

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

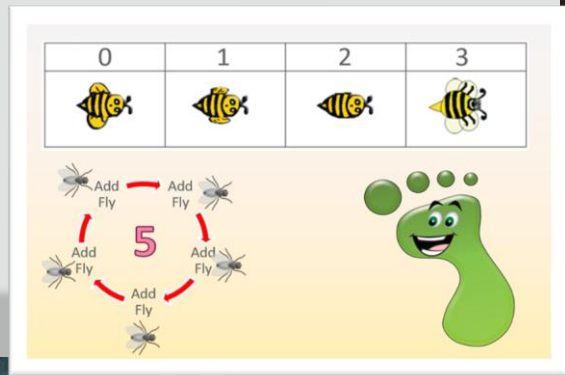
## Academy

# Java Fundamentals

3-10

Loops, Variáveis e Matrizes

**ORACLE**  
Academy



Copyright © 2022, Oracle e/ou suas empresas afiliadas. Oracle, Java e MySQL são marcas comerciais registradas da Oracle Corporation e/ou de suas empresas afiliadas. Outros nomes podem ser marcas comerciais de seus respectivos proprietários.

# Objetivo

- Esta lição abrange os seguintes objetivos:
  - Criar um loop while em um construtor para criar um mundo
  - Descrever um loop infinito e como impedir sua ocorrência
  - Usar uma matriz para armazenar diversas variáveis usadas para criar um mundo
  - Criar uma expressão com operadores lógicos
  - Descrever o escopo de uma variável local em um método



# Usando Loops

- Escrever instruções de programação no construtor Mundo é uma forma eficiente de criar novas instâncias com parâmetros transmitidos a elas
- No entanto, uma forma mais eficiente de criar várias instâncias é usar um loop



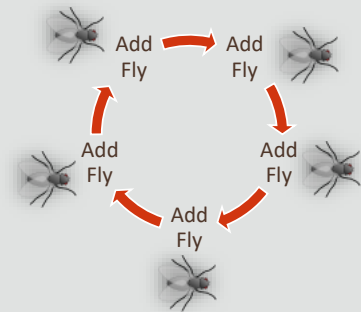
Loop é uma instrução que pode executar uma seção de código várias vezes. Há vários tipos de loops na programação Java

A repetição de linhas de códigos na programação é uma tarefa comum. Adicionar um loop ao redor de uma seção de código é uma tarefa que será observada com frequência no código de programação.

A maioria das linguagens de programação tem três métodos de loop básicos. Examinaremos todos eles durante o curso.

# Loop while

- O loop while executa uma instrução ou um conjunto de instruções várias vezes enquanto uma condição é verdadeira
- Por exemplo, com um loop while, você pode:
  - Criar 50 instâncias de uma vez
  - Executar um método 10.000 vezes
  - Criar uma instância toda vez até que a tecla "s" for pressionada



Podemos usar qualquer loop para qualquer ocasião, mas alguns são mais adequados para determinados cenários.

# Componentes do Loop while

- Componentes de um loop while:
- Palavra-chave while Java
- Condição entre parênteses
- Uma ou mais instruções

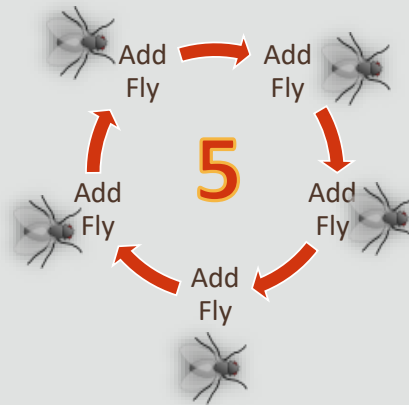


```
while (condition)
{
    statement;
    statement;
}
```

Lembre-se de que a condição só deve retornar verdadeiro ou falso.

