

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

3-3

## Código-Fonte e Documentação

**ORACLE**  
Academy

```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Bee here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Bee extends Actor
{
    /**
     * 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()
    {
        // Add your action code here.
    }
}
```

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

# Objetivos

- Esta lição abrange os seguintes objetivos:
  - Demonstrar as alterações do código-fonte para chamar métodos de maneira programática
  - Demonstrar as alterações do código-fonte para escrever uma declaração de decisão if
  - Descrever um método para exibir a orientação de um objeto



# Código-fonte

- O código-fonte é o plano gráfico ou o mapa que define como seus objetos e seu programa funcionam
- Ele comanda os objetos no cenário para moverem e interagirem



```
Bee
Compile Undo Cut Copy Paste Find... Close Source Code
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Bee here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Bee extends Actor
{
    /**
     * 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()
    {
        // Add your action code here.
    }
}
```

ORACLE  
Academy

JF 3-3  
Código-Fonte e Documentação

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados. 4

Depois do posicionamento inicial dos atores, você gastará a maior parte do tempo trabalhando com o código-fonte.

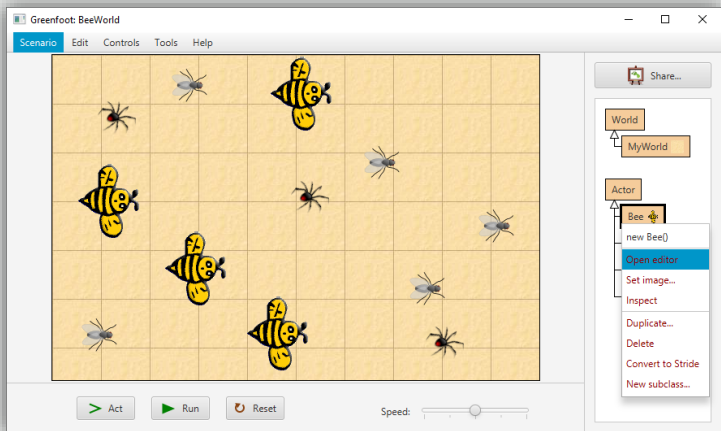
# Code Editor

- O código-fonte é gerenciado no Code editor
- Para exibir o Code editor, clique com o botão direito do mouse em qualquer classe no ambiente e selecione Open editor no menu



**ORACLE**  
Academy

JF 3-3  
Código-Fonte e Documentação



Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados. 5

Você também pode clicar duas vezes na classe. O code editor padrão será aberto em uma nova janela.

# Funções do Code Editor

- No Code editor, você pode:
  - Gravar o código-fonte para programar a ação das instâncias da classe
  - Modificar o código-fonte para alterar o comportamento de uma instância
  - Analisar as propriedades e os métodos herdados da classe
  - Analisar os métodos criados especificamente para a classe pelo programador que gravou o código-fonte

Não se preocupe se o código parecer muito complicado no momento. Logo você conseguirá entendê-lo com mais facilidade.

# Componentes do Código-Fonte

1	Descrição da Classe
2	Método act()
3	Assinatura do Método
4	Corpo do Método
5	Comentários
6	Documentação
7	Definição da Classe

**ORACLE**  
Academy

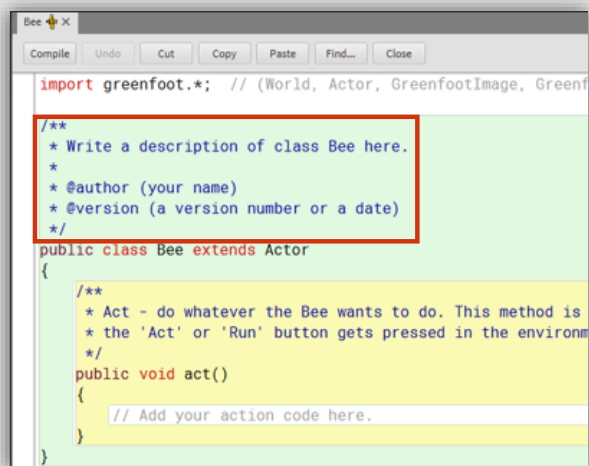
JF 3-3  
Código-Fonte e Documentação

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados. 7

Todos eles, exceto para a descrição, os comentários e a documentação, são obrigatórios para fazer com que o programa funcione. A descrição, os comentários e a documentação fazem com que seja mais fácil para você e outras pessoas entenderem o código. Tire o máximo proveito destas seções; você verá como é importante a documentação fornecida pela equipe de desenvolvimento do Greenfoot. Adquira o hábito de comentar e documentar seu próprio código.

