

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

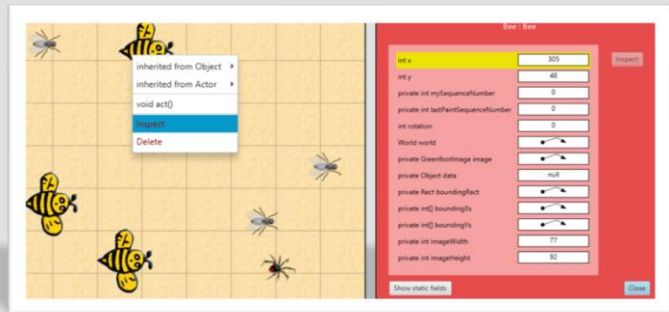
Academy

Java Fundamentals

3-2

Métodos, Variáveis e Parâmetros

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.

Objetivos

- Esta lição abrange os seguintes objetivos:
 - Definir parâmetros e como eles são usados nos métodos
 - Compreender a herança
 - Descrever propriedades de um objeto
 - Examinar a finalidade de uma variável
 - Analisar conceitos de programação e definir a terminologia



Exemplos de Métodos

- Para poder completar uma tarefa, como um exercício de casa de matemática, existem várias subtarefas:
 - O aluno completa o exercício de matemática
 - O aluno vai para a escola
 - O aluno entrega o dever de casa para o professor
- Em função das experiências obtidas na escola, combinadas com habilidades pré-programadas (como o raciocínio), o aluno é capaz de completar esta tarefa



Quando você está aprendendo a programar, uma das tarefas mais difíceis é dividir os problemas em etapas mais simples para que o computador compreenda. A maioria das pessoas pode realizar tarefas sem muitos problemas, mas pedir a elas que descrevam para outras pessoas em detalhes, passo a passo, como elas fizeram isso pode ser bem mais difícil do que parece. Isso requer prática, mas com experiência pode ser bem mais fácil.

Métodos

- Na programação, cada objeto tem um conjunto de operações (ou tarefas) que podem ser executadas
- Os programadores gravam um programa para informar a um objeto como e quando as tarefas devem ser executadas, como:
 - Comandar um objeto para executar uma ação
 - Fazer uma pergunta a um objeto para aprender mais sobre o que ele faz



Os métodos são um conjunto de operações ou tarefas a que as instâncias de uma classe podem pertencer. Quando um método for chamado, ele executará a operação ou a tarefa especificada no código-fonte.

Um objeto pode ser qualquer coisa que você deseje modelar: algo real como um carro ou algo em um jogo como um teletransporte (que não é algo real neste momento!). Aprenderemos sobre os benefícios de modelar objetos à medida que fizermos este curso, mas é importante que você saiba que já está familiarizado com a maioria dessas ideias.

Herança

- Os objetos do Greenfoot herdam os métodos e as propriedades de suas classes e superclasses
- Por exemplo, uma instância Jacaré herdaria os métodos da superclasse Ator e da classe Jacaré

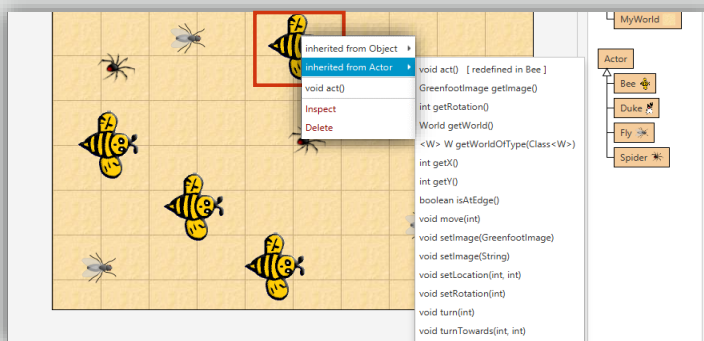
A herança significa que cada subclasse herda os respectivos métodos de sua superclasse.



Quando estamos falando sobre a classe Jacaré, podemos dizer que ela é uma subclasse de Ator. Também podemos dizer que Ator é uma superclasse de Jacaré. A classe Ator também terá uma superclasse. Então, quando estamos falando da classe Ator, ela é uma superclasse para Jacaré e uma subclasse para seu pai. Por isso, uma classe pode ser tanto uma superclasse quanto uma subclasse ao mesmo tempo.

