

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

Java Fundamentals

3-8

Mundo, Animação e Fim do Jogo

ORACLE
Academy

```
private void caughtBySpider() {  
    if (isTouching(Spider.class)) {  
        setLocation(20,20);  
        lives--;  
        if (lives < 0) {  
            Greenfoot.stop();  
        }  
    }  
}
```

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.

Objetivos

- Esta lição abrange os seguintes objetivos:
 - Construir um objeto mundo usando um método construtor
 - Criar um objeto usando um construtor
 - Escrever instruções de programação para usar a nova palavra-chave
 - Definir a finalidade e a sintaxe de uma variável
 - Reconhecer a sintaxe para definir e testar variáveis
 - Escrever instruções de programação para alternar entre duas imagens
 - Escrever instruções de programação para terminar um jogo



Construtores

- Quando uma nova subclasse Mundo é criada e compilada, o Greenfoot executa um construtor que cria uma instância dela para ser exibida no cenário
- Os construtores configuram uma instância e definem um estado inicial, como o tamanho e a resolução da instância
 - Os construtores não têm um tipo de retorno
 - O nome deles, logo após a palavra "public", é o mesmo que o nome da classe em que eles estão definidos

Os construtores são métodos especiais que são executados automaticamente sempre que uma nova instância da classe é criada



Como vimos anteriormente declaramos um construtor como
`public <ClassName>(optional parameters)`

Parâmetros do Construtor

- Os parâmetros de um construtor permitem que valores iniciais de uma instância sejam passados para o construtor
- Esses parâmetros:
 - Só estão disponíveis para a instância criada pelo construtor.
 - Têm um escopo restrito limitado ao momento em que o construtor é declarado
 - Têm um ciclo de vida limitado a uma única execução do construtor
 - Desaparecem quando termina a execução de um construtor
 - São variáveis válidas desde que a instância exista

No Greenfoot, o construtor na subclasse mundo é adicionado por padrão. As subclasses do ator não têm um construtor adicionado ao código.

Construtor - Exemplo

- Este construtor na subclasse Mundo usa a palavra-chave `super()` para passar os valores de altura, largura e resolução do mundo para a instância

```
public class BeeWorld extends World
{
    /**
     * Constructor for objects of class BeeWorld.
     *
     */
    public BeeWorld()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(600, 400, 1);
    }
}
```

"super" no Java é uma chamada para o construtor pai. Em BeeWorld, super é adicionado com três parâmetros definindo a largura, a altura e o tamanho da célula.

Isso informa-nos que a classe Mundo tem um construtor o qual aceita três valores inteiros.

Parâmetros - Exemplo

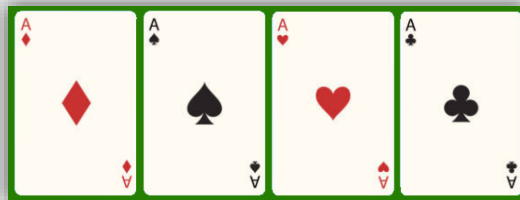
- Para alterar o tamanho do tabuleiro do jogo, modifique os argumentos no parâmetro do construtor
- Este exemplo torna o mundo quadrado, em vez de retangular, alterando o limite da coordenada x para 400

```
public class BeeWorld extends World
{
    /**
     * Constructor for objects of class BeeWorld.
     *
     */
    public BeeWorld()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(600, 400, 1);
    }
}
```

Quando estiver projetando seu jogo, sempre lembre-se de que talvez outros usuários não tenham um tamanho de tela que corresponda à sua resolução. Então, se você criar um jogo com dimensões de 1900 x 1200, muitos usuários não conseguirão exibir tudo isso.

Criação Automática de Instâncias Ator

- Escreva o código no construtor Mundo para adicionar automaticamente instâncias Ator ao jogo quando o cenário for inicializado
- Assim, o jogador não precisará adicionar instâncias manualmente antes de o jogo começar
- Por exemplo, em um jogo de correspondência, as cartas deverão ser exibidas automaticamente no cenário quando o jogo começar



ORACLE
Academy

JF 3-8
Mundo, Animação e Fim do Jogo

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.