## Descrição da Classe

- A descrição da classe é um conjunto de comentários que podem ser modificados para descrever a classe. Ela inclui o seguinte:
  - Uma descrição do que a classe faz
  - O nome do autor do código
  - A data da última modificação do código-fonte



The screenshot shows the Bee IDE interface. The code editor displays the following code:

```
import greenfoot.*; // (World, Actor, GreenfootImage, GreenfootSound, GreenfootTimer)

/**
 * Write a description of class Bee here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Bee extends Actor
{
    /**
     * Act - do whatever the Bee wants to do. This method is
     * the 'Act' or 'Run' button gets pressed in the environment
     */
    public void act()
    {
        // Add your action code here.
    }
}
```

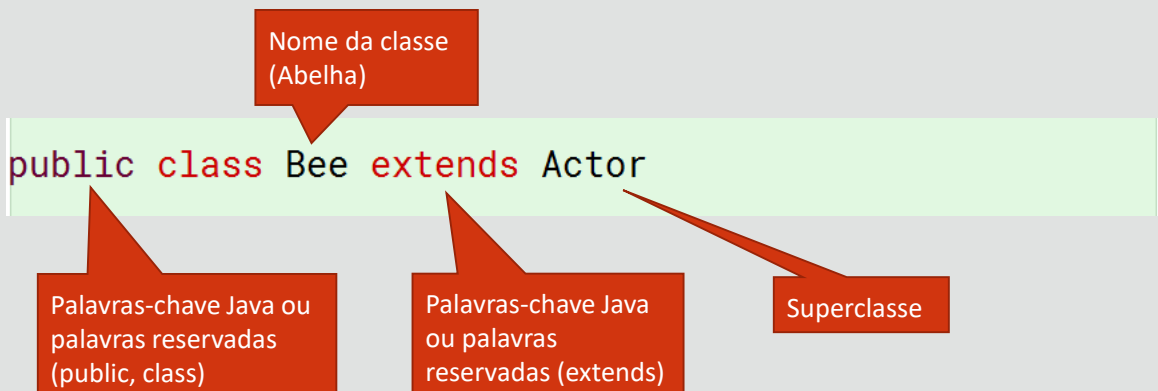
A red rectangle highlights the class description comment block (the first four lines of code).

Em um estágio anterior, analisamos a documentação gerada de forma automática. O texto desses comentários é usado para criar uma documentação mais significativa.



# Componentes da Definição da Classe

- A definição da classe inclui o seguinte:
  - Palavras-chave Java ou palavras reservadas
  - O nome da classe conforme definida pelo programador
  - O nome da superclasse da qual a subclasse estende-se

