Administração Central

ROBOCODE

ROBÓTICA PAULA SOUZA

2019

São Paulo

www.robotica.cpscetec.com.br
Rua dos Andradas, 140 • Santa Ifigênia • 01208-000 • São Paulo • SP • Tel.: (11) 3324.3300

Material Didático sobre Robocode

1 Definição de Classes em Orientação a Objetos

Como falamos no material anterior, o **Robocode** utiliza a linguagem **Java** e **Orientação a Objetos** para funcionar.

Neste material vamos explorar um pouquinho mais desses conceitos e aplicar nossos conhecimentos em uma das principais classes do Robocode: a classe **Robot**.

Para entendermos o conceito de uma **classe** em Orientação a Objetos, precisamos saber que a classe possui **atributos** e **métodos**. Assim, sabemos que os atributos são valores armazenados e os métodos são comportamentos. Por exemplo, se precisarmos criar uma classe Pessoa, sabemos que são atributos(características físicas) a altura, o peso, a idade, além de outras características que a classe pessoa possa ter como o número do RG ou do CPF, endereço, etc, de acordo com a necessidade que essas características vão apresentar para o seu projeto. Agora em relação ao comportamento, a classe Pessoa pode indicar uma ação contendo instruções para uma determinada classe. Assim se falarmos de comportamentos físicos podemos citar comer, locomover, dormir, ou se estivermos falando de uma regra de negócio que gerencie os dados das pessoas podemos citar cadastrarDados, mostrarDados. Esses métodos serão chamados no método principal (main() , no nosso caso no método run()), e podem necessitar da passagem de parâmetros, ou seja, valores atribuídos em uma variável em que esse valor é um endereço, uma referência. Vamos visualizar algumas classes em seus diagramas de classes:

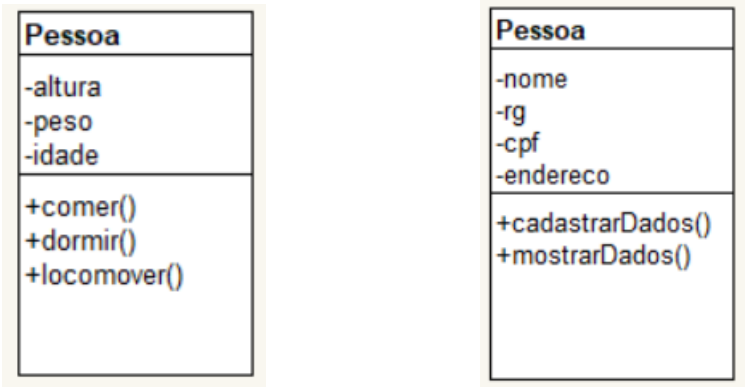

Figura 1 – Diagramas de classes utilizadas no exemplo
Fonte: do autor

2 Convenção para nomes em JAVA

A ideia de convenção em linguagens de programação faz com que todos os programados possam trabalhar da mesma forma. É uma espécie de combinado para tornar os programas mais compreensíveis, mais fáceis de serem lidos por outros programadores.

Essa convenção também ajuda na informação sobre a função do identificador, ou seja, se se trata de um pacote, constante, classe, atributo, método, etc.

O nome da **classe** deve ser um substantivo, em maiúsculas e minúsculas com a primeira letra de cada palavra interna em maiúscula. Deve ser simples e descritivo, evitando-se palavras-ligadas, sem usar todas em siglas e abreviaturas, de forma semântica, ou seja, que o nome possa passar o significado do que aquela classe representa em seu programa. No exemplo acima, nomeamos nossa classe **Pessoa**, pois é o que desejamos que ela represente.

Os **atributos** devem seguir a mesma linha de raciocínio das classes, porém devem ser escritos com a primeira letra em minúscula.

Os **métodos** devem ser verbos, com a letra minúscula em primeiro lugar, com a primeira letra de cada palavra interna em maiúscula.

3 A classe Robot

A classe básica Robot é a classe estendida quando vamos criar nossos próprios robôs no Robocode. Existem alguns padrões utilizados nessa classe:

- ✓ **heading** - ângulo absoluto em graus com 0 voltado para cima na tela, positivo no sentido horário. 0 <= heading <360.
- ✓ **bearing** - ângulo relativo a algum objeto do rumo do seu robô, positivo no sentido horário. -180 < bearing <= 180
- ✓ Todas as **coordenadas** são expressas como (**x, y**).
- ✓ Todas as **coordenadas são positivas**.
- ✓ A **origem (0,0)** está no canto inferior esquerdo da tela.
- ✓ **X positivo para direita**.
- ✓ **Y positivo para cima**.

4 Métodos da Classe Robot

Vamos detalhar um pouco mais os métodos da classe Robot. Quando o método já está criado para uma determinada classe, só fazemos a chamada ao método quando vamos utilizá-lo em nosso código. Cada método apresentado aqui necessitará de um valor para o parâmetro. Nesse material trataremos dos métodos para realizar a movimentação do robô pela arena de batalha.

Métodos:

run – É o método principal em todos os robôs. Aqui iremos definir o comportamento do nosso robô.

