

Workshop

BATALHAS VIRTUAIS COM

ROBO{CODE}



Ensino fora dos
livros

Aprendizagem
Baseada por
Desafios

Aprendizagem
Baseada em
Projetos

Foco na
construção de
conhecimento

Desenvolvimento
de habilidades

Integração de
conhecimentos

Formação de
profissionais



Ações do Projeto



ROBOCODE

ROBÓTICA PAULA SOUZA



11º TORNEIO DE ROBOCODE - 2023

ROBOCODE

2013	06 professores 15 alunos 04 Etecs	2014	77 professores 1213 alunos 43 Etecs	2015	79 professores 564 alunos 65 Etecs 01 Fatec	2019	37 professores 744 alunos 29 Etecs 01 Fatec	2021	14 professores 172 alunos 16 Etecs
2016	125 professores 1302 alunos 117 Etecs 08 Fatecs	2017	67 professores 936 alunos 33 Etecs 03 Fatecs	2018	36 professores 888 alunos 28 Etecs 04 Fatecs	2020	10 professores 87 alunos 09 Etecs 01 Fatec		

2022


22
ETECS


04
FATECS


483
ALUNOS


33
PROFESSORES


184
EQUIPES



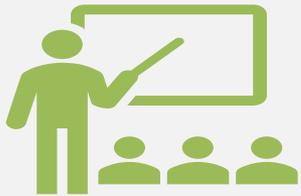
***Construa o melhor,
destrua o resto!***

ROBOCODE

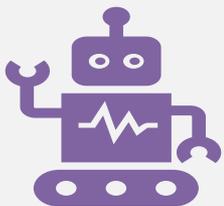
ROBÓTICA PAULA SOUZA



É uma ferramenta (jogo) de programação de cunho educativo



Usado como ferramenta no processo de ensino e de aprendizagem em programação e inteligência artificial



O usuário desenvolve um robô (tanque de guerra) para lutar com outros robôs numa arena virtual

Por onde
começar?

ROBOCODE
ROBÓTICA PAULA SOUZA

Requisitos:



Baixar e instalar o Java JDK



Baixar e instalar o ROBOCODE

Página Oficial Robocode:

<https://robocode.sourceforge.io/>

Robocoding

- **Java** é necessário para executar o Robocode. De preferência, um Java Developer Kit (JDK) versão 12 a 18 ou mais recente
- **Baixe** o Robocode do SourceForge. Alternativamente a partir de **versões do GitHub**
- **ReadMe** for Robocode, que oferece uma boa visão geral do Robocode
- **Introdução** - Introdução ao Robocode no RoboWiki. Use o **Web Archive para Robocode** se o RoboWiki estiver inoperante!
- **API do Robocode** - API do Robô e API de Controle
- **Meu Primeiro Robô** - tutorial sobre como criar seu primeiro robô
- **FAQ** - Perguntas Frequentes sobre o Robocode



Baixar e instalar o Java JDK:

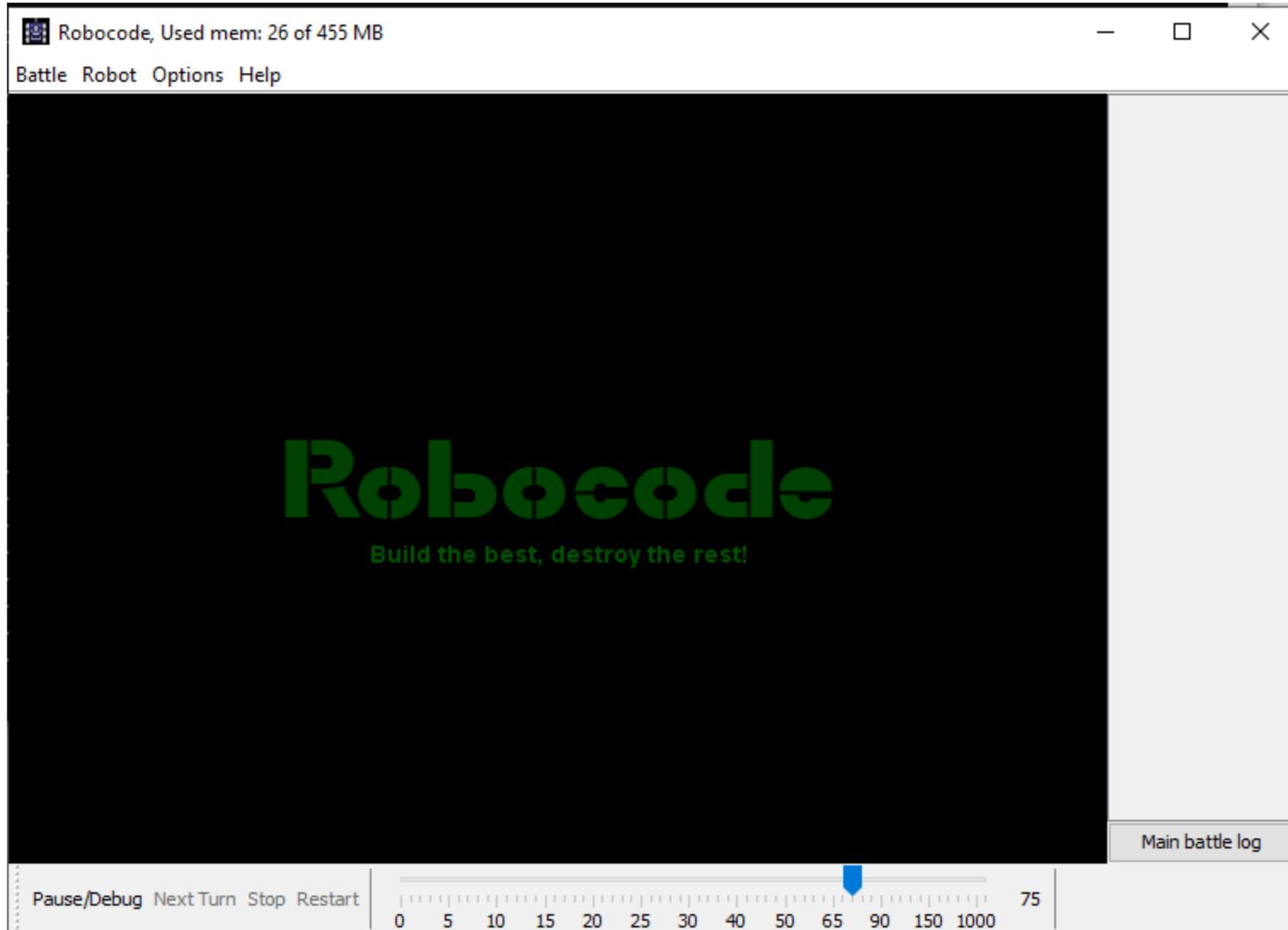
<https://www.oracle.com/java/technologies/downloads/> e
escolha a versão do Java para instalação de acordo com o seu
Sistema Operacional.



Baixar e instalar o ROBOCODE:

<https://sourceforge.net/projects/robocode/files/> e
escolha a ultima versão

Ambiente de desenvolvimento pronto!



Menu: BATALHA

Robocode (paused), Used mem: 30 of 455 MB

Battle Robot Options Help

New	Ctrl+N
Open	Ctrl+O
Save	Ctrl+S
Save As	Ctrl+Shift+S
Open Record	Ctrl+Shift+O
Save Record	Ctrl+R
Import XML Record	Ctrl+I
Export XML Record	Ctrl+X
Take Screenshot	Ctrl+T
Exit	