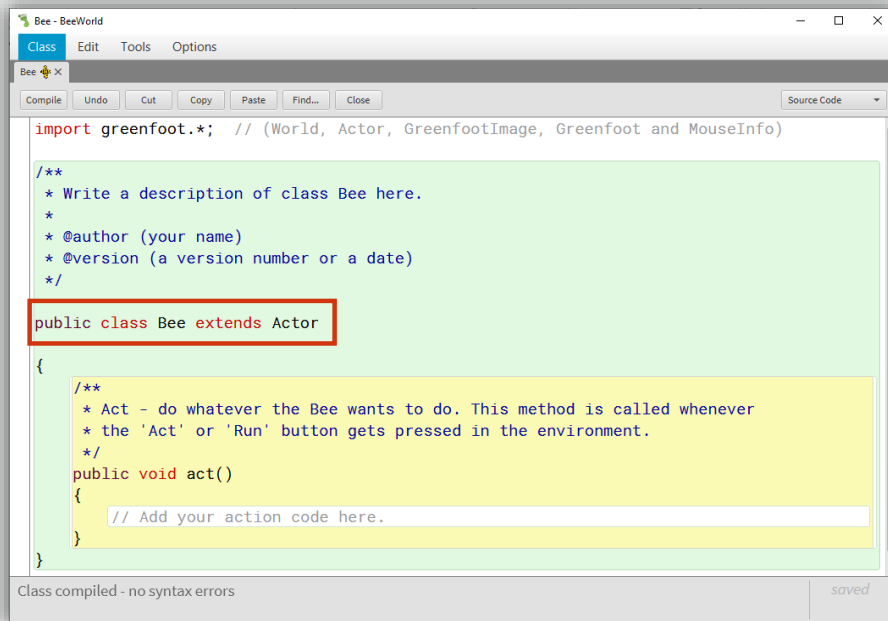


Public informa ao java o quão acessível o objeto deverá estar. Public é o menos restritivo e private é o mais restritivo. Analisaremos isso em maiores detalhes posteriormente.

Class informa ao java que estamos prestes a definir nossa própria classe.

Extends informa ao Java que queremos baseá-la em uma classe definida já existente denominada Ator. Isso significa que estamos criando uma subclasse de Ator denominada Abelha.

# Definição de Classe - Exemplo



The screenshot shows the Bee - BeeWorld IDE interface. The main window displays the source code for the Bee class. The code is as follows:

```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Bee here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */

public class Bee extends Actor
{
    /**
     * 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()
    {
        // Add your action code here.
    }
}
```

The line `public class Bee extends Actor` is highlighted with a red box. The IDE status bar at the bottom indicates "Class compiled - no syntax errors" and "saved".

Lembre-se de que é comum atribuir um nome à classe começando com uma letra maiúscula. Então, no exemplo acima, usamos Abelha, em vez de abelha.

## Método act()

- O método act() é a parte da definição de classe que informa aos objetos quais métodos devem ser executados quando clica-se nos controles de execução Act ou Run no ambiente

```
/**
 * Write a description of class Bee here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */

public class Bee extends Actor
{
    /**
     * 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()
    {
        // Add your action code here.
    }
}
```

A área vermelha realçada também inclui os comentários. Esses comentários também são usados na documentação que é gerada automaticamente. Lembre-se de que anteriormente afirmamos que o método act() faz um loop contínuo. Ou seja, ele executa o código e, em seguida, executa-o sucessivamente.

# Definindo Classes

- A definição da classe define o seguinte:
  - Variáveis (ou campos) que armazenam dados de forma persistente dentro de uma instância
  - Construtores que configuram uma instância inicialmente
  - Métodos que fornecem os comportamentos de uma instância
- Ao definir uma classe, use um formato consistente
  - Por exemplo, primeiro defina variáveis, depois construtores e, em seguida, métodos

```
public class Duke extends Actor
{
    //Create Instance variables first

    //Create Constructors next

    //Create all methods next
}
```

Os métodos enquadram-se em três variedades que exploraremos mais adiante. Eles retornam informações dos campos. Geralmente, esse é o valor ou algum cálculo no valor. Eles definem o valor armazenado nos campos informando um valor ou, de novo, por meio de algum cálculo. Outra possibilidade seria eles fornecerem alguma funcionalidade.

# Assinatura do Método

- A assinatura do método descreve o que o método faz
- Ela contém o nome de um método e uma lista de parâmetros

```
/**
 * Write a description of class Bee here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */

public class Bee extends Actor
{
    /**
     * 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()
    {
        // Add your action code here.
    }
}
```

Nome do método

Lista de parâmetros ()

Assim como uma classe, a definição do método também tem acessibilidade. Esses são os mesmos valores que são usados para definir a acessibilidade da classe. "public" é o mais acessível.

O método act não retorna nada; o termo void é usado para denotar esse fato. Se você não adicionar void ou um tipo de retorno, o java reportará isso como um erro de sintaxe.

## Comentários

- Os comentários descrevem o que o código-fonte faz.
  - Eles não prejudicam a funcionalidade do programa
  - Começam com uma barra e dois asteriscos ( `/**` ) ou apenas uma barra dupla
  - Terminam com `*/` comments with `*/`
  - São escritos em fonte azul (no Greenfoot)

```
/**
 * 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()
{
    // Add your action code here.
}
```

O Java ignora os comentários. Eles são usados para compreendermos melhor o código e seu significado. Lembre-se de que você pode estar trabalhando com um código que foi gravado por outra pessoa. Você agradecerá por ela ter adicionado comentários.

