

The Oracle Academy logo 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

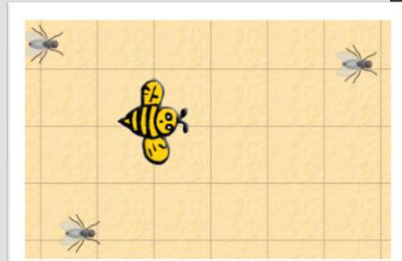
Academy

Java Fundamentals

3-5

Randomização e Construtores

ORACLE
Academy



```
if (Greenfoot.getRandomNumber(100) < 10)
{
    turn(Greenfoot.getRandomNumber(20));
}
```

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

Objetivos

- Esta lição abrange os seguintes objetivos:
 - Criar comportamentos randomizados
 - Definir operadores de comparação
 - Criar instruções de controle if-else
 - Criar uma instância de uma classe
 - Reconhecer e descrever uma notação de pontos



Método getRandomNumber()

- O método getRandomNumber() é um método estático que retorna um número aleatório entre zero e um limite de parâmetro
- Este método é usado para eliminar a previsibilidade no seu programa
- Assinatura do método:

```
public static int getRandomNumber(int limit)
```

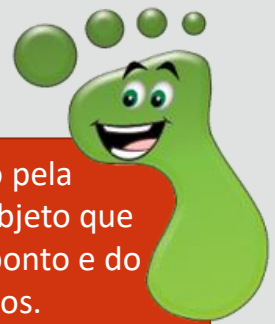
A randomização permite criar jogos que variarão toda vez que forem reproduzidos e, conseqüentemente, ficarão cada vez mais divertidos.

Os métodos estáticos são métodos que pertencem a uma classe, e não a uma instância. Essa não é uma ideia fácil de ser compreendida neste momento, mas você já utilizou os métodos quando usamos o método isKeyDown. Explicaremos a diferença mais adiante no curso.

Notação de Pontos

- As novas subclasses que você cria não herdam o método `getRandomNumber()`
- Esse método deve ser chamado na classe do Greenfoot usando a notação de pontos
- Exemplo:

```
public void act()  
{  
    Greenfoot.getRandomNumber(20);  
}
```



Quando você quiser usar um método, mas ele não for 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.

Para obter acesso ao método `getRandomNumber`, é necessário informar ao java que ele pode ser encontrado na classe do Greenfoot. Então escrevemos `Greenfoot.getRandomNumber()`.

Formato da Notação de Pontos

- O formato do código da notação de pontos inclui o seguinte:
 - Nome da classe ou do objeto a que o método pertence
 - Ponto
 - Nome do método a ser chamado
 - Lista de parâmetros
 - Ponto-e-vírgula

```
className.methodName (parameters);  
objectName.methodName (parameters);
```



Durante a referência à classe que estamos codificando, podemos usar a palavra "this" opcional para representar a classe atual. Então quando usamos anteriormente `move(2)`, podíamos ter dito `this.move(2)`.

"this" representa o objeto para o qual estamos editando o código, mas é opcional e costuma ser omitido.

Notação de Pontos - Exemplo

- O método `getRandomNumber()` mostrado abaixo:
 - Chama um número entre 0 e 15, não incluindo 15
 - Retorna um número aleatório entre 0 e 14

```
public void act()  
{  
    Greenfoot.getRandomNumber(15);  
}
```

`Greenfoot.getRandomNumber(15)` significa que retornará um dos 15 números aleatórios entre 0 e 14.