8

Já vimos que é exatamente isso que a opção "Save The World" faz, mas podemos modificar essa ação no código-fonte para ampliar seu recurso.

Código para Criar Instâncias Automaticamente

- O código no construtor Mundo inclui os seguintes componentes:
- Instrução `super()` com o tamanho do mundo como argumentos
- Método `addObject()` com os seguintes argumentos:
 - A palavra-chave `new`, seguida do nome da classe, informa ao construtor que uma nova instância dessa classe deve ser adicionada
 - As coordenadas X e Y em que a nova instância deverá ser posicionada no mundo

```
public BeeWorld()  
{  
    // Create a new world with 600x400 cells with a cell size of 1x1 pixels.  
    super(600, 400, 1);  
    addObject (new Bee(), 150, 100);  
}
```

`Super(560,560,1)` – cria um mundo com a largura 560 e a altura 560.

Instâncias Ator do Greenfoot

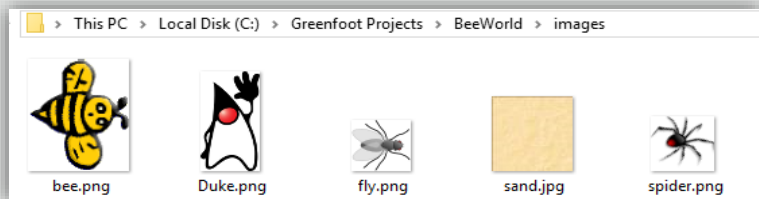
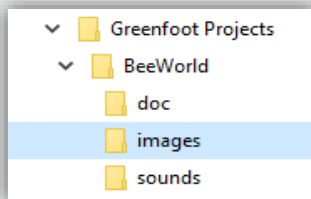
- Alternar entre duas imagens que têm uma aparência ligeiramente diferente proporciona a uma instância a aparência de movimento
- Instâncias Ator do Greenfoot:
 - Recebem e armazenam uma imagem da respectiva classe
 - A imagem foi atribuída à classe quando a classe foi criada
 - Têm a capacidade de armazenar várias imagens
 - Podem ser programadas para alterar a qualquer momento a imagem que exibem



Não precisamos nos limitarmos a apenas duas imagens. Quanto mais imagens usarmos na animação, mais suave será o efeito.

Classe GreenfootImage

- A classe GreenfootImage permite que atores do Greenfoot mantenham sua imagem visível armazenando um objeto do tipo GreenfootImage
- Esta classe é usada para ajudar uma classe a obter e manipular tipos diferentes de imagens
- É necessário que as imagens que essa classe usará pré-existam na pasta Images do cenário



Lembre-se de que imagens transparentes permitem criar efeitos mais realistas, em vez das imagens que têm aparência de um bloco.

Construtor para Obter um Novo Objeto de Imagem

- Crie um construtor que recupere um novo objeto de imagem de um arquivo ao criar uma instância de uma classe
- O construtor de exemplo abaixo cria a nova imagem e anexa-a à classe Ator

```
/**
 * Bee - sets the initial values of the bee
 */
public Bee()
{
    setImage(new GreenfootImage("bee.png"));
} //end constructor
```

Nome do Arquivo de Imagem como Argumentos na Lista de Parâmetros

Palavra-chave new

Método setImage

Classe GreenfootImage

A palavra-chave new gerará uma chamada para o construtor de GreenfootImage. Neste exemplo, o construtor aceita o valor de uma string que está relacionada ao nome de uma imagem armazenada na pasta Images

Atribuindo uma Nova Imagem a uma Classe

- A instrução abaixo cria o objeto da nova imagem com base no arquivo de imagem nomeado
- Quando este objeto de imagem é inserido no código-fonte da classe, ele está pronto para ser usado pela classe
- A instrução é executada da seguinte maneira:
 - Primeiro é criado o objeto `GreenfootImage`
 - A chamada do método `setImage()` é executada, passando o objeto de imagem recém-criado para a lista de parâmetros

```
public Bee()  
{  
    setImage(new GreenfootImage("bee.png"));  
} //end constructor
```

Lembre-se de que o Java faz distinção entre letras maiúsculas e minúsculas. Por isso, o uso de maiúsculas e minúsculas é importante na string "bee.png".

