Administração Central

Unidade do Ensino Médio e Técnico – Cetec Capacitações

ROBÓTICA Paula Souza

2019
São Paulo

www.centropaulasouza.sp.gov.br
Rua dos Andradas, 140 • Santa Ifigênia • 01208-000 • São Paulo • SP • Tel.: (11) 3324.3300

Material Didático sobre Robocode
Trabalhando com Alguns Métodos e Eventos

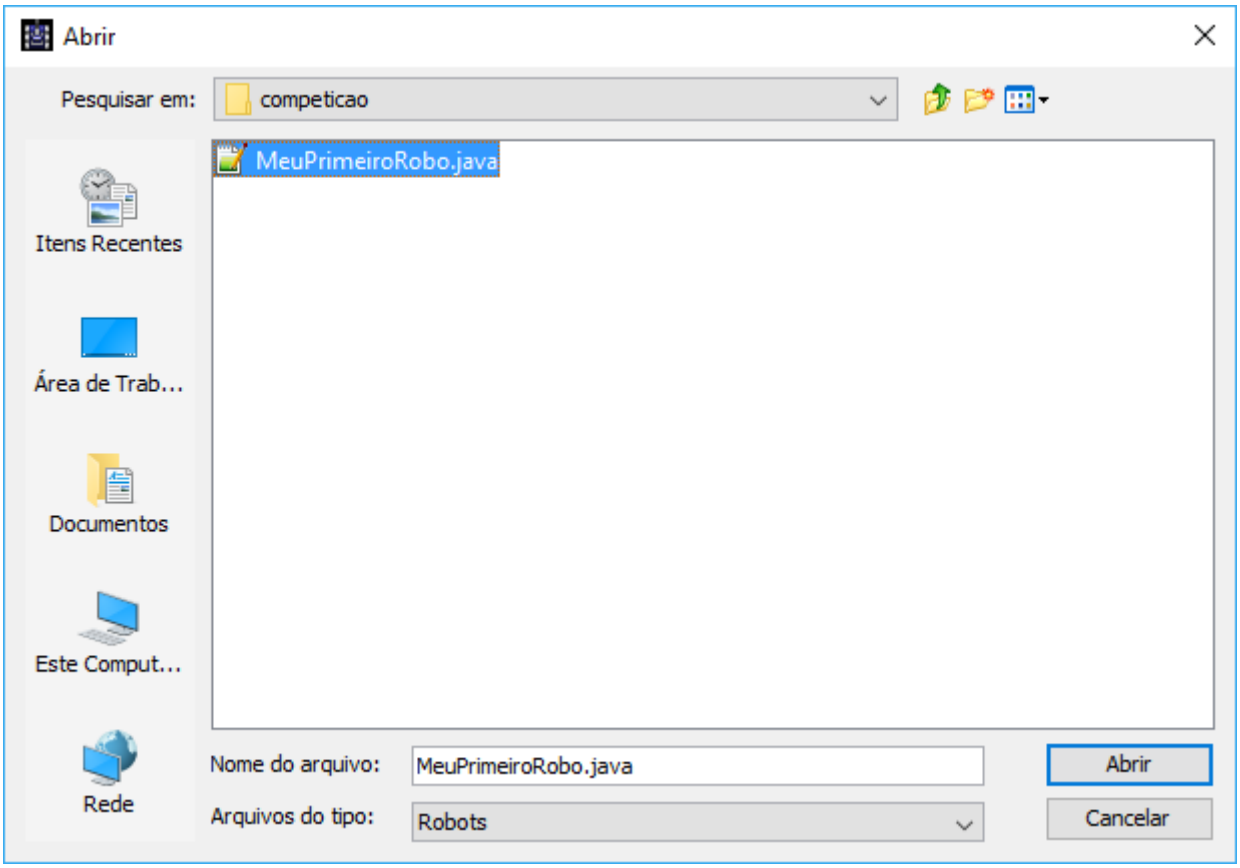
1 Modificando nosso primeiro Robô

Através de Métodos, é possível realizar modificações nos robôs. A seguir teremos alguns exemplos.