E se quiséssemos um número aleatório entre 1 e 10? Bastaria usarmos
`Greenfoot.getRandomNumber(10)+1;`

API do Greenfoot

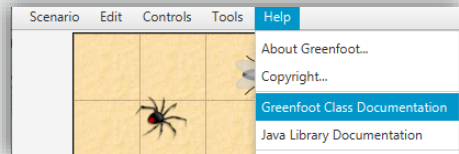
- Consulte a API (Application Programmers Interface) do Greenfoot para examinar métodos adicionais a serem chamados usando a notação de pontos

A API do Greenfoot lista todas as classes e os métodos disponíveis no Greenfoot

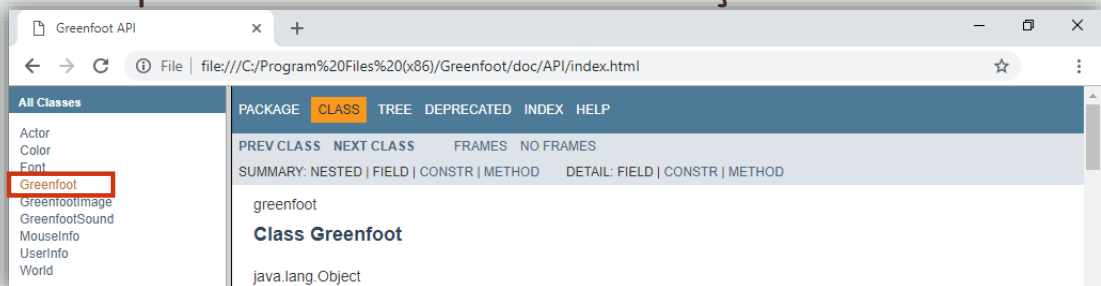


Etapas para Exibir Métodos na Classe do Greenfoot

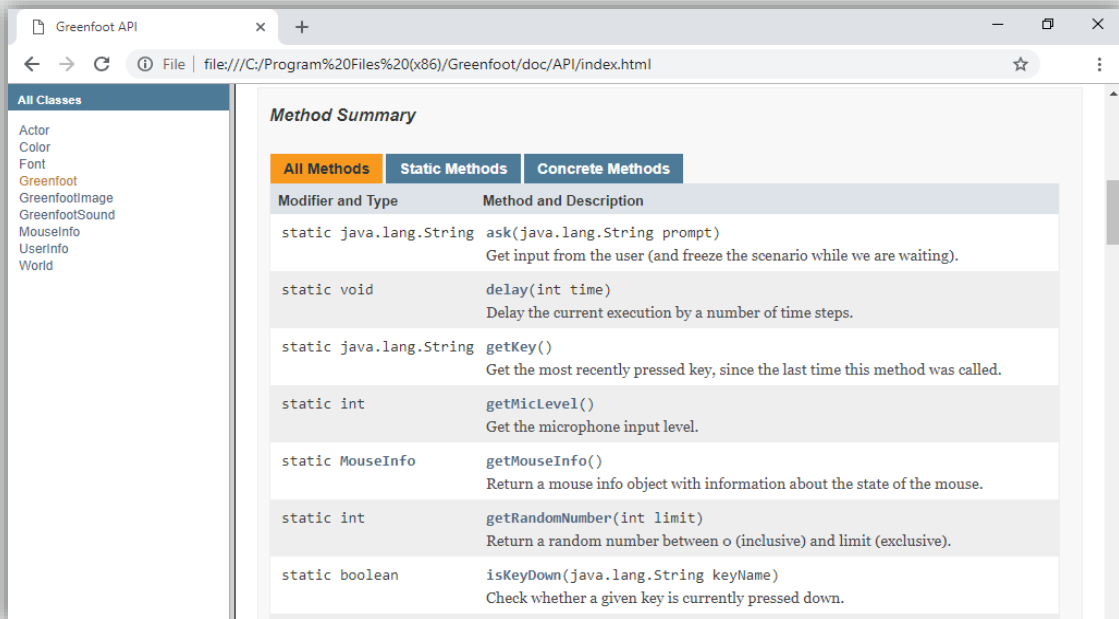
- No ambiente do Greenfoot, selecione o menu Help
- Selecione a Documentação da Classe do Greenfoot



- Clique na classe do Greenfoot
- Verifique as assinaturas e as descrições do método



API do Greenfoot



The screenshot shows a web browser window displaying the Greenfoot API documentation. The left sidebar lists 'All Classes' including Actor, Color, Font, Greenfoot, GreenfootImage, GreenfootSound, MouseInfo, UserInfo, and World. The main content area is titled 'Method Summary' and features a table with three tabs: 'All Methods', 'Static Methods', and 'Concrete Methods'. The 'Static Methods' tab is selected, showing a table of static methods.