Atribuindo um Exemplo de Imagem

- O método setImage() atribui a imagem no arquivo "bee.png" à classe Ator
- Cada vez que uma instância dessa classe é adicionada ao cenário, ela exibe a imagem "bee.png"

```
/**
 * Bee - sets the image of the bee
 */
public Bee()
{
    setImage(new GreenfootImage("bee.png"));
} //end constructor
```

Cria a nova imagem.

Nome do arquivo de imagem como argumento.

Permite que um objeto de imagem seja usado pela classe Ator. Espera uma imagem como parâmetro.

Imagem de uma classe Greenfoot.

Por que Instâncias Armazenam Várias Imagens

- Talvez você queira que uma instância armazene e acesse várias imagens:
 - Para simular mudança de cor
 - Para simular mudança de um tipo de objeto para outro Por exemplo, mude de um coelho para uma tartaruga como em um passe de magia
 - Para simular movimento:
 - Andar:
 - mude de um objeto com a perna esquerda estendida para um objeto com a perna direita estendida
 - Virar cartas do baralho:
 - mude de uma carta em branco para uma que não esteja em branco
 - Voar:
 - mude de asas estendidas para asas dobradas

A parte mais difícil de uma animação no Greenfoot ou em qualquer outra ferramenta de software é desenhar as imagens ou obter acesso a imagens relevantes na internet.

Acessando Várias Imagens

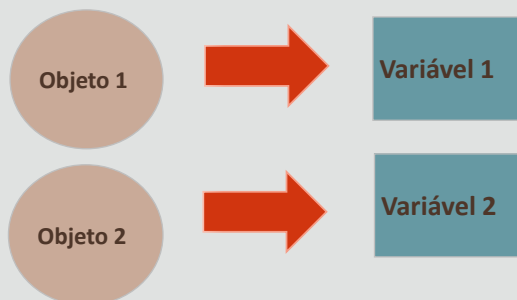
- Por exemplo, uma instância poderia acessar duas imagens, cada uma com uma pequena diferença na posição da asa
- Assim, a instância pareceria bater as asas enquanto se move
- Para produzir esse movimento:
 - Crie duas imagens da instância, cada uma com uma posição levemente diferente da asa
 - Armazene as duas imagens na instância para que elas possam ser acessadas repetidamente enquanto o objeto se mover
 - Codifique a classe para alternar entre as duas imagens que são exibidas



Variáveis

- Use variáveis de classe para armazenar os dois objetos de imagem
- Assim, a classe pode acessá-los facilmente para serem usados dentro das instâncias

Uma variável é declarada em uma classe. Ela é utilizada para armazenar informações para uso posterior ou para transmitir informações. Ela pode armazenar objetos ou valores.



Só criamos variáveis do tipo int ou booleano. Para armazenar uma imagem, use o tipo GreenfootImage.

Formato da Variável

- O formato de uma variável inclui o seguinte:
 - Tipo de dados:
 - o tipo de dados que será armazenado na variável
 - Nome da variável:
 - uma descrição da finalidade da variável para que seja possível fazer referência a ela mais tarde

```
private variable-type variable-name;
```

- Neste exemplo, o nome da variável é `image1` e o tipo da variável é `GreenfootImage`

```
private GreenfootImage image1;
```


Normalmente declaramos variáveis de classe como privadas para que elas não possam ser acessadas diretamente fora da classe. Criamos métodos que permitam obter e definir o respectivo valor.