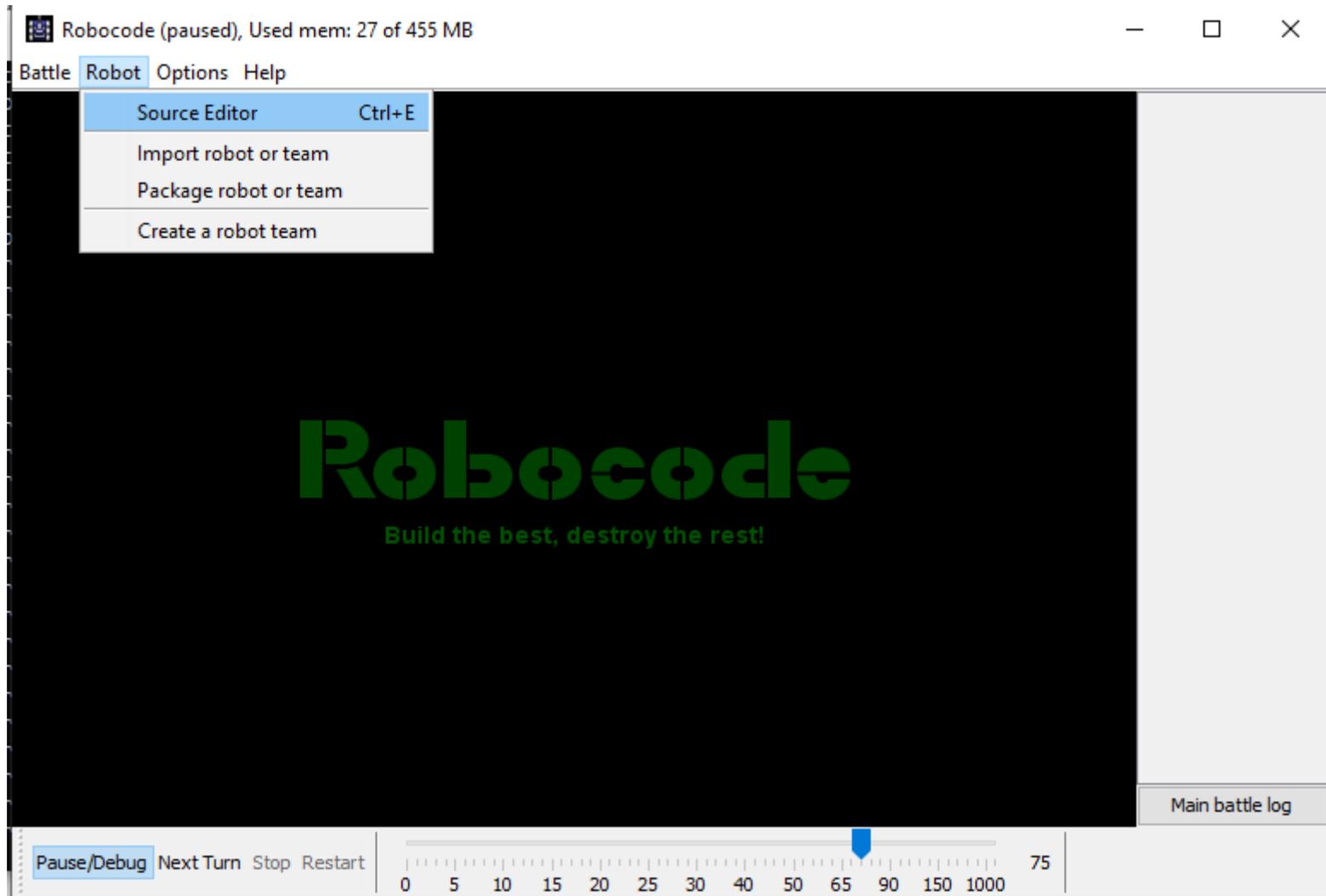
robocode
Build the best, destroy the rest!