# Controlar a Execução do Loop while

- Componentes para controlar quantas vezes o loop while é executado:
- Variável de loop:
  - contador que diz quantas vezes o loop deverá ser executado (geralmente denominado i)
- Operadores de controle
- Variável de local



## Variáveis de Local

- Geralmente, uma variável de local é usada nos constructos dos loops
- Embora seja semelhante a um campo, é diferente porque:
  - É declarada dentro do corpo do método, não no início de uma classe
  - Não pode ter um modificador de visibilidade (público ou privado) na frente de sua definição
  - Ela só existirá até que a execução do método atual termine e, em seguida, será apagada da memória



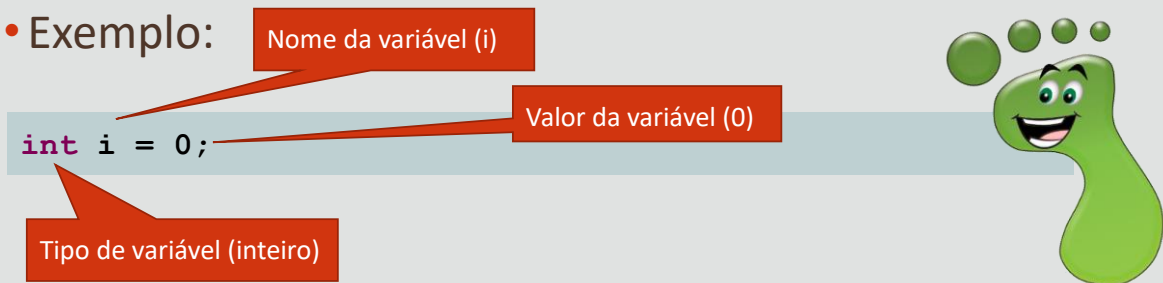
Variável de local é uma variável declarada dentro do corpo do método para armazenar valores temporariamente, como referências a objetos ou inteiros

Embora uma variável de instância tenha visibilidade dentro da classe inteira (pode ser acessada), não é possível ver uma variável de local fora dos colchetes nos quais ela foi declarada. Normalmente, isso é chamado de escopo.



## Declarar uma Variável de Local

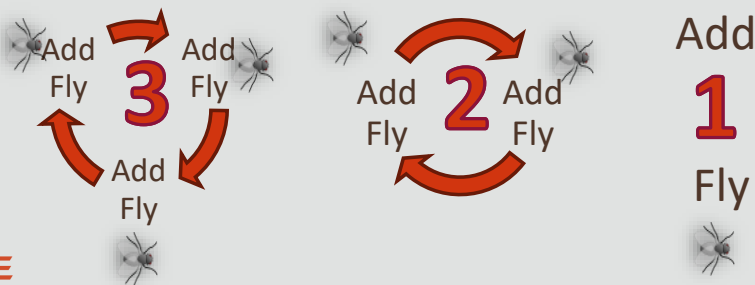
- Para criar um loop while, primeiro declare a variável de local e atribua a ela um valor
- Para declarar uma variável de local:
  - Declare o tipo de variável (referência a objeto ou inteiro)
  - Nomeie a variável
  - Inicialize a variável para um número (geralmente, zero)
- Exemplo:



A letra "i", geralmente, é usada como o nome da variável de local. Se você tiver um loop dentro de outro loop depois a letra "j", em geral, "k" será usada. Acredita-se que essa letra seja usada na programação em virtude de sua origem na matemática, em que ela designava números inteiros. Em seguida, esse conceito foi empregado na linguagem Fortran.

## Criar a Condição

- Abaixo da variável inicializada, crie a condição que especifica quantas vezes o corpo do loop deverá ser executado
- O loop continuará a ser executado enquanto a condição for verdadeira
- Use um operador de controle para parar a execução quando for atingido o número especificado



ORACLE  
Academy

JF 3-10  
Loops, Variáveis e Matrizes

Copyright © 2022, Oracle e/ou suas empresas afiliadas. Oracle, Java e MySQL são marcas comerciais registradas da Oracle Corporation e/ou de suas empresas afiliadas. Outros nomes podem ser marcas comerciais de seus respectivos proprietários.