Modifier and Type	Method and Description
static java.lang.String	ask (java.lang.String prompt) Get input from the user (and freeze the scenario while we are waiting).
static void	delay (int time) Delay the current execution by a number of time steps.
static java.lang.String	getKey () Get the most recently pressed key, since the last time this method was called.
static int	getMicLevel () Get the microphone input level.
static MouseInfo	getMouseInfo () Return a mouse info object with information about the state of the mouse.
static int	getRandomNumber (int limit) Return a random number between 0 (inclusive) and limit (exclusive).
static boolean	isKeyDown (java.lang.String keyName) Check whether a given key is currently pressed down.

Quando criamos atores, herdamos da classe Ator.

Quando criamos mundos, herdamos da classe Mundo. Vale a pena aprender quais métodos estão disponíveis nessas duas classes.

Operadores de Comparação

- Use os operadores de comparação para comparar um valor randomizado com outro valor em uma instrução de controle
 - O exemplo abaixo determina se o número aleatório é menor que 20
 - Caso seja, o objeto gira dez graus

```
public void act()  
{  
    if (Greenfoot.getRandomNumber(100) < 20)  
    {  
        turn(10);  
    }  
}
```



Os operadores de comparação são símbolos que comparam dois valores

Lembre-se de que `getRandomNumber(100)` gerará um número entre 0 e 99.

Símbolos de Operadores de Comparação

Símbolo	Descrição
<	Menor que
>	Maior que
<=	Menor que ou igual
>=	Maior que ou igual
==	Igual
!=	É diferente de

Um erro comum durante uma comparação É Igual a é só adicionar um sinal de igualdade "=". Isso tentará atribuir um segundo valor ao primeiro e, na maioria dos casos, gerará um erro de sintaxe.

Problema de Jogo Resolvido com o Comportamento Aleatório

- Problema:

- Um objeto Mosca deve mover-se aleatoriamente, fazendo com que mais desafiador para o objeto controlado pelo teclado, uma Abelha, capturá-lo

- Solução:

- A Mosca deve fazer pequenos giros enquanto se move
- Para codificar esta solução, gire a Mosca um número aleatório de graus, até 20 graus, 10% do tempo em que ela move-se

```
public void act()  
{  
    if (Greenfoot.getRandomNumber(100) < 10)  
    {  
        turn(Greenfoot.getRandomNumber(20));  
    }  
}
```

Formato do Comportamento Aleatório

- A instrução de programação a seguir inclui o seguinte:
 - Uma instrução de controle IF com o método `getRandomNumber()`
 - Um limite de parâmetros igual a 100
 - O operador de comparação `<`
 - O número 10 para limitar de 0 a 9 o intervalo de valores a serem retornados
 - O corpo do método com a instrução para indicar que o objeto deverá girar até 20 graus se a condição for verdadeira

```
if (Greenfoot.getRandomNumber(100) < 10)
{
    turn(Greenfoot.getRandomNumber(20));
}
```

Formato do Comportamento Aleatório

- O problema com o exemplo anterior é que a mosca sempre girará em círculo no sentido horário
- Podemos mudar isso modificando o parâmetro no método turn, como é mostrado a seguir
- Isso gerará um número aleatório entre -45 e +44
- Lembre-se de que um valor de giro negativo faz o movimento no sentido anti-horário

```
public void act()
{
    move(1);
    if (Greenfoot.getRandomNumber(100) < 10)
    {
        turn(Greenfoot.getRandomNumber(90)-45);
    }
}
```

Você pode usar alguns números aleatórios de sua escolha para testar a lógica do movimento da mosca usando papel e lápis caso não esteja convencido da faixa que será retornada.