Main battle log

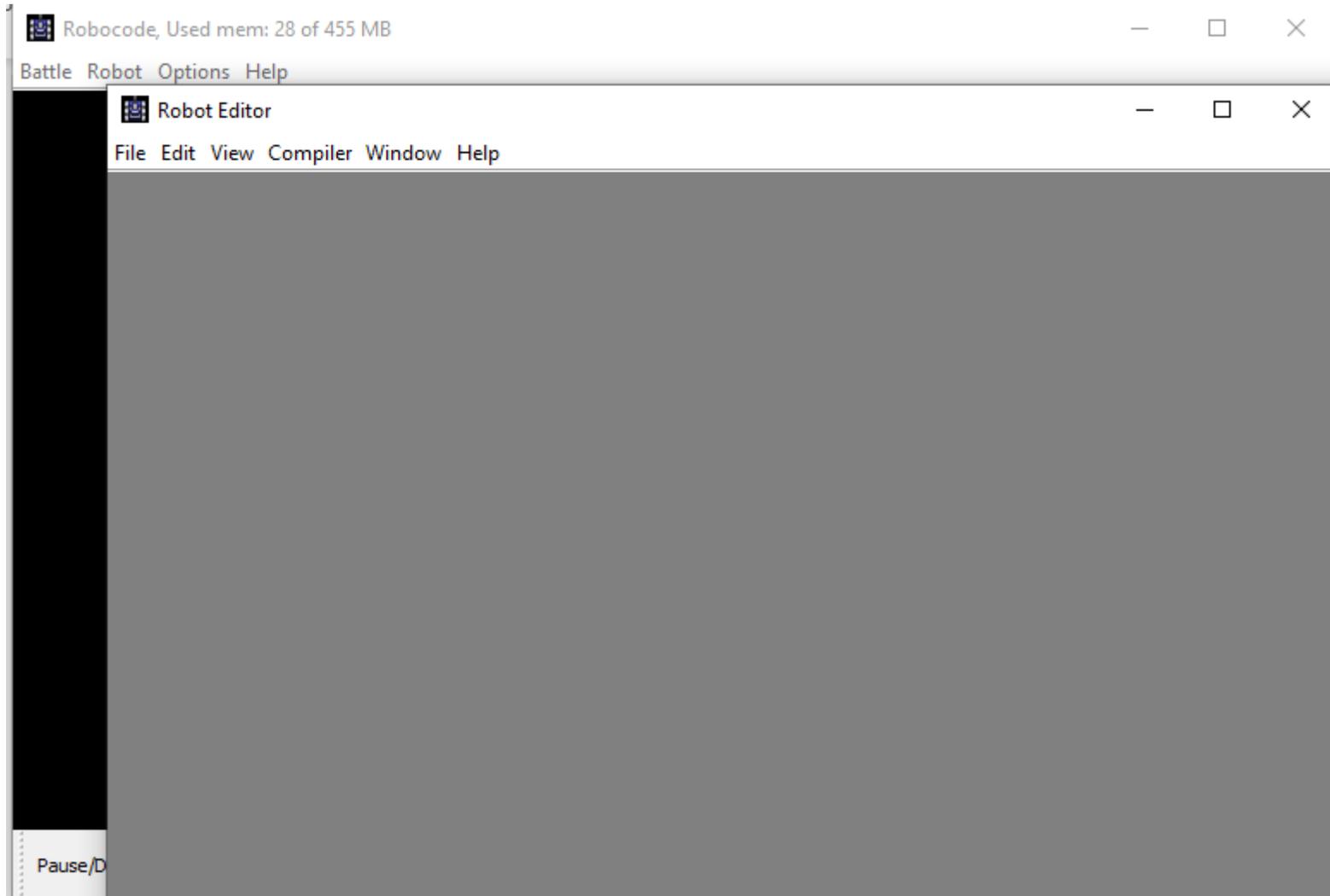
Pause/Debug Next Turn Stop Restart