10

Veremos um exemplo no próximo slide.

## Criar a Condição

- Exemplo:
- Executar o corpo do loop enquanto o número de execuções for inferior, mas não igual a 10
- Quando o loop tiver sido executado 10 vezes (0-9), ele parará

```
int i = 0;  
while (i < 10){  
    <o código a ser repetido é inserido aqui>  
}
```



Qualquer um que tenha feito programação antes poderá notar um problema. Veremos isso em alguns slides.

## Inserir as Instruções a Serem Executadas

- Em colchetes, insira as instruções a serem executadas
- Exemplo:
  - Adicione 10 novos objetos Mosca com uma velocidade e uma direção específicas

```
int i = 0;  
while (i < 10)  
{  
    addObject (new Fly (2, 90), 150, 100);  
}
```



# Incrementar a Variável de Loop

- Incremente a variável de loop da seguinte forma:
  - Insira uma instrução no final do corpo do loop para aumentar a variável de loop em 1 toda vez que o loop for executado
  - Use colchetes fechados para finalizar a instrução
- Isso mudará a variável com cada loop para garantir que o loop não ocorra indefinidamente

```
int i = 0;
while (i < 10)
{
    addObject (new Fly (2, 90), 150, 100);
    i = i + 1;
}
```



Lembre-se que `i = i + 1` normalmente seria escrito como `i++`;

Se esquecermos de adicionar essa linha, o loop do código nunca será interrompido. Isso é chamado de loop infinito.

## Exemplo de Loop while

- Quando inserido no construtor BeeWorld, ele cria 10 moscas quando o mundo é inicializado

```
/**
 * Prepare the world for the start of the program.
 * That is: create the initial objects and add them to the world.
 */
private void prepare()
{
    addObject (bee, 150, 100);
    addObject(new Spider(), 510, 360);

    int i = 0;
    while(i<10){
        addObject(new Fly(1,90), 503, 70);
        i = i+1;
    } //end while
} //end method prepare
```

Como isso está no construtor, ele sempre será chamado uma vez.

## Colocação de Objetos e Loops while

- No exemplo anterior, quando o construtor é executado, todas as instâncias são colocadas nas mesmas coordenadas
- Isso faz com que elas fiquem umas em cima das outras
- Crie uma expressão que calcule o tamanho de uma instância e, em seguida, coloque as instâncias subsequentes em coordenadas aleatórias diferentes
- A velocidade máxima das moscas será aumentada dentro do loop

## Calcular a Colocação das Instâncias

- Para programar instâncias para permanecerem em coordenadas diferentes, substitua a coordenada x fixa para a largura do objeto com uma expressão que inclua:
  - Variável i
  - Coordenadas x e y aleatórias

```
private void prepare()
{
    addObject (bee, 150, 100);
    addObject(new Spider(), 510, 360);

    int i = 0;
    while(i<10){
        int xCoord = Greenfoot.getRandomNumber(this.getWidth());
        int yCoord = Greenfoot.getRandomNumber(this.getHeight());
        addObject(new Fly(i+1,90), xCoord, yCoord);
        i = i+1;
    } //end while
} //end method prepare
```

Os métodos getWidth() e getHeight() retornam a largura e a altura atuais do mundo. Então, se alterarmos as dimensões de mundo poderemos acessá-las sem alterar nosso código.



# Loops Infinitos

- Se não for estabelecido um final para o loop, ele continuará a ser executado e nunca será interrompido
- Loops infinitos são um problema comum na programação
- Um loop infinito é executado da seguinte forma:
  - A variável nunca muda
  - A condição sempre permanece verdadeira
  - O loop continua a ser executado indefinidamente

Chamamos de loop infinito quando ele continua a ser executado e não para porque seu fim não foi estabelecido



Você sempre deve verificar com cuidado sua lógica e código para garantir que o loop só será executando o número de vezes esperado.