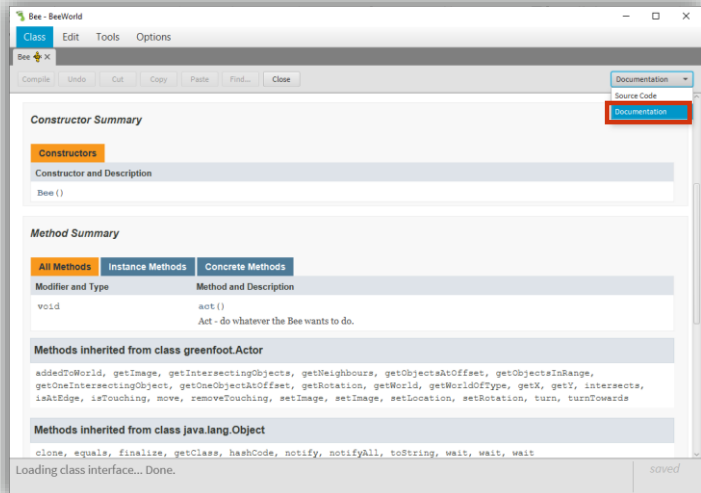
# Documentação

- A documentação descreve as propriedades da classe
- Para exibi-las, selecione Documentation no menu drop-down no canto superior direito do Code editor



**ORACLE**  
Academy

JF 3-3  
Código-Fonte e Documentação



Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados. 15

Observe que, se você alterar o nome do autor, o número da versão ou os comentários antes de act, eles agora aparecerão na documentação.

## Chamar Métodos de Forma Programática

- Os métodos devem ser chamados para comandar as instâncias a agirem em um jogo
- Chame os métodos de maneira programática gravando-os no corpo do método `act()` no espaço entre as chaves

```
public class Bee extends Actor
{
    /**
     * 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()
    {
        move(3);
        turn(15);
    }
}
```

Esses métodos serão chamados sequencialmente dentro do método `act()`.



# Componentes de Chamada do Método

- Componentes de chamada do método:
  - Tipo de retorno
    - Tipo de dados do valor de retorno
    - Os tipos de retorno void não exigem variáveis nem dados de retorno.
  - Nome do método
  - Lista de parâmetros para indicar o tipo de argumentos a ser chamado, se necessário
  - Ponto-e-vírgula para marcar o fim da chamada do método

```
public void act()  
{  
    move(3);  
    turn(15);  
}
```

Se você não tiver certeza do que um método faz, retorna ou exige, consulte a documentação dele ou a documentação da superclasse dele.

# Chamando Métodos - Exemplo 1

- Cada método é gravado no espaço entre as chaves

```
public class Bee extends Actor
```

```
{
```

Nome do método

Parâmetros

```
    * This method is called whenever  
    * the 'Act' or 'Run' button gets pressed in the environment.
```

```
    */
```

```
    public void act()
```

```
{
```

```
    move(3);
```

Ponto-e-vírgula

```
    turn(15);
```

```
}
```

```
}
```

Note que o código depois das chaves é recuado. Essa é uma boa prática de codificação e facilita a leitura.

## Chamando Métodos - Exemplo 2

- A chamada do primeiro método é gravada no corpo do método `act()`, terminando com um ponto-e-vírgula
- A chamada de cada método adicional é digitada logo abaixo, até todos os métodos serem inseridos no espaço entre as chaves

```
/**
 * Act - do whatever the Bee wants to do. This method is called whenever
 * the 'Act' or 'Run' button is pressed.
 */
public void act()
{
    move(3);
    turn(15);
}
```

Parâmetro - 3

O ponto-e-vírgula marca o fim da instrução de programação

Nome do método

À medida que inserirmos isso no método `act()`, eles serão sempre chamados sequencialmente.

## Métodos que Instruem Objetos a Executarem Ações

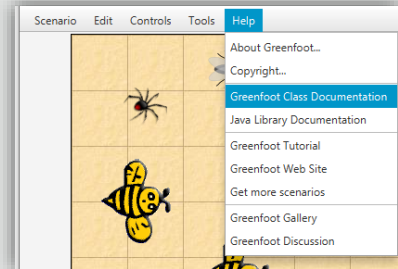
Nome do Método	Descrição
<code>void move(int distance)</code>	Atribui ao objeto um número de etapas para que ele seja movido, ou o comando simplesmente moverá quando os botões Act ou Run forem clicados.
<code>void turn(int amount)</code>	Atribui ao objeto um número de graus para que ele ser girado.
<code>void act()</code>	Dá ao objeto a oportunidade de executar uma ação no cenário. As chamadas do método são inseridas neste método.
<code>void setLocation(int x, int y)</code>	Atribui um novo local a este objeto.
<code>void setRotation(int rotation)</code>	Define uma nova rotação para este objeto.