0 5 10 15 20 25 30 40 50 65 90 150 1000 75

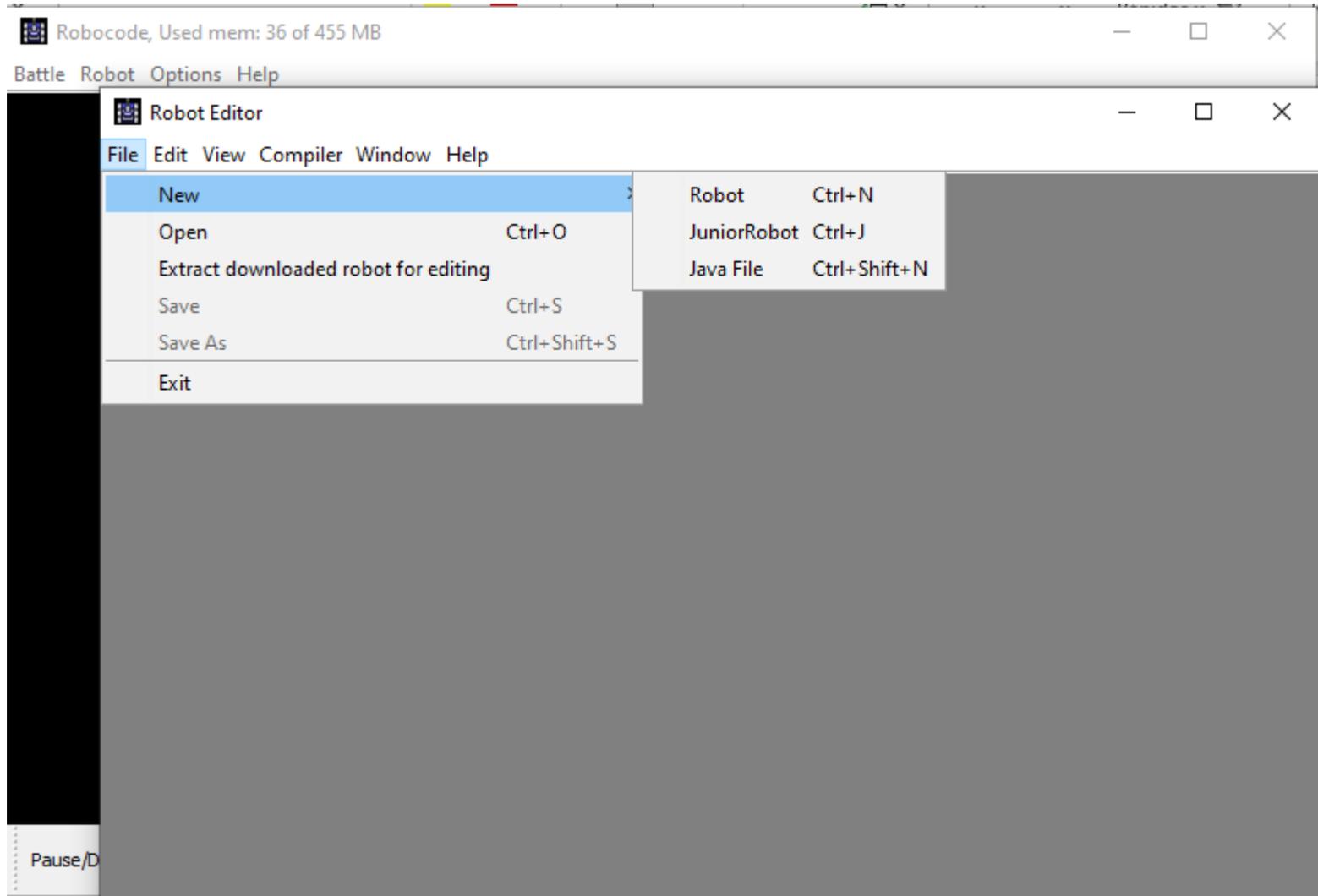