Declarando Variáveis

- Declare variáveis antes dos construtores e métodos
- O formato para declarar uma variável inclui o seguinte:
 - A palavra-chave "private" para indicar que a variável só está disponível dentro da classe Ator
 - A classe a que a imagem pertence
 - Um espaço reservado para a variável em que a imagem será armazenada

```
public class Bee extends Actor
{
    private GreenfootImage image1;
    private GreenfootImage image2;

    /**
     * Bee - sets the initial values of the bee
    */
}
```



Podemos ver que adicionamos dois campos de classe adicionais, ambos do tipo GreenfootImage. Se clicássemos em uma instância Abelha e seleccionássemos inspect, veríamos esses dois campos.

Instruções de Atribuição

- É necessária uma atribuição para armazenar objetos em uma variável
- Quando um objeto é atribuído a uma variável, a variável contém uma referência a esse objeto
- Uma instrução de atribuição:
 - Armazena o objeto ou o valor em uma variável
 - É escrita com o símbolo de igualdade
- Formato:



variável = expressão;

ORACLE
Academy

JF 3-8
Mundo, Animação e Fim do Jogo

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.

20

Lembre-se de que um único sinal de igual = corresponde uma atribuição e dois sinais de igual == correspondem a uma comparação.

Componentes da Instrução de Atribuição

- Uma instrução de atribuição inclui o seguinte:
 - Variável: o nome da variável que armazenará o objeto ou um valor
 - Símbolo de igualdade, que é o símbolo de atribuição
 - Expressão:
 - O nome do objeto ou do valor a ser atribuído
 - Uma instrução que informa que o objeto ou o valor é novo
 - A classe a que a imagem pertence

- Exemplo:

```
public Bee()  
{  
    image1 = new GreenfootImage("bee.png");  
    image2 = new GreenfootImage("bee2.png");  
} //end constructor
```

Lembre-se de que todas as nossas imagens devem estar na pasta Images.

Inicializando Imagens ou Valores

- A inicialização é o processo de definir a instância e seus valores iniciais
- Quando a classe cria novas instâncias, cada instância contém uma referência às imagens ou aos valores contidos nas variáveis
- Diretrizes:
 - A assinatura não inclui um tipo de retorno
 - O nome do construtor é o mesmo que o nome da classe
 - O construtor é executado automaticamente para passar a imagem ou o valor à instância quando uma instância da classe é criada

Construtores de Atores - Exemplo

- O construtor de ator a seguir informa ao Greenfoot para criar automaticamente uma nova instância Abelha e inicializar, ou atribuir, duas variáveis para a instância
- A última linha do construtor, setImage(image1), indica que a primeira variável deverá ser exibida quando a instância for adicionada ao cenário



ORACLE
Academy

JF 3-8
Mundo, Animação e Fim do Jogo

```
private GreenfootImage image1;  
private GreenfootImage image2;  
  
/**  
 * Bee - sets the initial values of the bee  
 */  
public Bee()  
{  
    image1 = new GreenfootImage("bee.png");  
    image2 = new GreenfootImage("bee2.png");  
    setImage(image1);  
} //end constructor
```

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

Usamos duas imagens aqui, mas poderíamos usar quantas imagens desejássemos.

Testar Valores de Variáveis

- Depois que a classe tiver inicializado as duas variáveis com as imagens, programe a instância para alternar automaticamente a imagem que é exibida conforme ela se move
- À medida que essas imagens alternam a cada movimento, isso faz com que a instância pareça mais animada
- É possível programar a alternância entre as imagens sem precisar escrever muitas linhas do código que associa cada imagem a cada movimento



Escrever Ações em um Pseudocódigo

- Identifique as ações a serem programadas escrevendo-as em um pseudocódigo
- O pseudocódigo expressa as tarefas ou as operações das instâncias a serem executadas em uma combinação de linguagem Java e palavras em inglês simples
- Isso ajuda a compreender melhor quais comportamentos as instâncias devem executar antes de escrever o código propriamente dito



ORACLE
Academy

JF 3-8
Mundo, Animação e Fim do Jogo

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.