Sintaxe:

public void run()

ahead - Move seu robô para frente. Essa chamada é executada imediatamente e não retorna até que seja concluída. Se o robô colidir com uma parede, o movimento está completo. Se o robô colidir com outro robô, o movimento estará completo se você estiver indo em direção ao outro robô.

Sintaxe:

public void ahead(double distance)

A variável distance é um parâmetro com tipo de dado definido como double. É o parâmetro para definir quanto ele irá para frente.
Exemplo de chamada de método dentro do nosso robô **Tanque.java** criando anteriormente:
package meustanques;

```
import robocode.*;

import java.awt.Color;

public class Tanque extends Robot
{
	public void run() {
	    setColors(Color.YELLOW,Color.BLUE,Color.RED,Color.BLACK,Color.GREEN);
	    ahead(100);
	}
}
```

Porém, a chamada ao método **ahead(100)** faz com que o nosso robozinho se mova para frente por 100 pixels e pare. E vai ficar parado até o fim da batalha. Para que ele fique em constante movimento, precisamos definir uma **estrutura de repetição**. Aqui utilizaremos uma estrutura de repetição **while()**. O while executa as instruções que estão dentro do seu bloco enquanto a condição definida em seu parâmetro for verdadeira. No nosso caso utilizaremos o parâmetro **true**, ou seja, enquanto o robozinho estiver rodando ele irá se mover 100 pixels para frente. Porém quando ele bater na parede irá ficar parado por não ter mais como ir para frente.

```
package meustanques;
import robocode.*;
import java.awt.Color;
public class Tanque extends Robot
{
	public void run() {
	    setColors(Color.YELLOW,Color.BLUE,Color.RED,Color.BLACK,Color.GREEN);
	    while(true)
	        ahead(100);
	}
}
```

back - Move seu robô para trás. Essa chamada é executada imediatamente e não retorna até que seja concluída. Se o robô colidir com uma parede, o movimento está completo. Se o robô colidir com outro robô, o movimento estará completo se você estiver indo em direção ao outro robô.

Sintaxe:

public void back(double distance)

A variável distance é um parâmetro com tipo de dado definido como double. É o parâmetro para definir quanto ele irá para trás. Se você só acrescentar a chamada do método back() ao nosso robozinho irá perceber que ele irá para frente e para trás, mas seguirá até colidir com uma parede ou outro robô. Se o parâmetro for pequeno, ele ficará parado até acabar a batalha, se colocar um parâmetro maior, ficará indo para frente e para trás até acabar toda sua energia.

Uma observação importante quando trabalhamos com estruturas de repetição é que se tivermos mais de uma instrução dentro da estrutura, devemos colocar **{** (abre chave) logo após a definição do parâmetro e **}** (fecha chave) assim que colocarmos todas as instruções.

```
package meustanques;
import robocode.*;
import java.awt.Color;

public class Tanque extends Robot
{
	public void run() {

		setColors(Color.YELLOW,Color.BLUE,Color.RED,Color.BLACK,Color.GREEN);
		while(true){
			ahead(100);
			back(500);
		}
	}
}
```

turnLeft - Gira seu robô para a esquerda. Essa chamada é executada imediatamente e não retorna até que seja concluída. A arma e o radar girarão a mesma quantidade, pois estão presos ao robô. O parâmetro passado será em graus.

Sintaxe:

public void turnRight(double degrees)

Exemplo:

```
package meustanques;
import robocode.*;
import java.awt.Color;

public class Tanque extends Robot
{

	public void run() {

		setColors(Color.YELLOW,Color.BLUE,Color.RED,Color.BLACK,Color.GREEN);
		while(true){
			ahead(100);
			back(500);
			turnLeft(45);
		}
	}
}
```

turnRight - Gira seu robô para a direita. Essa chamada é executada imediatamente e não retorna até que seja concluída. A arma e o radar girarão a mesma quantidade, pois estão presos ao robô. O parâmetro passado será em graus.

Sintaxe:

```
public void turnLeft(double degrees)
```

Exemplo:

```
package meustanques;
import robocode.*;
import java.awt.Color;

public class Tanque extends Robot
{
```

```
        public void run() {

            setColors(Color.YELLOW,Color.BLUE,Color.RED,Color.BLACK,Color.GREEN);
            while(true){
                        ahead(100);
                        back(500);
                        turnLeft(45);
                        turnRight(180);
             }
        }
}
```

Se você usou os mesmos parâmetros do exemplo no seu robô Tanque.java, irá perceber que agora nosso robô além de estar colorido, também consegue se mover pela arena sem ficar travado na parede.

Figura 2 – Tela da batalha
Fonte: do autor

No próximo material vamos trazer alguns métodos para que possamos utilizar no canhão do nosso tanque exemplo.

5 Referências

Caellum. Java: Orientação a objetos. Disponível em: <https://www.caelum.com.br/apostila-java-orientacao-objetos> . Acesso em 15/05/2019.

Computer Science. **Class.** Disponível em: <http://www.cs.xu.edu/csci170/02s/robocode/javadoc/>. Acesso em 05/05/2019.

DevMedia. **Convenções de código Java**. Disponível em:<https://www.devmedia.com.br/convencoes-de-codigo-java/23871>. Acesso em 31/05/2019