Exibir Métodos Herdados no Menu do Objeto

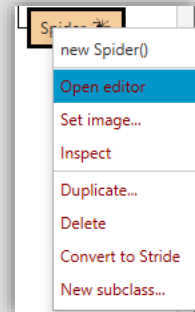
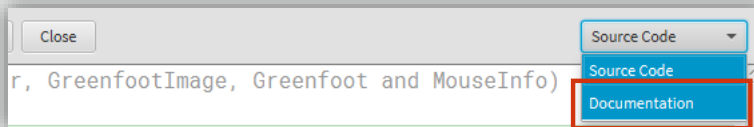
- O menu do objeto exibe todos os métodos que a instância herda de sua classe e de sua superclasse
 - Clique com o botão direito do mouse na instância para exibir o menu
 - A opção Inherited From Actor exibe uma lista dos métodos que a classe herda da superclasse Actor



Note que o primeiro menu pop-up mostra todos os métodos que são definidos na classe Abelha. Neste caso, apenas o método act(). O segundo menu drop-down informa que o ato que foi definido na superclasse Actor foi definido pela subclasse Abelha. Também vemos todos os métodos que herdamos da superclasse Actor.

Etapas para Exibir Métodos Herdados no Code Editor

- Clique com o botão direito do mouse em uma classe (neste exemplo em Aranha)
- Clique em Open Editor
- No Code editor, selecione Documentation no menu drop-down no canto superior direito



- Role até Method Summary

Method Summary	
All Methods	Instance Methods
Concrete Methods	
Modifier and Type	Method and Description
void	act () Act - do whatever the Spider wants to do.

Antes de selecionar um aviso da documentação na classe, você selecionou que só temos um método definido – act()

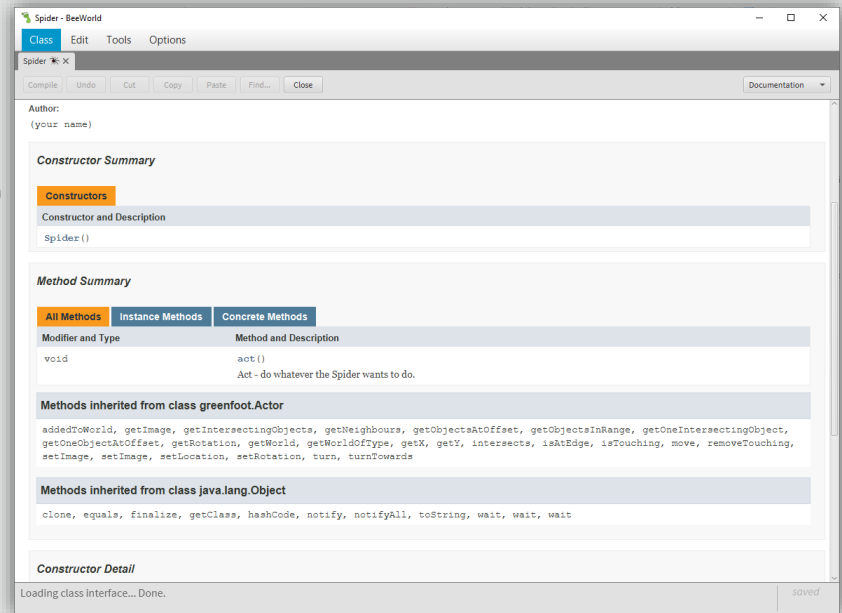
Resumo do Método

- O resumo do método é exibido nos métodos herdados da classe



ORACLE
Academy

JF 3-2
Métodos, Variáveis e Parâmetros



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.

9

Se você observar a página do código-fonte, poderá ver que existem comentários em toda ela. Atribuir a esses comentários um texto mais descritivo fará com que sua documentação seja mais complexa.

Componentes do Método

- Um método tem vários componentes que descrevem as operações ou tarefas que ele executa
 - Tipo de retorno: Especifica o tipo de dados que o método retorna
 - Nome do método: Descreve o que o método faz
 - Lista de parâmetros: Informações que são incluídas na chamada do método
- Exemplos de métodos:

```
void move(3)  
int getX()
```

Uma chamada de método comanda a instância para executar uma operação ou uma tarefa no Greenfoot. Leia o método para compreender qual operação ou tarefa deverá ser executada.