25

O pseudocódigo permite que você foque no algoritmo, e não na sintaxe.

Exemplo de Pseudocódigo

- A image1 é exibida quando a instância é criada
- Quando a Abelha fizer seu próximo movimento, image2 deverá ser exibida, e vice-versa
- Isso é expresso como uma instrução if-else

```
if (current image displayed is image1) then
    use image2 now
else
    use image1 now
```



Ter uma boa ideia do pseudocódigo facilita bastante a criação de instruções Java equivalentes. É fácil ficar muito focado na sintaxe, em vez de pensar no algoritmo.

Operador '=='

- As instruções de programação que instruem a instância a alternar entre imagens contêm o seguinte:
 - instrução if-else
 - o operador '==' (dois sinais de igual)
- O operador '==':
 - É usado em uma instrução if para testar se dois valores são iguais
 - Compara um valor com outro
 - Retorna um resultado booleano (verdadeiro ou falso)
- Lembre-se de que '=' é o símbolo de atribuição, e não o símbolo para testar se dois valores são iguais

O compilador java reportará quase sempre um erro se você combinar um sinal de igual com dois sinais de igual.

Componentes da Instrução if-else

- Componentes da Instrução if-else:
 - O método `getImage` recebe a imagem atual da instância
 - O operador `'=='` verifica se o valor que a instância exibiu é igual a `image1`
 - Se for igual, ele exibirá `image2`
 - caso contrário, exibirá `image1`



ORACLE
Academy

JF 3-8
Mundo, Animação e Fim do Jogo

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.

28

`getImage` é um método da classe `Ator`.

Exemplo da Instrução if-else

- A instrução if-else abaixo é escrita no método act para fazer a instância alternar a exibição de duas imagens quando ela mover-se para frente

```
/**
 * Act - do whatever the Bee wants to do. This method is called whenever
 * the 'Act' or 'Run' button gets pressed in the environment.
 */
public void act()
{
    if(getImage()==image1)
        setImage(image2);
    else
        setImage(image1);
    //endif
    handleMovement();
    catchFly();
    turnAtEdge();
} //end method act
```

Note que temos dois métodos que criamos anteriormente e estão listados na parte inferior do método act. Movemos esses métodos para seus próprios métodos para tornar o código mais legível.