1.1 Abrindo o Robô

Para abrir o primeiro robô que criamos, vamos selecionar no menu **Robot** e depois clicar em **Source Editor**.

Na janela do editor, vamos clicar em **File**, depois em **Open**, abrir a pasta **competicao**, selecionar o **MeuPrimeiroRobo.java** e clicar em **Abrir**.



Então o código aparecerá código-fonte do robô para que possamos edita-lo.

1.2 Mudando a cor

Importando a Classe Cor:

```
1  package competicao;
2  import robocode.*;
3  import java.awt.Color;
```

Habilitar esta linha retirando as barras de comentário de seu início

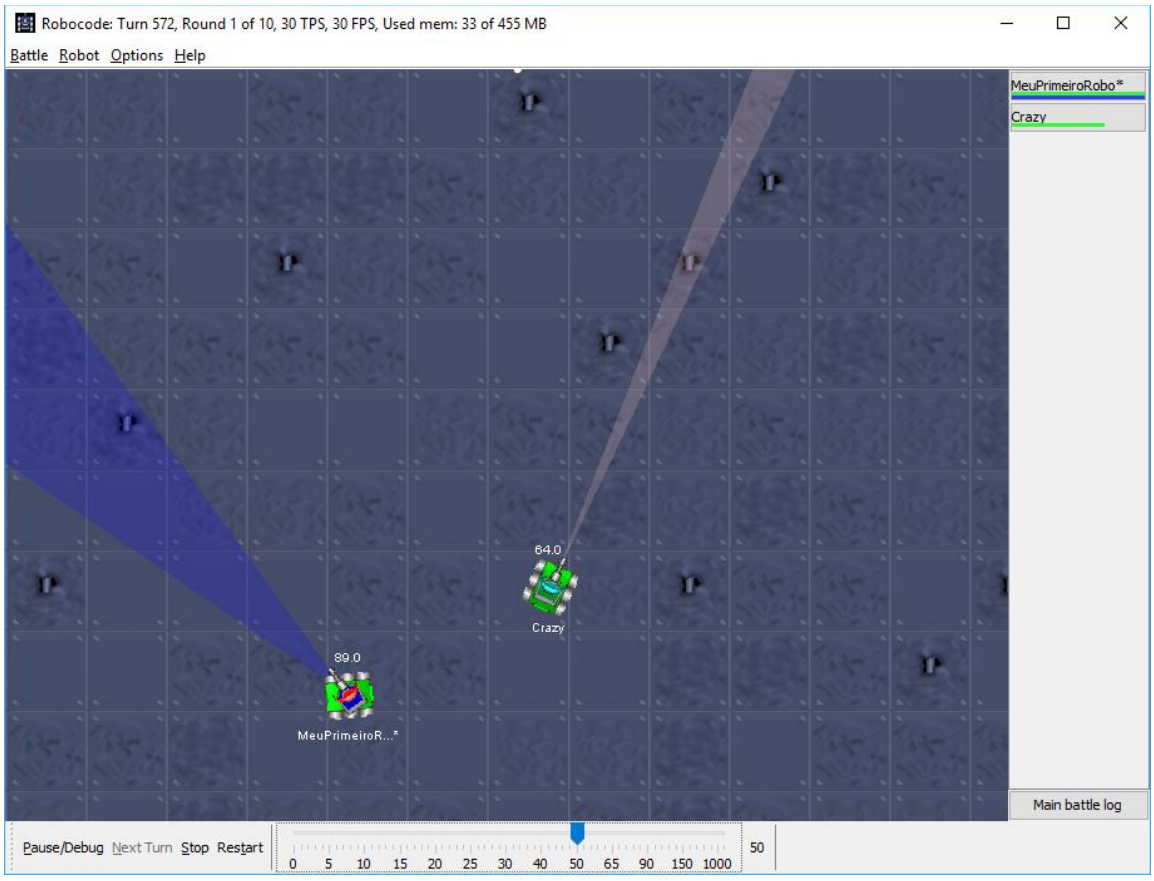
A seguir estão sendo definidas algumas cores para alguns elementos:

```
21          // setColors(Color.red,Color.blue,Color.green); // body,gun,radar
22          setBodyColor(Color.green);
23          setRadarColor(Color.red);
24          setGunColor(Color.blue);
```