Todos os métodos mostrados podem ser testados clicando com o botão direito do método em uma instância Ator e selecionando o método. Se ele exigir valores para os parâmetros, então esses valores serão solicitados

# Maneiras de Exibir Métodos Herdados de uma Classe

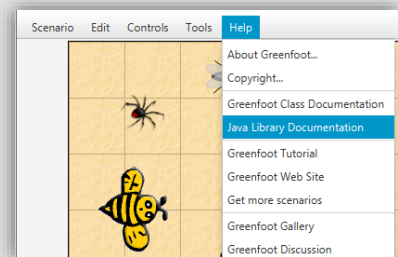
- Exiba a Documentação da Classe do Greenfoot

- Abra o Greenfoot
- Selecione Help
- Selecione a Documentação da Classe do Greenfoot



- Exiba a Documentação da Biblioteca Java

- Abra o Greenfoot
- Selecione Help
- Selecione a Documentação da Biblioteca Java



Os métodos herdados são os métodos que foram definidos na superclasse ou em um nível superior.

## Tarefas Sequenciais

- Uma tarefa simples, como ir para a escola, exige várias subtarefas:

- Acordar
- Tomar um banho
- Escovar os dentes
- Vestir-se...



- Dentro de uma subtarefa, pode haver mais subtarefas (caminhar até a escola requer que as pernas esquerda e direita andem para frente, em ordem)

Dividir um problema em problemas menores é uma técnica bastante conhecida para que possamos compreender melhor um cenário. Essa mesma técnica funciona bem na programação quando dividimos um problema em partes menores até chegarmos em um nível de especificidade tal que nos permita compreender o problema menor.

# Métodos Sequenciais

- Os métodos sequenciais são vários métodos executados pelo Greenfoot na ordem em que eles são gravados no programa
- Esses métodos possibilitam a um objeto executarem tarefas sequenciais, como correr e depois pular, ou reproduzirem um som depois que algo explode
- Os objetos podem ser programados para executarem métodos sequenciais sempre que o botão Act for clicado

```
public void act()  
{  
    move(3);  
    turn(15);  
}
```

A ordem em que realizamos as tarefas pode ser importante, dependendo do que estamos tentando fazer. Se tivermos duas tarefas –

Calçar a meia esquerda

Calçar a meia direita

A ordem dessas tarefas não mudará o resultado final – colocar as duas meias.

Se tivermos duas tarefas

Calçar a meia esquerda

Calçar o sapato esquerdo

A ordem dessas tarefas é importante. A menos que desejemos colocar a meia sobre o sapato...

## Relacionamentos If-then

- Muitas ações que nos cercam têm um relacionamento de causa e efeito, ou seja, um relacionamento "if-then"
- Se seu celular toca, você atende-o
- Se ele não toca, você não o atende
- Se uma flor começa a murchar, você rega-a
- Se ela parece estar saudável, você não coloca água nela



**ORACLE**  
Academy

JF 3-3  
Código-Fonte e Documentação

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados. 24

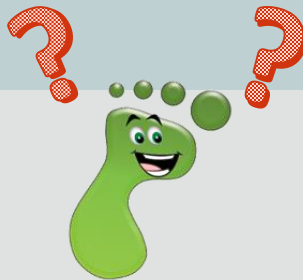
Você já deve ter visto as instruções IF em outras linguagens ou mesmo em pacotes de software, como em planilhas.