Exemplo da Instrução if-else

- Moveremos também o código da animação para seu próprio método, a fim de manter o código mais limpo

```
public void act()
{
    animateBee();
    handleMovement();
    catchFly();
    turnAtEdge();
} //end method act

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

É provável que a execução do programa produza uma animação que seja muito rápida. Podemos adicionar um atraso usando um contador que só mudará a animação quando o contador for redefinido.

Terminar o Jogo

- A classe Greenfoot tem um método stop() que pode ser usado para terminar o jogo no momento que você indicar
- Talvez você queira terminar o jogo quando:
 - O jogador atingir uma meta
 - O tempo se esgotar no relógio
 - A instância atingir determinada coordenada ou objeto



Pode ser também que você não deseje parar o jogo do Greenfoot; apenas queira retornar à tela inicial.

Exemplo do Jogo da Abelha

- Exemplo do jogo:
 - O jogador decide quantas vezes a Abelha deve ser capturada pelo objeto Aranha para terminar o jogo
 - Quando o jogo termina, é emitido o som "Game Over" (Fim do jogo)
- Especificações do jogo:
 - Criar e inicializar variáveis para armazenar vidas e pontuação
 - Fornecer uma contagem do total de Moscas comidas (pontuação)
 - Informe o método stop() para interromper o jogo quando o número de vidas do jogador atingir o valor 0

ORACLE
Academy



JF 3-8
Mundo, Animação e Fim do Jogo



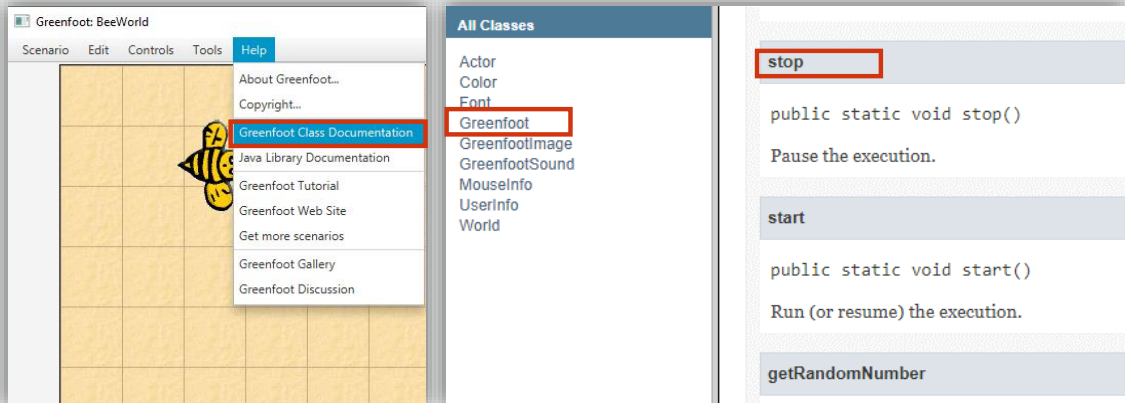
Copyright © 2022, Oracle e/ou suas empresas afiliadas. Oracle, Java e MySQL são marcas comerciais registradas da Oracle Corporation e/ou de suas empresas afiliadas. Outros nomes podem ser marcas comerciais de seus respectivos proprietários.

32

Você pode gravar seu próprio arquivo de som .wav para o fim do jogo.

Localizar o Método stop() na API do Greenfoot

- Vá até o menu Help e selecione a Documentação da Classe do Greenfoot
- Localize o método stop() na API do Greenfoot



Podemos chamar o método stop() de uma classe Ator ou de uma classe Mundo.

Escrever o Método stop() no Código-Fonte

- No ponto em que o jogo deve terminar, escreva o método da seguinte forma no código-fonte
- A notação de pontos é usada para chamar o método

```
private void endGame(){  
    Greenfoot.stop();  
} //end method endGame
```



Atribuir Variáveis a Instâncias - Exemplo

- A Abelha deve capturar um número de objetos Mosca para aumentar a pontuação
- A Abelha também perderá uma vida se for pega pela aranha
- As variáveis são definidas antes dos construtores e dos métodos
- O construtor Abelha atribui as variáveis às instâncias que ele produz



```
public class Bee extends Actor
{
    private GreenfootImage image1;
    private GreenfootImage image2;
    private int score;
    private int lives;

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

É uma boa prática também adicionar comentários às variáveis das classes porque, assim, o leitor compreende melhor o significado dessas variáveis. Neste exemplo, elas foram omitidas devido ao espaço do screenshot.

Método catchfly() Definido - Exemplo

- O método catchFly() definido é escrito abaixo no método act(), a fim de orientar a Abelha a capturar objetos mosca
- Vamos somar uma unidade à variável de pontuação para cada Mosca que for comida

```
/**
 * catchFly - if the Bee touches a fly the fly is removed
 * A sound is played and a new fly is added to the game
 */
private void catchFly(){
    if(isTouching(Fly.class)){
        removeTouching(Fly.class);
        Greenfoot.playSound("slurp.wav");
        score++;
        getWorld().addObject(new Fly(), Greenfoot.getRandomNumber(getWorld().getWidth()),
                                Greenfoot.getRandomNumber(getWorld().getHeight()));
    } //endif
}
```

O código score++; produz o mesmo resultado que escrever score = score+1; que soma uma unidade à pontuação.

A última linha do código obtém o mundo atual e chama o respectivo método addObject() que vimos anteriormente. Desta vez, vamos chamá-lo em um ator para que getWorld() retorne uma referência ao mundo atual. Isso significa que, cada vez que uma Mosca for pega, outra reaparecerá.