Código que deve ser inserido

O tanque não ficará bonito, mas servirá para observar as mudanças. Toda vez que alteramos o código, é necessário salvar e compilar novamente.

Observe a cor modificada do tanque:



3 Fazendo alguns testes

Faça seu robô andar em um quadrado. Para isso mude o código para:

```
while(true) {
    // Replace the next 4 
    ahead(100);
    turnRight(90);
}
```

Compile e execute para ver o resultado.

Agora, vamos editar o robô para fazer curvas. Para isso altere o código:

01) Precisamos mudar o extends de nosso robô para

```
11 public class MeuPrimeiroRobo extends AdvancedRobot
12 {
```

02) Faça as mudanças propostas 1, 2, 3, 4 e 5. Uma sugestão seria fazer uma alteração de cada vez e a cada mudança compile e execute para perceber as modificações:

```
/*Ajusta o Radar com o canhão. Teste o comando abaixo com true e false
 * verá que no true o radar se movimenta em posição diferente a do canhão.
 */
1 setAdjustRadarForGunTurn(true);

// Robot main loop
while(true) {
    // Replace the next 4 lines with any behavior you would like

    //vai para frente
2   setAhead(100);
    //vira para a direita 90°
3   setTurnRight(90);

    //atira com força 2
4   setFire(2);
    //somente executa os comando quando for chamado
5   execute();
}
```

4 Listas de Métodos

Movimentação – Classe Robot

Método	Parâmetro	Descrição
ahead(*double* **)**	a distância que o robô deverá percorrer.	Movimenta o robô para frente, uma distância x dada por parâmetro. Se o robô bater em outro, ou na parede antes de completar a distância desejada o método é interrompido.
back(*double* **)**	a distância que o robô deverá percorrer.	Semelhante ao método anterior, a única diferença é que o robô move para traz.
turnRight(*double* **)**	o ângulo em graus que o robô deverá girar.	Gira o robô para a direita (sentido horário).
turnLeft(*double* **)**	o ângulo em graus que o robô deverá girar.	Gira o robô para a esquerda (sentido anti-horário).
turnGunRigth(*double* **)**	o ângulo em graus que o canhão deverá girar	Gira o canhão para a direita.
turnGunLeft(*double* **)**	o ângulo em graus que o canhão deverá girar	Gira o canhão para a esquerda.
turnRadarRigth(*double* **)**	o ângulo em graus que o radar deverá girar	Gira o radar para a direita.
turnRadarLeft(*double* **)**	o ângulo em graus que o radar deverá girar	Gira o radar para a esquerda.

Movimentação – Classe AdvancedRobot

Os comandos da classe AdvancedRobot que começam com "set" eles funcionam como os herdados da classe Robot. A diferença é que enquanto o método está sendo executado ele continua executando as linhas de comando abaixo. Com isso é possível misturar movimentos. Por exemplo, se tiver: **turnRight(90);**

O robô irá andar para frente e depois que tiver terminado de percorrer a distância 100, ele girará 90º. Mas se tiver: **setTurnRight(90);**

O robô andará para frente e girará 90º ao mesmo tempo, fazendo uma curva.