# Animando Objetos com uma Tecla do Teclado

- Outra forma de animar um objeto é fazer com que o objeto altere a imagem exibida quando uma tecla do teclado é pressionada
- O pseudocódigo dessa ação:
  - Alternar entre quatro imagens quando uma tecla é pressionada
  - Quando a tecla esquerda é pressionada, mostrar image1
  - Quando a tecla direita é pressionada, mostrar a image2
  - Quando a Abelha não estiver girando e a imagem atual for a image1, mostrar image2, se não estiver girando, mostrar image1
  - O objeto precisa lembrar se a Abelha está girando ou não
    - Caso contrário, ele trocará rapidamente o objeto exibido, e a tecla do teclado não poderá controlá-lo

Basicamente, devemos alternar entre uma de duas imagens, a menos que estejamos girando. Se estivermos girando, exibir a imagem girando à direita ou à esquerda.

## Exemplo de Tecla do Teclado

- A Abelha deve se inclinar para a esquerda ou direita ao girar
- Quatro imagens são salvas no cenário:
  - Uma com a Abelha inclinando-se para a esquerda, uma inclinando-se para a direita e as duas últimas com a animação de voo normal
- O pseudocódigo dessa ação:
  - Se a Abelha estiver girando, não usar a animação de movimento das asas
  - Se girar para a esquerda, mostrar a imagem esquerda e se girar para a direita, mostrar a imagem direita
  - Lembrar se a Abelha estiver girando no momento

Ao armazenar o estado de uma ação atual, geralmente, usamos uma variável de classe.

## Declarar Variáveis de Classe

- Primeiro, adicione novos campos à classe Abelha para armazenar os giros para a esquerda e direita, e um campo para armazenar o estado de giro atual

```
public class Bee extends Actor
{
    private GreenfootImage image1;
    private GreenfootImage image2;
    private GreenfootImage imageLeft;
    private GreenfootImage imageRight;
    boolean isTurning;
    private int score;
    private int lives;

    /**
     * Bee - sets the initial values of the bee
     */
    public Bee()
    {
```

É uma prática recomendada usar bons nomes de variáveis de classe, principalmente ao definir valores booleanos. Poderíamos ter chamado a variável de notTurning, mas é mais trabalhoso pensar no que a variável notTurning realmente significa do que isTurning.

# Inicializar Variáveis

- Em seguida, inicialize nossas variáveis de classe no construtor

```
/**
 * Bee - sets the initial values of the bee
 */
public Bee()
{
    image1 = new GreenfootImage("bee.png");
    image2 = new GreenfootImage("bee2.png");
    imageLeft = new GreenfootImage("beeLeft.png");
    imageRight = new GreenfootImage("beeRight.png");
    setImage(image1);
    score = 0;
    lives = 3;
    isTurning = false;
} //end constructor
```

Lembre-se de que Java faz distinção entre maiúsculas e minúsculas, portanto, "BeeLeft.png" não é o mesmo que "beeLeft.png". Verifique se você está usando o tamanho correto da letra em seus arquivos de som e imagem.

## Definir a Variável isTurning

- Em seguida, em nosso método `handleMovement()` que elaboramos antes, definiremos o estado de nossa variável `isTurning` como verdadeiro se estivermos girando, caso contrário, definiremos como falso
- Também definiremos a imagem de giro adequada

```
private void handleMovement(){
    move(1);
    if(Greenfoot.isKeyDown("left")){
        turn(-2);
        isTurning = true;
        setImage(imageLeft);
    }else if(Greenfoot.isKeyDown("right")){
        turn(2);
        isTurning = true;
        setImage(imageRight);
    }else{
        isTurning = false;
    }//endif
} //end method handleMovement
```

Nesse ponto, a abelha ainda tentaria animar durante o giro.