ORACLE
Academy

JF 3-2
Métodos, Variáveis e Parâmetros

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

O exemplo `void move(3)` instrui o ator a mover 3 casas em sua direção atual. O método não retorna um valor para a linha do programa que chama. Por isso, usamos a palavra `void`. Void significa que nenhum valor é retornado do método.

No segundo exemplo, chamamos o método `getX()`. Isso retorna a coordenada X atual do ator que é um número inteiro. No Java, representamos números inteiros como inteiros, então este método tem um tipo de retorno `int`.

Assinatura do Método

- A assinatura do método descreve as intenções do método
- Ela inclui os seguintes componentes:
 - Nome do método
 - Lista de parâmetros

```
void move(int)
```

Nome do
método

Lista de
Parâmetros ()



Normalmente, quando descrevemos um método, também incluimos seu tipo de retorno. Tecnicamente, no Java, a definição da assinatura do método não inclui o tipo de retorno.

Tipos de Retorno

- O tipo de retorno é a palavra no início do método que indica o tipo de informação que uma chamada de método retornará
- Existem dois tipos de retorno:
 - Void: não há retorno de dados - Emite um comando para o objeto
 - Não void: retorna dados - Faz uma pergunta a um objeto

```
void move(int)
```

Tipo de
retorno



ORACLE
Academy

JF 3-2
Métodos, Variáveis e Parâmetros

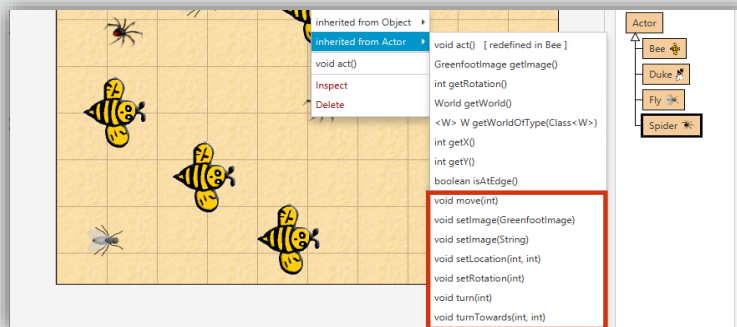
Copyright © 2022, Oracle e/ou suas empresas afiliadas. Oracle, Java e MySQL são marcas comerciais registradas da Oracle Corporation e/ou de suas empresas afiliadas. Outros nomes podem ser marcas comerciais de seus respectivos proprietários.

12

Normalmente, quando um método é void, estamos instruindo-o a fazer algo como mover e girar. Quando ele é não void, estamos solicitando a ele informações como as respectivas coordenadas atuais.

Métodos com Tipos de Retorno Void

- Os métodos com tipo de retorno void emitem um comando que realiza uma ação
- Eles incluem a palavra "void"
- Não retornam informações sobre o objeto
- É usado para fazer com que o objeto execute uma ação

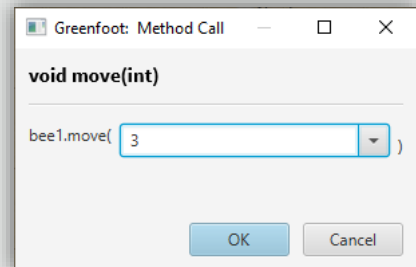
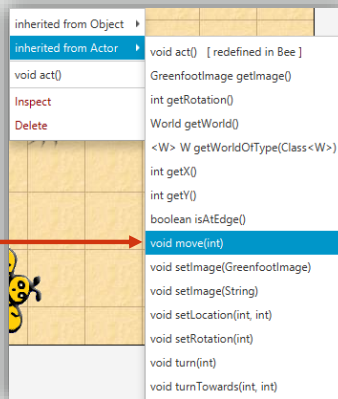


A escolha de nomes adequados facilita muito o trabalho com os objetos e métodos. Dessa forma, provavelmente você poderia deduzir o que cada método faz apenas vendo os nomes do método na screenshot. Dê uma olhada nos nomes dos métodos e veja de quais deles você consegue adivinhar a finalidade.

Chamando Métodos com Tipos de Retorno Void

- Você chamará métodos com tipos de retorno void:
- Para posicionar precisamente os objetos no cenário inicial (o ponto de partida do jogo)
- Para comandar objetos a executarem ações no jogo

Exemplo de seleção
O método move



ORACLE
Academy

JF 3-2
Métodos, Variáveis e Parâmetros

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.