# Instruções de Decisão if

- Uma instrução IF será gravada para informar seu programa a só executar um conjunto de instruções de programação se e quando determinada condição for verdadeira

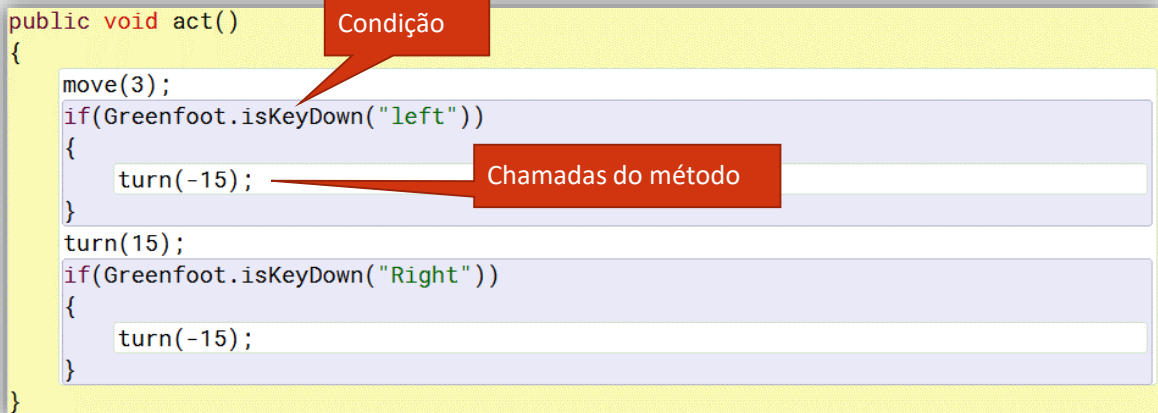
```
if (condition)
{
    instruction;
    instruction;
    ...
}
```



A instrução if pode controlar se um conjunto de instruções deve ou não ser executado. Você tirará bastante proveito da instrução if porque ela permite que o código reaja a determinados eventos no seu cenário, isto é, dois objetos colidindo, um usuário clicando em uma tecla ou um mouse selecionando um ator são apenas alguns exemplos em que talvez queiramos escolher se determinadas seções do código devem ser executadas.

## Componentes da Instrução de Decisão if

- A instrução de decisão if contém uma condição, que é uma expressão verdadeira ou falsa, e uma ou mais chamadas de métodos que são executadas caso a condição seja atendida



O diagrama mostra um trecho de código Java dentro de um método `act()`. O código é dividido em blocos de fundo amarelo e roxo. O primeiro bloco amarelo contém `move(3);`. O segundo bloco roxo contém uma instrução `if` com a condição `Greenfoot.isKeyDown("left")` e o corpo `{ turn(-15); }`. O terceiro bloco amarelo contém `turn(15);`. O quarto bloco roxo contém outra instrução `if` com a condição `Greenfoot.isKeyDown("Right")` e o corpo `{ turn(-15); }`. Duas anotações em caixas vermelhas apontam para partes do código: uma aponta para a condição `Greenfoot.isKeyDown("left")` e a outra aponta para a chamada de método `turn(-15);` dentro do primeiro bloco `if`.

```
public void act()
{
    move(3);
    if(Greenfoot.isKeyDown("left"))
    {
        turn(-15);
    }
    turn(15);
    if(Greenfoot.isKeyDown("Right"))
    {
        turn(-15);
    }
}
```

**ORACLE**  
Academy

JF 3-3  
Código-Fonte e Documentação

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados. 26

A condição `isKeyDown("left")` testa a tecla de seta para a esquerda.

A inclusão de um valor de giro negativo para o parâmetro `turn` significa que o ator girará no sentido anti-horário.

## Instrução de Decisão if - Exemplo

- No exemplo a seguir:
  - As teclas de seta para a esquerda e a direita no teclado fazem com que o objeto gire para a esquerda e para a direita
  - Se a condição for falsa, as chamadas do método definidas na instrução IF não serão executadas
  - O método move é executado independentemente da instrução IF

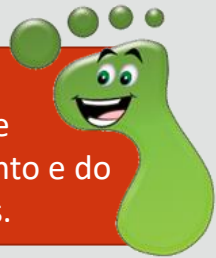
```
public void act()
{
    move(3);
    if(Greenfoot.isKeyDown("left"))
    {
        turn(-15);
    }//endif
    if(Greenfoot.isKeyDown("right"))
    {
        turn(15);
    }//endif
} //end method act
```

Lembre-se de que o método act() está em um loop contínuo. Portanto, os métodos estão sempre sendo chamados sequencialmente dentro de seus parênteses.