# Operadores Lógicos

- Para testar se a Abelha está girando durante a animação:
- Várias expressões booleanas para expressar se uma ou ambas são verdadeiras ou falsas
- Operadores lógicos para conectar as expressões booleanas
- Por exemplo, a instrução "Quando a Abelha não estiver girando e a imagem atual for 1..." seria codificada como:

```
if (getImage() == image1 && !isTurning)
```



**ORACLE**  
Academy

JF 3-10  
Loops, Variáveis e Matrizes

Copyright © 2022, Oracle e/ou suas empresas afiliadas. Oracle, Java e MySQL são marcas comerciais registradas da Oracle Corporation e/ou de suas empresas afiliadas. Outros nomes podem ser marcas comerciais de seus respectivos proprietários.

23

&& (significa "e") somente retornará verdadeiro se as instruções nos dois lados do && forem verdadeiras.

# Tipos de Operadores Lógicos



É possível usar operadores lógicos para combinar várias expressões booleanas em uma

| Operador Lógico         | Significa  | Definição   |
|-------------------------|------------|---|
| Ponto de exclamação (!) | <b>NOT</b> | Reverte o valor da expressão booleana (se b for verdadeiro, !b será falso. Se b for falso, !b será verdadeiro).                         |
| Dois &                  | <b>AND</b> | Combina dois valores booleanos e retorna um valor booleano que será verdadeiro se os dois operandos forem verdadeiros.                  |
| Duas linhas (  )        | <b>OR</b>  | Combina duas variáveis ou expressões booleanas e retorna um resultado que será verdadeiro se um ou os dois operandos forem verdadeiros. |

Você pode criar tabelas de valores verdadeiros para AND /NOT/OR. Para isso, siga este guia abaixo.

Para AND/OR, você pode usar duas entradas de verdadeiro e falso e mostrar o resultado.

ENTÃO

VERDADEIRO VERDADEIRO

VERDADEIRO FALSO

FALSO VERDADEIRO

FALSO FALSO

São as quatro entradas possíveis. Para AND/OR mostrar o valor booleano resultante.



## Exemplo de Operadores Lógicos

- Os operadores lógicos definem a imagem que aparecerá se a Abelha estiver girando ou não

```
private void animateBee(){  
    if(getImage()==image1 && !isTurning)  
        setImage(image2);  
    else if (!isTurning)  
        setImage(image1);  
    //endif  
} //end method animateBee
```

Lembre-se de que && representa "e" e retornará verdadeiro se os dois lados forem verdadeiros.

# Matrizes

- Quando criamos nossa animação básica, usamos duas imagens para simular o bater das asas



- A maioria das animações terá várias imagens, mas isso aumentaria o número de condições para testar em nosso código

Quanto mais imagens, mais suave será a animação.

# Matrizes

- Com apenas quatro imagens, nosso código terá a seguinte aparência:

```
private void animateBee(){
    if(getImage()==image1 && !isTurning)
        setImage(image2);
    else if (getImage()==image2 && !isTurning)
        setImage(image3);
    else if (getImage()==image3 && !isTurning)
        setImage(image4);
    else if (!isTurning)
        setImage(image1);
    //endif
} //end method animateBee
```

- Poderíamos facilmente ter oito ou mais imagens!

Lembre-se que, em uma instrução if-else, somente uma das seções de código pode ser executada.

# Matrizes

- Usando uma matriz, você pode manter e acessar diversas variáveis a partir de somente uma

Matriz é um objeto que mantém diversas variáveis. É possível usar um índice para acessar as variáveis



À medida que desenvolver suas habilidades e conhecimento em Java, você perceberá que há estruturas mais avançadas para armazenar diversas variáveis em uma só variável.

## Como as Variáveis Mantém os Valores

- Uma variável de string simples denominada "studentname" é um contêiner que mantém um valor:
  - o nome de um estudante

```
String studentname;
```




- exemplo de contêiner studentname: **Joe**

Strings mantêm qualquer dado alfanumérico. Então, também podemos armazenar "Joe!1"

# Como as Matrizes Mantêm as Variáveis

- Um objeto de matriz pode manter muitas variáveis
- Essa matriz denominada studentnames pode manter muitas variáveis

```
String[] studentname;
```



| String[ ] |          |        |        |
|-----------|----------|--------|--------|
| 0         | 1        | 2      | 3      |
| "Joe"     | "Debbie" | "Erma" | "Besa" |



Para definir uma matriz, usamos colchetes. O tipo antes dos colchetes é o tipo que cada célula da matriz pode armazenar. No caso acima `String[]` configura uma matriz que armazena Strings.