14

Teste alguns dos outros métodos, como setLocation, setRotation e turn.

Fazer Perguntas aos Objetos

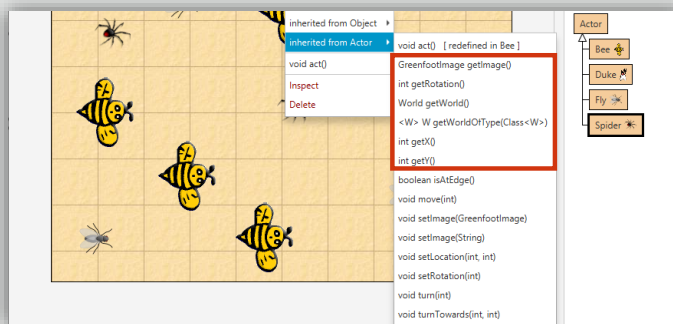
- Como programador, você fará perguntas aos objetos chamando métodos com tipos de retorno não void para aprender o que um objeto pode fazer ou o que ele fez no passado
- Por exemplo, na escola, os professores fazem perguntas aos alunos para ver se eles compreenderam o material explicado na aula daquele dia
- Os alunos fornecem respostas que permitem aos professores saberem o que eles aprenderam



Você pode escolher muitos objetos no mundo real e pensar em quais perguntas poderia fazer e como elas seriam retornadas para você. Exemplo – um cozinheiro. Quais métodos um fogão tem para informar a você dados sobre suas operações. Exemplos: Qual é a temperatura do fogão? O fogão está pronto? Há quanto tempo algo está cozinhando? O fogão está ligado?

Métodos com Tipos de Retorno Não Void

- Os métodos com tipos de retorno não void fazem uma pergunta ao objeto
- A assinatura do método não inclui a palavra "void"
- O método retorna informações sobre o objeto, mas não o altera nem o move



Mais adiante veremos quais são cada um dos tipos de Dados de retorno. Neste exemplo, temos um tipo int (número inteiro), World (instância Mundo) e boolean (verdadeiro ou falso).

Exemplos de Tipos de Retorno Não Void

- Número inteiro (exibido como int)
 - Refere-se a números inteiros
 - Pergunta ao objeto: Quantos?
- Booleano
 - Retorna um valor verdadeiro ou falso.
 - Tipos de perguntas que pode fazer a um objeto:
 - Você está tocando outro objeto?
 - Você está na extremidade do mundo?



Os tipos mais comuns que analisaremos são os números inteiros, as strings e os booleanos. Então vale a pena estarmos familiarizado com eles.

Parâmetros dos Métodos

- Os parâmetros fornecem métodos com dados adicionais para fazer com que um objeto execute uma tarefa quando são necessárias informações para chamar o método
- Os parâmetros são definidos por dois componentes:
 - Tipo de parâmetro
 - Nome do parâmetro

Os parâmetros são usados para comandar objetos a moverem-se ou para informar aos objetos qual tipo de resposta é esperada quando fazemos uma pergunta a um objeto.



Os parâmetros são um mecanismo que nos permite enviar valores ao objeto por meio de um método. Usaríamos parâmetros para girar, mover e posicionar instâncias de objetos no nosso mundo.

Exemplos de Parâmetros de Métodos

- Número inteiro (int): informe ou exiba valores numéricos

```
private void updateLives(int change)
{
    lives += change;
    updateImage(lives);
} //end of method
```

- Booleano: exibe valores verdadeiros ou falsos

```
private void storeCurrentPosition(boolean collided)
{
    if(collided){
        prevX = getX();
        prevY = getY();
    } //endif
} //end of method store current position
```

- String: informe ou exiba valores de texto

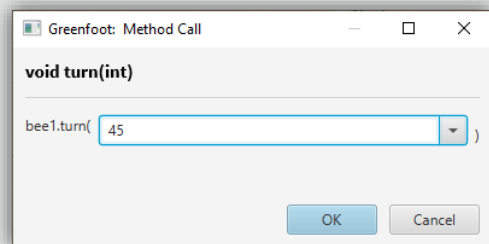
```
private void play(String sound)
{ //will play a sound file
    Greenfoot.playSound(sound);
} //end method Play
```

O sistema de coordenadas no Greenfoot baseia-se em números inteiros. Então, a maior parte da interação com números tenderá a seguir essa regra. Você pode tornar esse sistema mais exato usando números fracionários. Entretanto, nossas anotações se limitarão aos números inteiros.