Comportamento Condicional

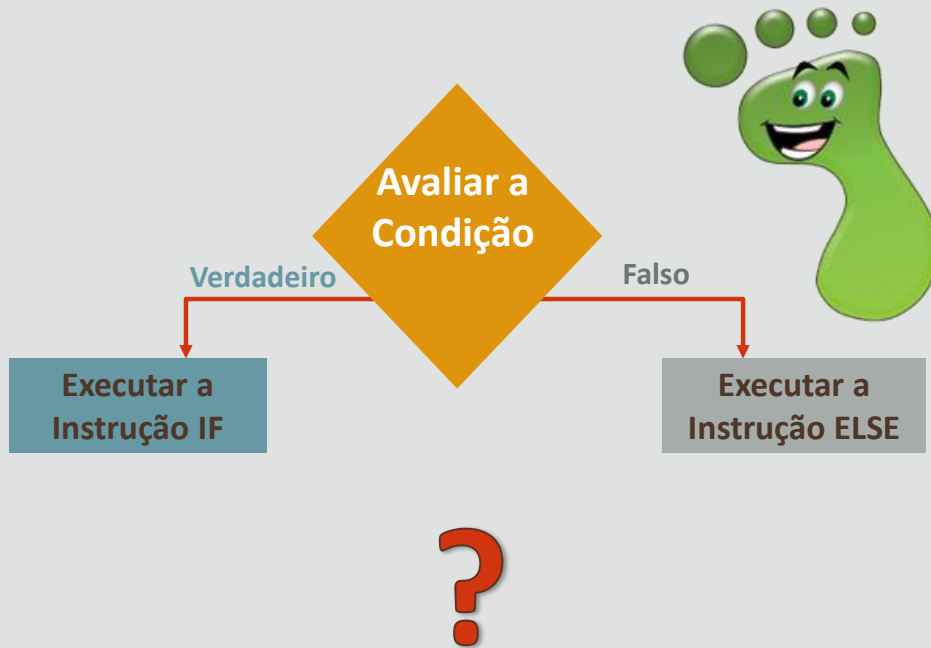
- As instâncias podem ser programadas para executarem comportamentos específicos se uma condição não for atendida, usando uma instrução if-else
- Por exemplo, se uma instância estiver programada para girar 6% do tempo, o que ela fará nos 94% do tempo restante?

Uma instrução if-else executará seu primeiro segmento de código se uma condição for verdadeira e executará seu segundo segmento de código se uma condição for falsa, mas não executará os dois



Precisamos decidir se são necessárias duas instruções IF ou uma instrução if-else. Se você quiser que as duas seções de código sejam executadas, então temos duas instruções IF. Se você sempre quiser que uma ou outra seja executada, então use uma instrução if-else.

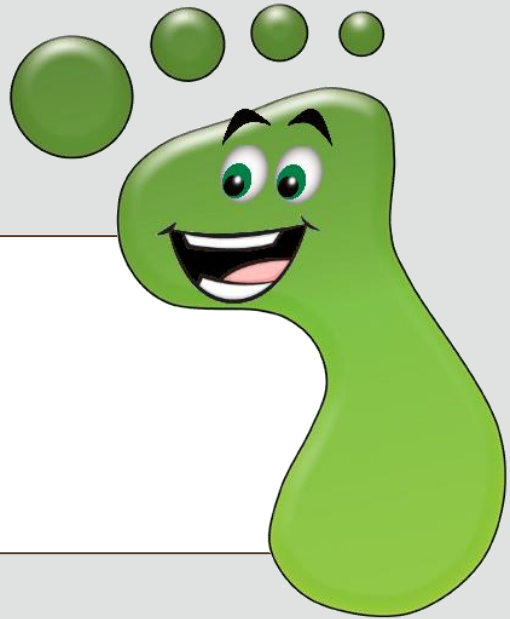
Execução da Instrução if-else



Em uma instrução IF – ELSE, só uma das instruções de código será executada.

Formato da Instrução if-else

```
if (condition)
{
    statements;
}
else
{
    statements;
}
```



Note que não existe um ponto-e-vírgula depois dos parênteses de um comando IF.