## Declaração de Variável para uma Matriz

- Para declarar um objeto de matriz, escreva a declaração da variável da seguinte forma:
  - Tipo de elemento:
    - String [ ] para uma matriz de Strings
    - int [ ] para uma matriz de valores inteiros
  - Colchetes [ ] para indicar que essa variável é uma matriz
  - Atribuição de variável
  - A expressão que cria o objeto de matriz e a preenche com várias strings ou números inteiros

Antes de usarmos uma matriz, precisaremos configurar o número máximo de células que estarão disponíveis.

## Exemplo de Matriz

- Nesta matriz:
  - A variável de string studentnames é criada
  - "Joe", "Debbie", "Ermal" e "Besa" são objetos de string que formam o objeto de matriz
  - Um objeto de matriz é atribuído à variável studentnames

```
String [] studentnames;  
studentnames = {"Joe", "Debbie", "Ermal", "Besa"};
```



Esse é um exemplo de inicialização de uma variável de matriz. Poderíamos ter escrito

```
String[] studentnames = new String[4];  
studentnames[0] = "Joe";  
studentnames[1] = "Debbie";  
studentnames[2] = "Ermal";  
studentnames[3] = "Besa";
```

Observe que o primeiro elemento de uma matriz se inicia no índice 0.



# Acessando Elementos em uma Matriz

- Use um índice para acessar os elementos no objeto de matriz
- Para usar um índice:
  - Cada elemento tem um índice, começando na posição zero [0]
  - A posição de cada elemento aumenta em 1

Os elementos são acessados usando colchetes [ ] e um índice para especificar qual elemento da matriz será acessado. Índice é um número de posição no objeto de matriz



| String[ ] |       |          |         |        |
|-----------|-------|----------|---------|--------|
| Index     | 0     | 1        | 2       | 3      |
| Element   | "Joe" | "Debbie" | "Ermal" | "Besa" |

Você gerará um erro se tentar acessar um elemento da matriz que estiver fora do intervalo. Neste exemplo, o intervalo de elementos corresponde aos valores de 0 a 3.

## Acessar Elementos em uma Matriz


- Para acessar um elemento da matriz, anexe ao nome da matriz o índice do elemento em questão
- A instrução `studentnames[3]` acessa o elemento da matriz no índice 3 — a string "Besa"
- Esse é o quarto elemento da matriz

| String[ ] |       |          |         |        |
|-----------|-------|----------|---------|--------|
| Index     | 0     | 1        | 2       | 3      |
| Element   | "Joe" | "Debbie" | "Ermal" | "Besa" |

Lembre-se que essa matriz é declarada como uma matriz de string. Se você retornar um elemento da matriz, esse valor será uma string

## Animação da Matriz

- Para que uma instância anime melhor várias imagens, podemos declarar uma matriz
- A matriz inclui todas as imagens para animar a Abelha
- Declare um campo na classe Abelha para a matriz
- Itere na matriz

| 0   | 1   | 2   | 3   |
|---|---|---|---|
|  |  |  |  |

Isso significa que podemos armazenar todas as nossas imagens em uma variável.

## Criar as Matrizes

- Crie uma matriz na classe Abelha para armazenar quatro imagens
- Crie uma variável de valor inteiro para armazenar o índice de imagem atual

```
public class Bee extends Actor
{
    private GreenfootImage[] images = new GreenfootImage[4];
    private int currentImage;

    boolean isTurning;
    private int score;
    private int lives;

    /**
     * Bee - sets the initial values of the bee
     */
    public Bee()
    {
```

Podemos declarar uma constante para o máximo de imagens em vez de usar o valor 4. Isso facilitaria a leitura do código e seria muito mais fácil alterá-lo no futuro.