As strings são para exibição de caracteres alfanuméricos.

Listas de Parâmetros de Métodos

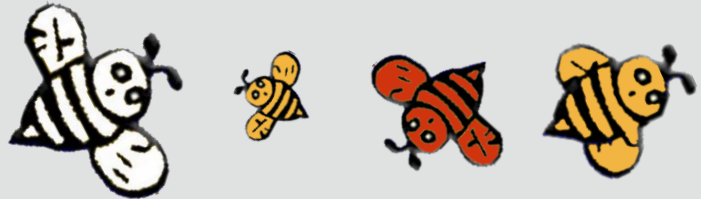
- As listas de parâmetros do método indicam se o método requer que mais informações sejam chamadas e quais tipos de informações
- As listas de parâmetros são exibidas como dados dentro de parênteses
- Em geral, elas têm dois estados:
 - Vazio: não são esperados dados para chamar o método (método `getRotation()`)
 - Não vazio: têm dados e esperam um ou mais parâmetros para chamar o método
 - isto é, método `turn(int)`



Na nossa screenshot, vemos que, se chamarmos o método `turn` dentro de uma instância da classe `Abelha`, ele solicitará que completemos valores de todos os respectivos parâmetros. Neste exemplo, o método `turn` só espera um valor inteiro. Portanto, só solicita uma entrada. Isso corresponde aos graus no sentido horário que desejamos girar.

Propriedades do Objeto

- As propriedades do objeto descrevem a aparência e as habilidades da instância, como:
 - Tamanho
 - Cor
 - Amplitude dos movimentos
- As propriedades podem ser exibidas e modificadas no código-fonte da classe
- Cada vez que você cria uma nova instância de um Ator, como uma Abelha, ela tem suas próprias propriedades



Você pode experimentar e pensar nas propriedades que um ser humano tem. Então, pense em algumas instâncias, como em você mesmo, e defina os valores das variáveis de cada um das propriedades dessa instância. Experimente isso com alguma outra pessoa. Cada pessoa tem as mesmas propriedades, mas pode haver valores diferentes para armazenar as variáveis.

Variáveis

- Uma variável, ou um campo, permite que a instância armazene informações para serem usadas imediatamente ou mais tarde
- Por exemplo, as propriedades dos objetos são variáveis que armazenam informações sobre a instância, como sua posição no mundo

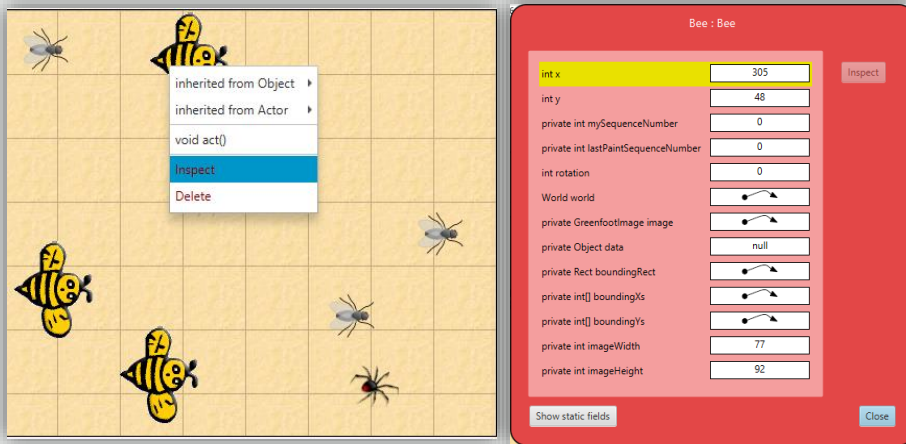
As variáveis da instância são a memória que pertence à instância da classe. Essa memória pode ser salva e acessada posteriormente, desde que a instância exista.