Atribuir Variáveis a Instâncias - Exemplo 2

- Se a Abelha tocar na aranha, ela deverá perder uma vida
- Vamos também reposicionar a Abelha no lado superior esquerdo
- Estenderemos a classe Abelha adicionando um novo método
 - caughtBySpider() e uma chamada a ele no método act()
- Em seguida, testaremos se o usuário não tem mais nenhuma vida e pararemos o jogo

```
private void caughtBySpider(){
    if(isTouching(Spider.class)){
        setLocation(20,20);
        lives--;
        if(lives<0){
            endGame();
        }//endif
    }//endif
} //end method caughtBySpider

private void endGame(){
    Greenfoot.stop();
} //end method endGame
```

Podemos então escrever o seguinte:

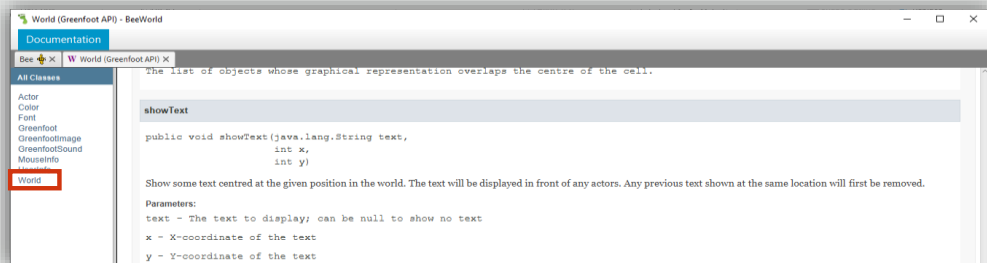
```
this.setLocation(20,20);
```

```
this.lives--;
```

Lembre-se de que esse "this" é opcional e, na maioria dos casos, não é obrigatório.

Mostrando Texto

- Às vezes, queremos que o usuário de um aplicativo esteja informado sobre determinados aspectos de sua interação, como vidas, pontuações ou cartas restantes
- Neste caso também o Greenfoot oferece várias maneiras de fazer isso
- A maneira mais simples é usando o método `World – showText()`



O método `Greenfoot.showText()` foi adicionado na versão 2.4.0

Atualizar o Método catchfly()

- Vamos adicionar um código ao método catchFly() para incrementar o campo de pontuação e, em seguida, exibir o resultado na tela
- Também movemos a pontuação de atualização para seu próprio método e a chamamos dentro de catchFly()

```
private void catchFly(){
    if(isTouching(Fly.class)){
        removeTouching(Fly.class);
        Greenfoot.playSound("slurp.wav");
        updateScore();
        getWorld().addObject(new Fly(), Greenfoot.getRandomNumber(getWorld().getWidth()),
                                Greenfoot.getRandomNumber(getWorld().getHeight()));
    } //endif
}

private void updateScore(){
    score++;
    getWorld().showText("Score : " + score, 60, 390);
} //end method updateScore
```

A criação de um novo método denominado updateScore() permite chamar a pontuação de qualquer outra parte na classe Abelha. Isso significa que, se adicionássemos algum outro elemento que aumentasse a pontuação, poderíamos simplesmente chamar updateScore() sem precisar escrever novamente o código. Também poderíamos adicionar um parâmetro como int scoreincrease que aumentaria a pontuação em mais de uma unidade –

```
updateScore(int scoreincrease) {
    score = score + scoreincrease;
}
```

Terminologia

- Estes são os principais termos usados nesta lição:
 - Construtor
 - Variável definida
 - Pseudocódigo

Resumo

- Nesta lição, você deverá ter aprendido a:
 - Construir um objeto mundo usando um método construtor
 - Criar um objeto usando um construtor
 - Escrever instruções de programação para usar a nova palavra-chave
 - Definir a finalidade e a sintaxe de uma variável
 - Reconhecer a sintaxe para definir e testar variáveis
 - Escrever instruções de programação para alternar entre duas imagens
 - Escrever instruções de programação para terminar um jogo