Se tivéssemos a constante MAXIMAGES = 4;




Poderíamos alterar o código para

..... = new GreenfootImage[MAXIMAGES];

## Criar as Matrizes

- Inicialize a nova imagem, a matriz de imagens e as variáveis currentimage no construtor

```
public Bee()  
{  
    images[0] = new GreenfootImage("bee1.png");  
    images[1] = new GreenfootImage("bee2.png");  
    images[2] = new GreenfootImage("bee3.png");  
    images[3] = new GreenfootImage("bee4.png");  
    currentImage = 0;  
    setImage(image1);  
    score = 0;  
    lives = 3;  
    isTurning = false;  
} //end constructor
```

| 0   | 1   | 2   | 3   |
|---|---|---|---|
|  |  |  |  |

**ORACLE**  
Academy

JF 3-10  
Loops, Variáveis e Matrizes

Copyright © 2022, Oracle e/ou suas empresas afiliadas. Oracle, Java e MySQL são marcas comerciais registradas da Oracle Corporation e/ou de suas empresas afiliadas. Outros nomes podem ser marcas comerciais de seus respectivos proprietários.

37

A imagem 2 é usada duas vezes, pois queremos mostrar as imagens – bee -> bee1 -> bee2 -> bee1 e depois repetir.

## Criar as Matrizes

- Podemos usar nosso conhecimento sobre loop while para codificar melhor o slide anterior

```
public Bee()  
{  
    images[0] = new GreenfootImage("bee1.png");  
    images[1] = new GreenfootImage("bee2.png");  
    images[2] = new GreenfootImage("bee3.png");  
    images[3] = new GreenfootImage("bee4.png");  
    currentImage = 0;  
}
```

- Pode ser escrito como

```
public Bee(){  
    int i = 0;  
    while(i<4){  
        images[i] = new GreenfootImage("bee" + (i+1) + ".png");  
        i = i+1;  
    } //endwhile  
    currentImage = 0;  
}
```

Na segunda solução seria mais fácil adicionar outras imagens. A ("bee" + i + ".png"); é conhecida como concatenação de strings. Ela une as strings. Então, se i=1, obteríamos "bee1.png". No exemplo do loop while, copiaríamos a bee1.png e a chamaríamos de bee3.png.

currentImage seria um campo de classe.

# Criar as Matrizes

- Modificar o método animateBee()

```
private void animateBee(){  
    if(currentImage == 3)  
        currentImage = 0;  
    else  
        currentImage++;  
    //endif  
    setImage(images[currentImage]);  
} //end method animateBee
```



**ORACLE**  
Academy



JF 3-10  
Loops, Variáveis e Matrizes

Copyright © 2022, Oracle e/ou suas empresas afiliadas. Oracle, Java e MySQL são marcas comerciais registradas da Oracle Corporation e/ou de suas empresas afiliadas. Outros nomes podem ser marcas comerciais de seus respectivos proprietários.

39

Observe que agora o método animateBee() lidaria com qualquer quantidade de imagens. Só precisaríamos alterar o valor de índice 3 para um novo valor. Nesse caso, a melhor substituição para o valor 3 seria uma constante.

if (currentimage == MAX\_IMAGES-1)...

MAX\_IMAGES seria declarado como um campo de classe

```
private final int MAX_IMAGES = 4;
```

Em seguida, atualizaríamos nosso código para usar essa constante em vez de 3. Isso facilita a leitura e a manutenção do código.

# Terminologia

- Os principais termos usados nesta lição foram:
  - Matriz
  - Elementos
  - Índice
  - Loop infinito
  - Variáveis de local
  - Operadores lógicos
  - Loop



# Resumo

- Nesta lição, você deverá ter aprendido a:
  - Criar um loop while em um construtor para criar um mundo
  - Descrever um loop infinito e como impedir sua ocorrência
  - Usar uma matriz para armazenar diversas variáveis
  - Criar uma expressão com operadores lógicos
  - Descrever o escopo de uma variável local em um método