As variáveis da instância definem o estado do objeto. Um ator pode ser posicionado no ponto (15, 20) de frente para 20 graus e outro ator do mesmo tipo pode estar localizado a (130, 200) de frente para 180 graus. Os dois são instâncias da mesma classe, mas têm valores diferentes associados às suas variáveis ou aos seus campos. Em outras palavras, eles têm um estado diferente. São os valores nas variáveis que distingue-os.

Exibir Variáveis da Instância

- Clique com o botão direito do mouse na instância de um ator e clique em Inspect para exibir as variáveis da instância no inspetor de objeto



Esta técnica de observar as variáveis da instância de um ator é muito útil quando você deseja gravar os respectivos valores atuais. Depois você poderá usá-lo no nosso código.

Sintaxe de Programação

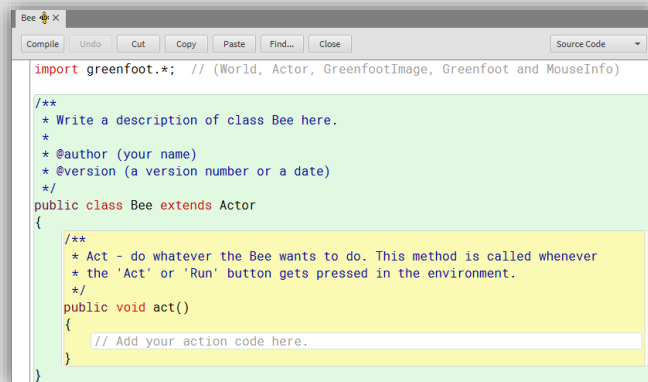
- O código-fonte especifica todas as propriedades e características de uma classe e de seus objetos
- Grave o código-fonte (também conhecido como sintaxe) no Code editor da classe para comandar a ação dos objetos no cenário

```
/**
 * Prepare the world for the start of the program.
 * That is: create the initial objects and add them to the world.
 */
private void prepare()
{
    Bee bee = new Bee();
    addObject(bee, 298, 184);
    Fly fly = new Fly();
    addObject(fly, 466, 289);
    Spider spider = new Spider();
    addObject(spider, 79, 79);
    fly.setLocation(462, 287);
    bee.setLocation(305, 48);
}
```

Lembre-se de que a sintaxe precisa ser exata para que o Java possa compilá-la. Como foi mencionado anteriormente, o fato de um código ser compilado e estar sintaticamente correto não significa que ele não tenha bugs.

Exibir o Código-Fonte da Classe

- No mundo, clique com o botão direito do mouse em uma classe e selecione Open Editor para exibir o Code editor
- O código-fonte exibido define o que os objetos na classe podem fazer



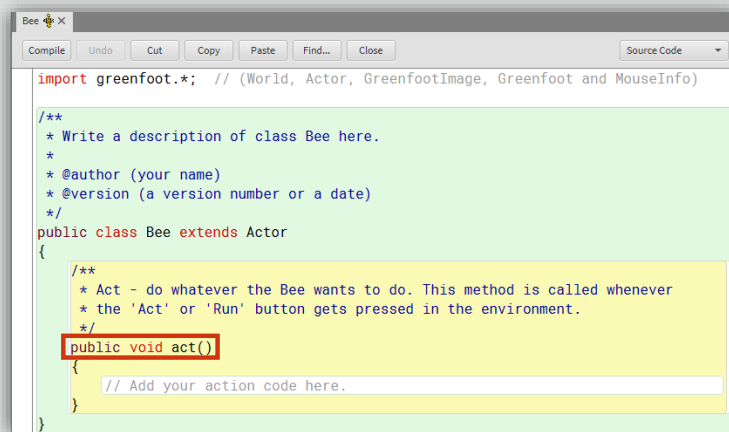
```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * 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.
    }
}
```

Observação: se você ler a documentação, quando você abrir sua janela de código, lembre-se de alterar a caixa drop-down na parte superior direita do Código-Fonte.