## Método isKeyDown()

- O método isKeyDown() é um procedimento pré-existente que o Greenfoot escuta para determinar se uma chave do teclado é pressionada durante a execução da programação
- Este método é chamado em uma classe usando uma notação de pontos

Quando um método não está na classe ou é herdado pela classe que você está programando, especifique a classe ou o objeto que contém o método antes do nome do método, seguido de um ponto e do nome do método. Essa técnica é denominada notação de pontos.



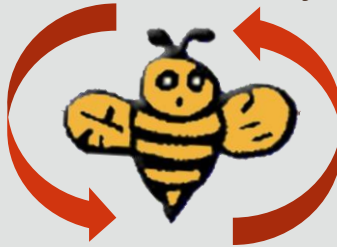
O método isKeyDown() costuma ser usado para dar ao usuário controle de um ator por meio do teclado.

## Orientação do Objeto no Mundo Real

- À medida que nos movemos no mundo em que vivemos, precisamos saber a orientação ou o senso de direção
  - Quando você dirige um carro, sempre precisa saber se ele está na pista correta
  - Quando um avião voa pelo ar, ele precisa saber onde está localizado em relação a outros aviões para que não aconteça uma colisão
  - Quando insere sua localização em um mapa em um celular, você recebe coordenadas que informam onde você está e o endereço

## Exibição da Orientação de um Objeto

- Os métodos informam-nos como um objeto está posicionado no mundo em relação a ele mesmo e a outros objetos
- Você pode chamar um método:
  - Com um tipo de dados específico, como booleano, para fazer uma pergunta sobre a orientação dele
  - No ambiente, para saber como o objeto está orientado no cenário



A orientação de um objeto no mundo e sua comparação com outros objetos são os blocos de construção do software que gravaremos no Greenfoot.

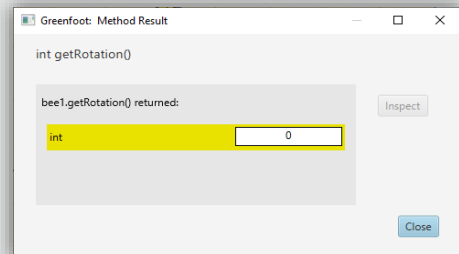
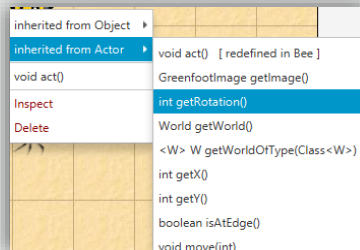
## Métodos que Retornam Informações sobre a Orientação de um Objeto

Nome do Método	Descrição
<code>int getRotation()</code>	Retorna a rotação atual do objeto.
<code>World getWorld()</code>	Retorna o mundo em que o objeto está.
<code>int getX()</code>	Retorna a coordenada x da localização atual do objeto.
<code>int getY()</code>	Retorna a coordenada y da localização atual do objeto.

360 graus é uma rotação completa. Cuidado porque 0 grau é o ângulo apontando para a direita da tela. Então, para baixo é 90, para a esquerda é 180 e para cima é 270 graus.

## Etapas para Chamar um Método que Exibe a Orientação de um Objeto

1. Clique com o botão direito do mouse na instância no mundo
2. Selecione Inherited from Actor para exibir os respectivos métodos
3. Chame (selecione) um método dentro de um tipo de dados específico para fazer uma pergunta ao objeto sobre a localização dele
4. O resultado do método será exibido
5. Observe o valor retornado e clique em Close



**ORACLE**  
Academy

JF 3-3  
Código-Fonte e Documentação

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados. 32

No Greenfoot, o tamanho padrão do mundo é 600 ao longo (x) e 400 para baixo (y).

Então, em um mundo deste tamanho, a parte superior esquerda é a posição (0,0) e a parte direita inferior é a posição (600,400).



# Terminologia

- Estes são os principais termos usados nesta lição:
  - Descrição da classe
  - Comentários
  - Instruções de decisão if
  - Chamando um método
  - Análise orientada a objeto
  - Métodos sequenciais

## Resumo

- Nesta lição, você deverá ter aprendido a:
  - Demonstrar as alterações do código-fonte para chamar métodos de maneira programática
  - Demonstrar as alterações do código-fonte para escrever uma declaração de decisão if
  - Descrever um método para exibir a orientação de um objeto