Exemplo da Instrução if-else

- Se for selecionado um número entre 0 e 6, gire 10 graus
- Caso contrário, gire 5 graus

```
public void act()
{
    move(1);
    if (Greenfoot.getRandomNumber(100) < 7)
    {
        turn(10);
    }
    else
    {
        turn(5);
    }
}
```

Criação Automática de Instâncias

- Usando a subclasse Mundo, as instâncias do ator podem ser programadas para aparecerem automaticamente no mundo quando um cenário for inicializado
- No Greenfoot, este é o comportamento padrão das instâncias:
 - A instância da subclasse Mundo é adicionada automaticamente ao ambiente depois da compilação ou da inicialização do cenário
 - As instâncias da subclasse Ator devem ser adicionadas manualmente pelo jogador

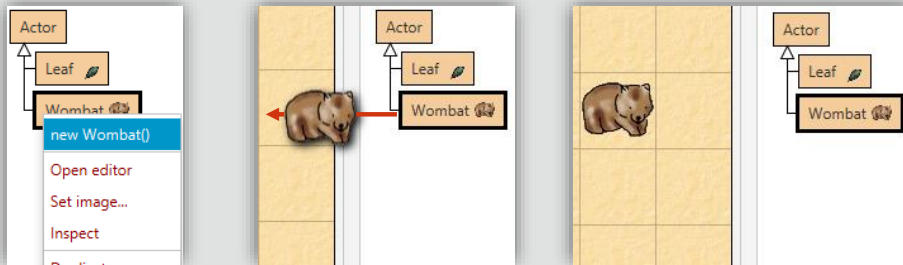


Mais adiante veremos que existe um método mais rápido usando o comando "Save the World".

Criação Automática de Instâncias em um Cenário

- Problema:

- Quando um cenário do Greenfoot (como folhas e wombates) é iniciado, as instâncias precisam ser adicionadas manualmente pelo jogador para começar a jogar



- Solução:

- Instâncias do programa serem adicionadas automaticamente ao mundo quando o cenário é inicializado

Código-fonte da Classe “World”

- O nome padrão da subclasse “World” é “MyWorld”

```
public class MyWorld extends World
{
    /**
     * Constructor for objects of class MyWorld.
     *
     */
    public MyWorld()
    {
```

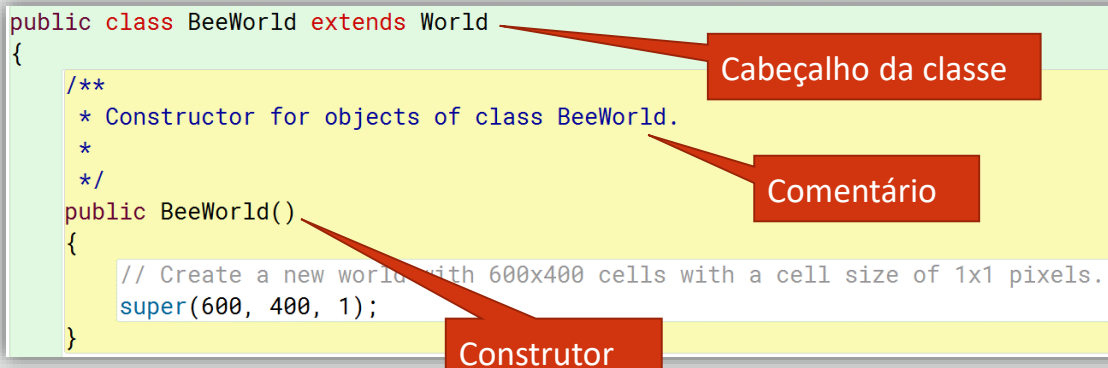
- Para identificar explicitamente sua classe “World” atual, talvez seja útil renomeá-la. Atualize todas as ocorrências de “MyWorld” no código-fonte para “BeeWorld”

```
public class BeeWorld extends World
{
    /**
     * Constructor for objects of class BeeWorld.
     *
     */
    public BeeWorld()
    {
```