Método act()

- Sempre que os controles de execução Act ou Run forem clicados no ambiente, o objeto fará repetidamente o que ele está programado para fazer no método act()



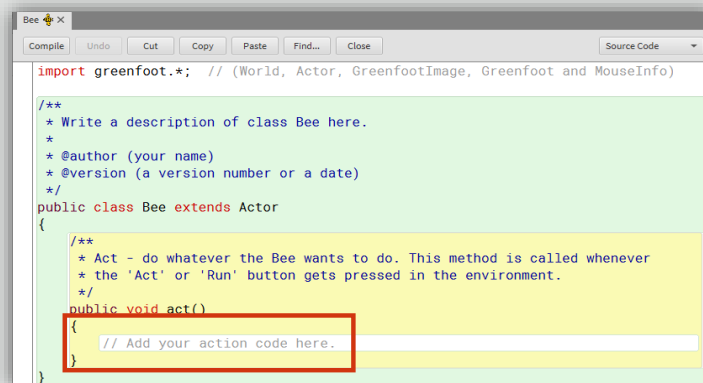
```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * 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.
    }
}
```

É importante observar aqui a palavra "repetidamente". O código em act() entrará em loop, isto é, ele executará todo o código, retornará ao início de act e será executado várias vezes consecutivamente.

Corpo do Método act()

- As chaves e o conteúdo dentro delas são o corpo do método
- Aqui você pode gravar o código para orientar as instâncias da classe a agirem quando os botões Act ou Run forem clicados

A screenshot of the Bee IDE window. The title bar says 'Bee'. The menu bar includes 'Compile', 'Undo', 'Cut', 'Copy', 'Paste', 'Find...', and 'Close'. There is a 'Source Code' dropdown menu. The code editor shows the following Java 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.
    }
}
```

The 'act()' method body is highlighted with a yellow background, and the opening and closing curly braces are enclosed in a red rectangular box.

Para cada "(" ou chave "{" (parêntese ou chave) que você adicionar, terá que adicionar o parêntese ou a chave de fechamento. Este é um dos maiores erros de sintaxe que você receberá. Deixar um número muito grande de parênteses abertos geralmente retornará o erro "reached end of file while parsing" no fim do código.

Exemplo do Método act()

- Chame os métodos move() e turn() no método act() para fazer com que as instâncias da classe sejam movidas ou giradas
- Os métodos devem ser gravados corretamente sem erros de digitação, caracteres faltando ou capitalização incorreta. Caso contrário, o código-fonte não será compilado

```
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(1);
        turn(15);
    }
}
```

Class compiled - no syntax errors

Adicione

move(1);

turn(15);

ao método act de um dos seus atores. Crie uma instância deste ator no seu mundo e selecione Run.

Chamar Métodos no Método act()

- Para chamar métodos no método act(), grave-os em sequência da seguinte maneira:
 - Nome do método em CamelCase
 - Parênteses, com a lista de parâmetros, caso necessário
 - Ponto-e-vírgula no fim da instrução

```
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(1);
        turn(15);
    }
}
```

Esquecer de adicionar um ponto-e-vírgula no fim dos comandos é um dos erros mais comuns de sintaxe. Normalmente, o Java detectará esse erro de ponto-e-vírgula faltando e o apontará para você.

Processo de Depuração no Greenfoot

- Caracteres incorretos ou gravados incorretamente no código-fonte dispararão mensagens de erro
- Se houver algum erro, será exibida uma mensagem de erro. O programador deverá corrigi-lo antes de o programar funcionar
 - O Greenfoot fornece essas mensagens de erro para que seja mais fácil corrigir os erros e aprender a evitá-lós

A depuração é o processo que consiste em encontrar e eliminar bugs, ou seja, erros em um programa de software.



Quanto mais você programa, mais familiarizado fica com as mensagens que são geradas por meio dos erros de sintaxe.

Erro de Sintaxe - Exemplo

- Falta um ponto-e-vírgula no método move()



Isso retornaria uma mensagem de erro de ponto-e-vírgula ";" esperada ao realçar a linha de código move(1).

Mensagem de Erro - Exemplo

- Aparece uma mensagem de erro na parte inferior da tela e o código incorreto é realçado



```
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()
    {
        move(1);
        turn(1;); // expected
    }
}
```


Terminologia

- Estes são os principais termos usados nesta lição:
 - Depurar
 - Herança
 - Variável da instância
 - Método
 - Chamada do método
 - Parâmetro
 - Tipo de retorno
 - Assinatura do método
 - Variável

Resumo

- Nesta lição, você deverá ter aprendido a:
 - Definir parâmetros e como eles são usados nos métodos
 - Compreender a herança
 - Descrever propriedades de um objeto
 - Examinar a finalidade de uma variável
 - Analisar conceitos de programação e definir a terminologia