Método	Parâmetro	Descrição
setAhead(*double* **)**	a distância que o robô deverá percorrer.	Herdado do método ahead.
setBack(*double* **)**	a distância que o robô deverá percorrer.	Herdado do método back.
setTurnRight(*double* **)**	o ângulo em graus que o robô deverá girar.	Herdado do método turnRight.
setTurnLeft(*double* **)**	o ângulo em graus que o robô deverá girar.	Herdado do método turnLetf.
setTurnGunRigth(*double* **)**	o ângulo em graus que o canhão deverá girar	Herdado do método turnGunRigth**.**
setTurnGunLeft(*double* **)**	o ângulo em graus que o canhão deverá girar	Herdado do método turnGunLeft.
setTurnRadarRigth(*double* **)**	o ângulo em graus que o radar deverá girar	Herdado do método turnRadarRigth.
setTurnRadarLeft(*double* **)**	o ângulo em graus que o radar deverá girar	Herdado do método turnRadarLeft.

Tiro – Classe Robot

Método	Parâmetro	Descrição
fire(*double* **)**	a força do tiro, e subtraído da energia de seu robô.	Atira imediatamente na força mandada por parâmetro, de 0.1 até 3. Se mandar um tiro maior que 3 ele considera força 3.
fireBullet(*double* **)**	a força do tiro, e subtraído da energia de seu robô.	A diferença do método anterior é que ele é uma função e retorna um valor do tipo *Bullet,* além disso, manda outro tiro em seguida, este com mais velocidade, se o primeiro tiro tiver boas possibilidades de acertar.

Tiro – Classe AdvancedRobot

Método	Parâmetro	Descrição
setFire(double **)**	a força do tiro, e subtraido da energia de seu robô.	Herdado do método fire.
setFireBullet(double **)**	a força do tiro, e subtraido da energia de seu robô.	Herdado do método fireBullet.

Envia dados para o Robô

Método	Parâmetro	Descrição
setAdjustGunForRobotTurn(*boolean* **)**		
setAdjustRadarForGunTurn(*boolean* **)**		
setColors(*Color, Color, Color* **)**	a cor do robô, a cor do canhão, a cor do radar, nesta ordem.	Atribui as cores do robô.

Retorna Dados do Rôbo

Método	Tipo do Retorno	Descrição do Retorno
getName()	String	Retorna o nome do robô.
getEnergy()	double	Retorna a energia corrente do robô.
getX()	double	A posição X (eixo horizontal) do robô na arena de batalha. Quando 0 (zero) ele estará encostado no lado esquerdo.
getY()	double	A posição Y (eixo vertical) do robô na arena de batalha. Quando 0 (zero) ele estará encostado na parte de baixo.
getWidth()	double	Retorna a largura do robô.
getHeight()	double	Retorna a altura do robô.
getHeading()	double	Retorna o ângulo em graus (de 0 até 360) que o robô está virado. Se retornar 0(zero) ele está virado para a esquerda, se retornar 90 ele está voltado para cima.
getGunHeading()	double	Retorna o ângulo em graus que o canhão está virado. Como no método anterior.
getRadarHeading()	double	Retorna o ângulo em graus que o radar está virado.
getGunCoolingRate()	double	
getGunHeat()	double	Retorna quanto o canhão está virando no momento corrente.
getVelocity()	double	Retorna a velocidade do robô.

Retorna Dados da Batalha

Métodos	Tipo do Retorno	Retorno
getOthers()	int	Retorna o total de oponentes ainda vivos no round.
getBattleFieldHeight()	double	Retorna a altura da arena de batalha.
getBattleFieldWidth()	double	Retorna a largura da arena de batalha.
getNumRounds()	int	Retorna o total de rounds da batalha.
getRoundNum()	int	Retorna o número do round corrente.
getTime()	long	Retorna o tempo do round.

5 Alguns Eventos

É possível programar os robôs utilizando Eventos. Seguem alguns exemplos.