Código-Fonte da Classe Mundo

- Para compreender como automatizar a criação das instâncias Ator, você precisa entender como o código-fonte da classe Mundo está estruturado
- O construtor Mundo é usado para automatizar a criação de instâncias Ator quando o cenário é inicializado

```
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);
    }
}
```



ORACLE
Academy

JF 3-5
Randomização e Construtores

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados. 23

Um construtor normalmente é definido por

```
public <classname>()
```

No exemplo acima, temos public BeeWorld()

Construtores

- Construtores:

- Definem o tamanho e a resolução da instância
- Não têm um tipo de retorno
- Têm o mesmo nome que o nome da classe
- Por exemplo, um construtor Mundo é denominado Mundo

Um construtor é um tipo especial de método que é executado automaticamente quando uma nova instância da classe é criada

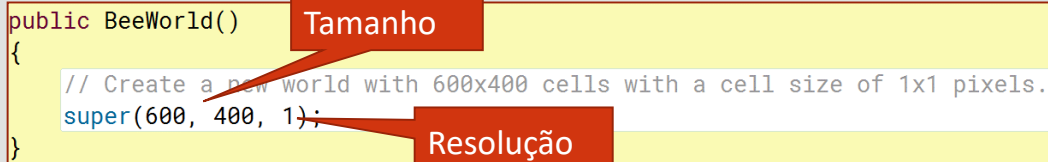


Os construtores serão explicados em detalhes mais adiante, mas eles fornecem um mecanismo excelente para configurar valores padrão para os campos de classe na instância do objeto.

Construtor Mundo - Exemplo

- O exemplo de construtor a seguir cria a instância da superclasse Mundo da seguinte maneira:
 - Tamanho: $x = 600$, $y = 400$
 - Resolução: 1 pixel por célula
 - A palavra-chave `super` no corpo do construtor chama a superclasse do construtor Mundo para cada instância da subclasse BeeWorld

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

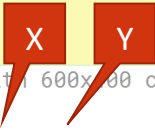


Podemos modificar facilmente os valores do Mundo aqui. Depois de alterado, seu cenário refletirá o novo tamanho após uma compilação.

Criação Automática de Instâncias Ator

- Esse construtor Mundo adiciona um novo objeto Abelha às coordenadas X e Y especificadas usando o método addObject()

```
/**
 * 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);
    addObject (new Bee(), 150, 100);
}
```



Como o construtor BeeWorld() só é chamado quando BeeWorld é criada, esse código é executado uma única vez.

Método addObject()

- O método `addObject()` é um método da classe `Mundo` que adiciona um novo objeto ao mundo nas coordenadas `x` e `y` específicas
- Ele inclui o seguinte:
 - A palavra-chave `new` para informar ao Greenfoot para criar um novo objeto de uma classe específica
 - Parâmetros de métodos:
 - Objeto denominado com base na classe `Ator`
 - Valor inteiro da coordenada `X`
 - Valor inteiro da coordenada `Y`
- Definição do método:

```
void addObject(Actor object, int x, int y)
```

Quando `new <classname>()` é chamado, ele procura um construtor para essa classe. Se não houver um, ele assumirá como seu padrão de tipo os valores do campo da classe, isto é, os valores inteiros tornam-se 0

Palavra-chave new

- A palavra-chave new cria novas instâncias de classes existentes
- Ela começa com a palavra-chave new, seguida do construtor a ser chamado
 - A lista de parâmetros passa argumentos (valores) para o construtor que são necessários para inicializar as variáveis de instância do objeto
 - O construtor padrão tem uma lista de parâmetros vazia e define as variáveis de instância do objeto como seus valores padrão



```
new ConstructorName()
```

