
            continue;  
        }  
        else {  
            sum += counter;  
        }  
    }  
    System.out.println("Sum of 1 - 99: " + sum);  
}
```

Este é um múltiplo de 10?  
Caso seja, ignore a iteração  
atual

Esta é a saída deste exemplo:  
Soma de 1 - 99: 4500

# Missão para os Anéis de Saturno: Implementando as Condições



- Vamos usar um loop `while` e uma instrução `continue` para implementar as condições especificadas no início deste tópico.

```
public static void main(String[] args) {  
    long distTravelled = 0;  
    long maxDistance=50000000;  
  
    while (distTravelled <= maxDistance) {  
        if (getVisibility() == 0) {  
            continue;  
        }  
        else {  
            cameraSnap();  
        }  
    }  
    shipRotate();  
}
```

**ORACLE**  
Academy

JFo 6-3  
Using break and continue Statements

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

21

Suponha que existam métodos disponíveis da seguinte forma:

`getVisibility()` = o número de milhas de visibilidade da nave espacial para Saturno

`isCameraBroken()` – retorna verdadeiro ou falso

`cameraSnap()` – tira uma foto

`shipRotate()` – a nave espacial gira ao redor de Saturno

Neste exemplo, se a visibilidade for zero, as fotos dos anéis de Saturno não serão tiradas, o controle será transferido para a condição e o programa continuará a ser executado com a próxima iteração.

Essas ações ocorrem porque você usou a instrução `continue`.



## Exercício 2

- Importe e abra o projeto `BreakContinueEx`
- Examine `CountChar.java`
  - O programa é usado para contar o número de ocorrências do caractere 'w' na string
  - Modifique o programa para...
    - Resolver o erro de sintaxe
    - Imprimir a contagem de caracteres 'w'
  - Saída Esperada:
    - Número de w: 3



## Exercício 3

- Importe e abra o projeto `BreakContinueEx`
- Examine `BreakContinue.java`
- Modifique o programa usando as instruções `break` e `continue`...
  - Se o número for par, ele não deverá ser impresso
  - A execução do loop deve parar quando o valor do contador de loops é 7

### Saída Esperada:

O número é 1  
O número é 3  
O número é 5  
O número é 7

## Escrevendo Comentários de Loop

- É uma boa prática adicionar comentários apropriados aos loops
- Caso contrário...
  - O código tende a ficar confuso de ser lido
  - Você não conseguirá compreender a lógica muito facilmente
- Ele ajuda a entender...
  - Variáveis de loop usadas e sua finalidade
  - Lógica do loop
  - Número de iterações
  - Execução das instruções no loop dependendo da condição, dos critérios ou de ambos



# Escrevendo Comentários de Loop: Exemplo

```
public static void main(String[] args) {  
  
    Scanner in = new Scanner(System.in);  
    int numInputs = 10, input = 0;  
  
    //This loop is executed 10 times  
    for (int i = 0; i < numInputs; i++) {  
        input = in.nextInt(); //user inputs a number  
  
        if (input % 2 == 0) { //if the number is even skip the  
            continue;        //remaining code and restart the loop  
        }//endif  
  
        System.out.println("That number was odd");  
    }//end for  
}//end method main
```



## Exercício 4

- Importe e abra o projeto BreakContinueEx
- Examine Divisors.java
- O programa encontrará todos os divisores de um número



## Exercício 4

- Modifique o programa para incluir comentários do loop sobre...
  - Variáveis do loop usadas
  - Lógica do loop
  - Número de iterações
  - Condição usada
  - Fluxo de controle no loop

# Resumo

- Nesta lição, você deverá ter aprendido a:
  - Usar uma instrução break para sair de um loop
  - Usar uma instrução continue para ignorar parte de um loop
  - Explicar a necessidade de comentários do loop