5.1 Evento run()

O Evento run(), é o evento básico no Robocode®, por isso que quando criamos um robô, ele já está criado no código fonte:

```
public void run() {

    while(true) {
        ahead(100);
        turnGunRight(360);
        back(100);
        turnGunRight(360);
    }
}
```

Este evento é o básico para fazer com que o tanque ande para frente, gire, atire e retorne.

5.2 Evento onScannedRobot()

O Evento onScannedRobot() é o evento que verifica se há algum robô no scanner, se tiver ele atira:

```
public void onScannedRobot(ScannedRobotEvent e) {
    fire(1);
}
```

5.3 Evento onHitByBullet()

O Evento onHitByBullet() é o evento que verifica se levou algum tiro, se levar ele recua 10:

```
public void onHitByBullet(HitByBulletEvent e) {
        back(10);
}
```

5.4 Evento onHitWall()

O Evento onHitWall () é o evento que verifica se bateu na parede, se bater ele recua 20:

```
public void onHitWall(HitWallEvent e) {
        back(20);
}
```

Obs:- Sobre as mensagens que iremos colocar para os eventos: Caso, no ambiente de batalha não conseguir visualizar a mensagem, basta clicar sobre o nome do Robô na tela da Batalha.

5.5 Evento onHitRobot()

O Evento onHitRobot() é o evento que verifica bateu em outro robô, se bater ele envia uma mensagem:

```
public void onHitRobot(HitRobotEvent e){
    System.out.println("Choquei com outro Robô");
}
```

5.6 Evento onBattleEnded()

O Evento onBattleEnded() é o evento que verifica se a batalha chegou ao fim, se chegar ele envia uma mensagem:

```
public void onBattleEnded(BattleEndedEvent e){
    System.out.println("Acabou");
}
```

5.7 Evento onBattleEnded()

O Evento onBulletHitBullet() é o evento que verifica se o tiro dado acertou outro tiro, se acertou ele envia uma mensagem:

```
public void onBulletHitBullet(BulletHitBulletEvent e){
    System.out.println("Acertei outro tiro");
}
```

5.8 Evento onBulletHit()

O Evento onBulletHit() é o evento que verifica se o tiro dado acertou o alvo, se acertou ele envia uma mensagem:

```
public void onBulletHit(BulletHitEvent e){
    System.out.println("Acertei um tiro");
}
```

5.9 Evento onBulletMissed()

O Evento onBulletMissed() é o evento que verifica se o tiro dado errou o alvo, se errou ele envia uma mensagem:

```
public void onBulletMissed(BulletMissedEvent e){
    System.out.println("Errei o tiro");
}
```

5.10 Evento onDeath()

O Evento onDeath() é o evento que verifica se acabou a partida para o nosso robô, se acabou ele envia uma mensagem:

```
public void onDeath(DeathEvent e){
    System.out.println("Morri");
}
```

5.11 Evento onRobotDeath()

O Evento onRobotDeath() é o evento que verifica se algum concorrente morreu, se morreu ele envia uma mensagem:

```
public void onRobotDeath(RobotDeathEvent e){
    System.out.println("Um concorrente meu morreu");
}
```

5.12 Evento onRoundEnded()

O Evento onRoundEnded() é o evento que verifica se o round acabou, se acabou ele envia uma mensagem:

```
public void onRoundEnded(RoundEndedEvent e){
    System.out.println("O Round Acabou agora");
}
```

5.13 Evento onWin()

O Evento onWin() é o evento que verifica se o robô ganhou:

```
public void onWin(WinEvent e){
    System.out.println("Ganhei");
}
```

6 Referência

[1] Autor Desconhecido. **GSIGMA – Universidade Federal de Santa Catarina**. Disponível em: http://www.gsigma.ufsc.br/~popov/aulas/robocode/metodos.html Acesso em 12 de março de 2019.

[2] Autor Desconhecido. **Site Oficial ROBOCODE**. Disponível em: http://robocode.sourceforge.net/ Acesso em 10 de março de 2019.