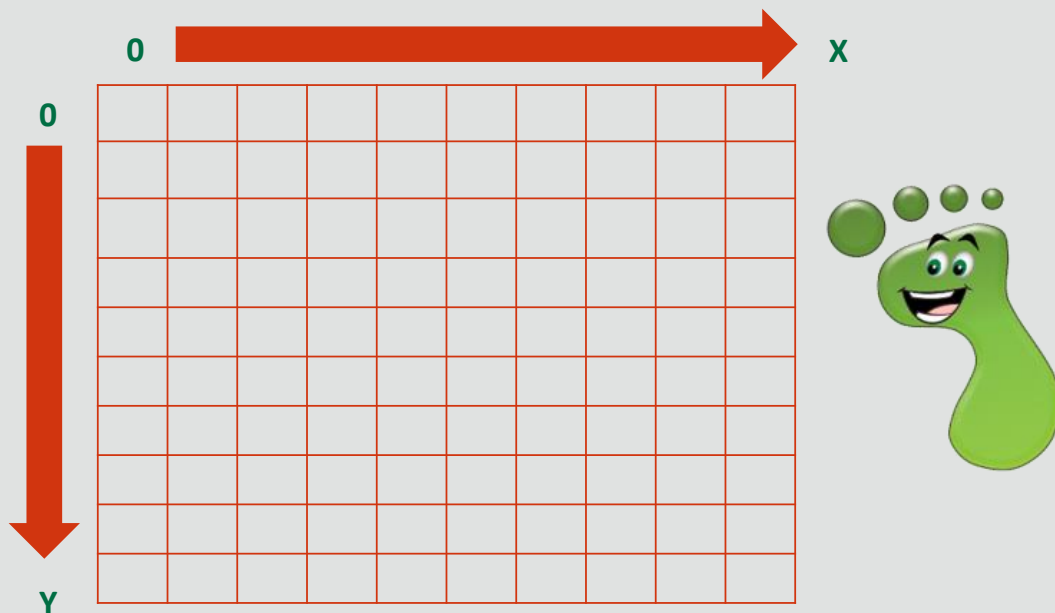
ORACLE
Academy

JF 3-5
Randomização e Construtores

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados. 28

Se você não definir um construtor para suas classes, o compilador Java gerará um construtor padrão. Você não verá isso no código.

Sistema de Coordenadas do Mundo no Greenfoot



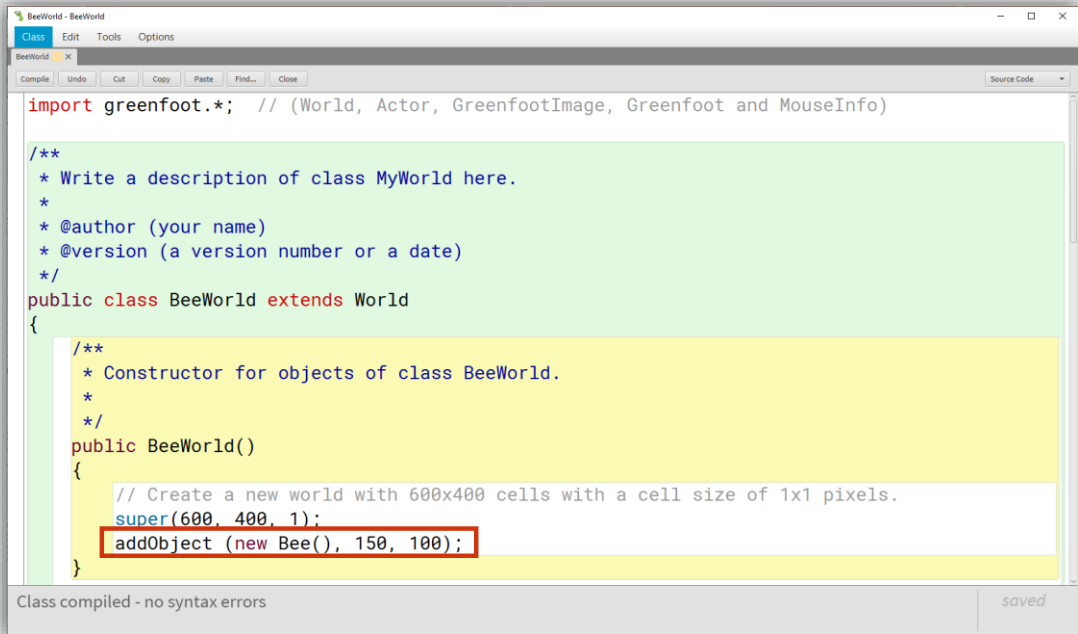
ORACLE
Academy

JF 3-5
Randomização e Construtores

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados. 29

A parte superior esquerda é o ponto (0,0) e a parte inferior direita é as dimensões do seu mundo.

Adicionar Objetos Usando o Construtor Mundo - Exemplo



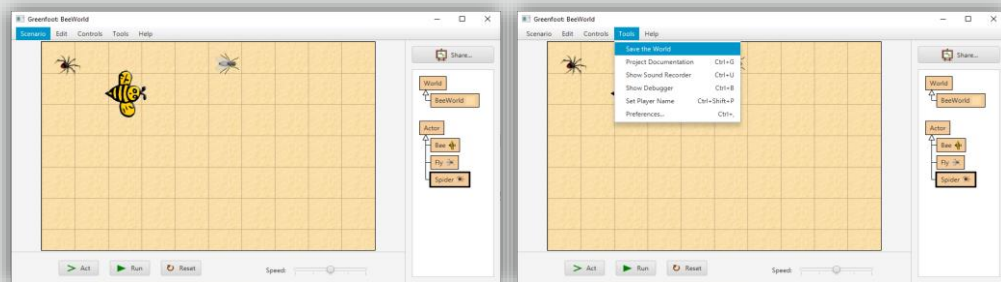
```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class MyWorld here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
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);
        addObject (new Bee(), 150, 100);
    }
}
```

Podemos ver uma nova instância da classe Abelha sendo criada e posicionada no ponto (150, 100).

Recurso Save the World

- O Greenfoot tem uma maneira adicional de configurar a posição inicial dos Atores usando o recurso "Save the World"
- Posicione os atores ao redor do mundo manualmente em uma posição inicial
- Em seguida, selecione Tools -> Save the World



Recurso Save the World

- Isso cria um novo método no mundo denominado `prepare()` e cria uma chamada para ele dentro do construtor
- O método `prepare()` criará instâncias do ator e adicionará essas instâncias ao mundo no local em que você posicionou-as
- Esse será um recurso muito útil se houver muitos objetos para serem posicionados

```
public BeeWorld()  
{  
    // Create a new world with 600x400  
    super(600, 400, 1);  
    addObject (new Bee(), 150, 100);  
    prepare();  
}
```

ORACLE
Academy

JF 3-5
Randomização e Construtores

```
/**  
 * Prepare the world for the start of the program.  
 * That is: create the initial objects and add them to the world.  
 */  
private void prepare()  
{  
    Fly fly = new Fly();  
    addObject(fly, 334, 42);  
    Spider spider = new Spider();  
    addObject(spider, 46, 48);  
}
```

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

`Bee bee = new Bee()` cria uma nova instância Abelha que pode ser acessada por meio da variável de referência denominada `abelha`. Lembre-se de que o Java faz distinção entre letras maiúsculas e minúsculas. Portanto, `Abelha` e `abelha` são tratadas de forma diferente. Então, em vez de dizer `addObject (new Bee(), 100,100)`, isso é substituído por

```
Bee bee = new Bee();
```

```
addObject(bee, 100,100);
```

Isso fornece algumas opções adicionais que exploraremos mais adiante.

Terminologia

- Estes são os principais termos usados nesta lição:
 - Operadores de comparação
 - Construtor
 - Notação de pontos
 - Palavra-chave new

Resumo

- Nesta lição, você deverá ter aprendido a:
 - Criar comportamentos randomizados
 - Definir operadores de comparação
 - Criar instruções de controle if-else
 - Criar uma instância de uma classe
 - Reconhecer e descrever uma notação de pontos
 - Usar o recurso Save the World