Menu: ROBOT



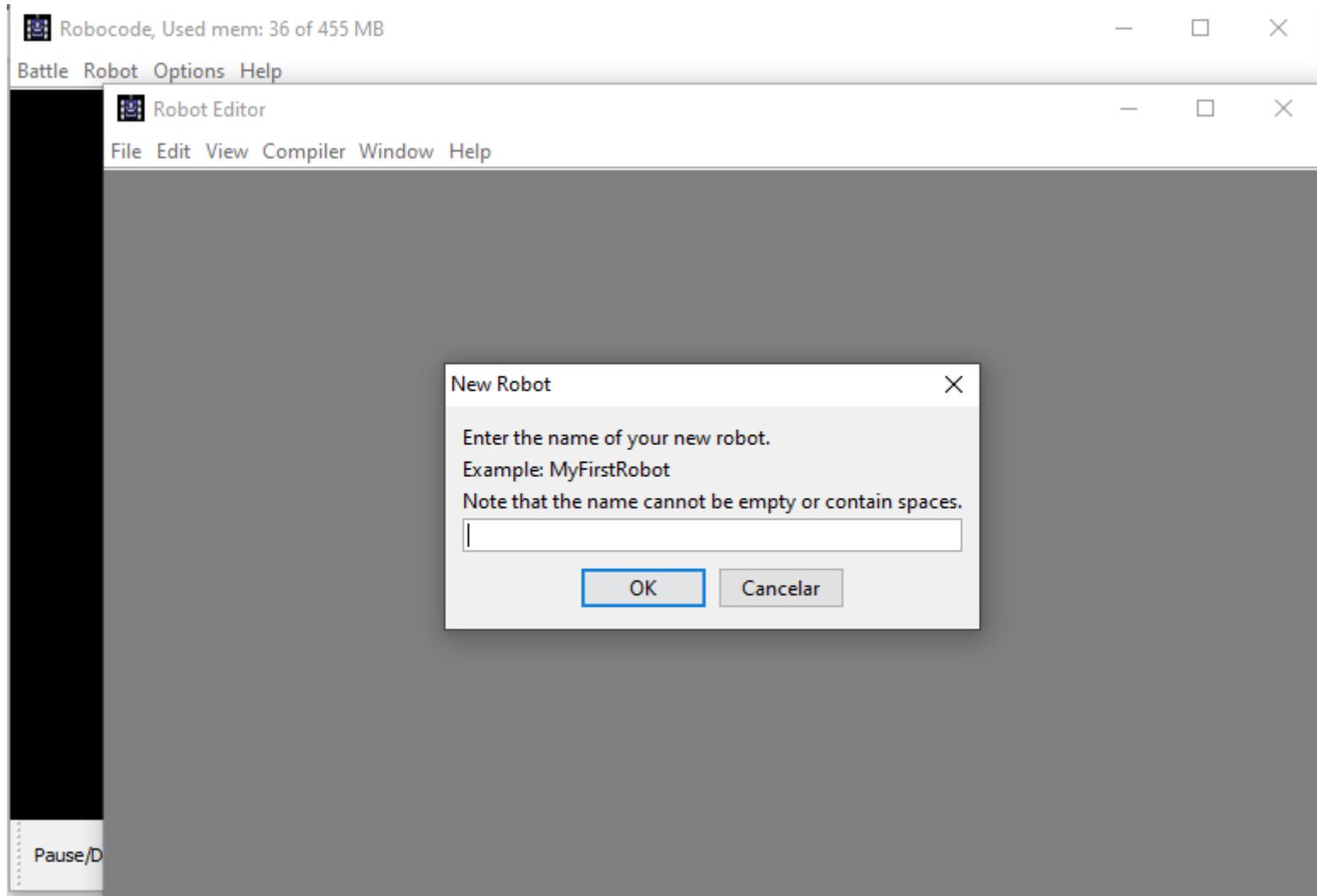
SOURCE EDITOR



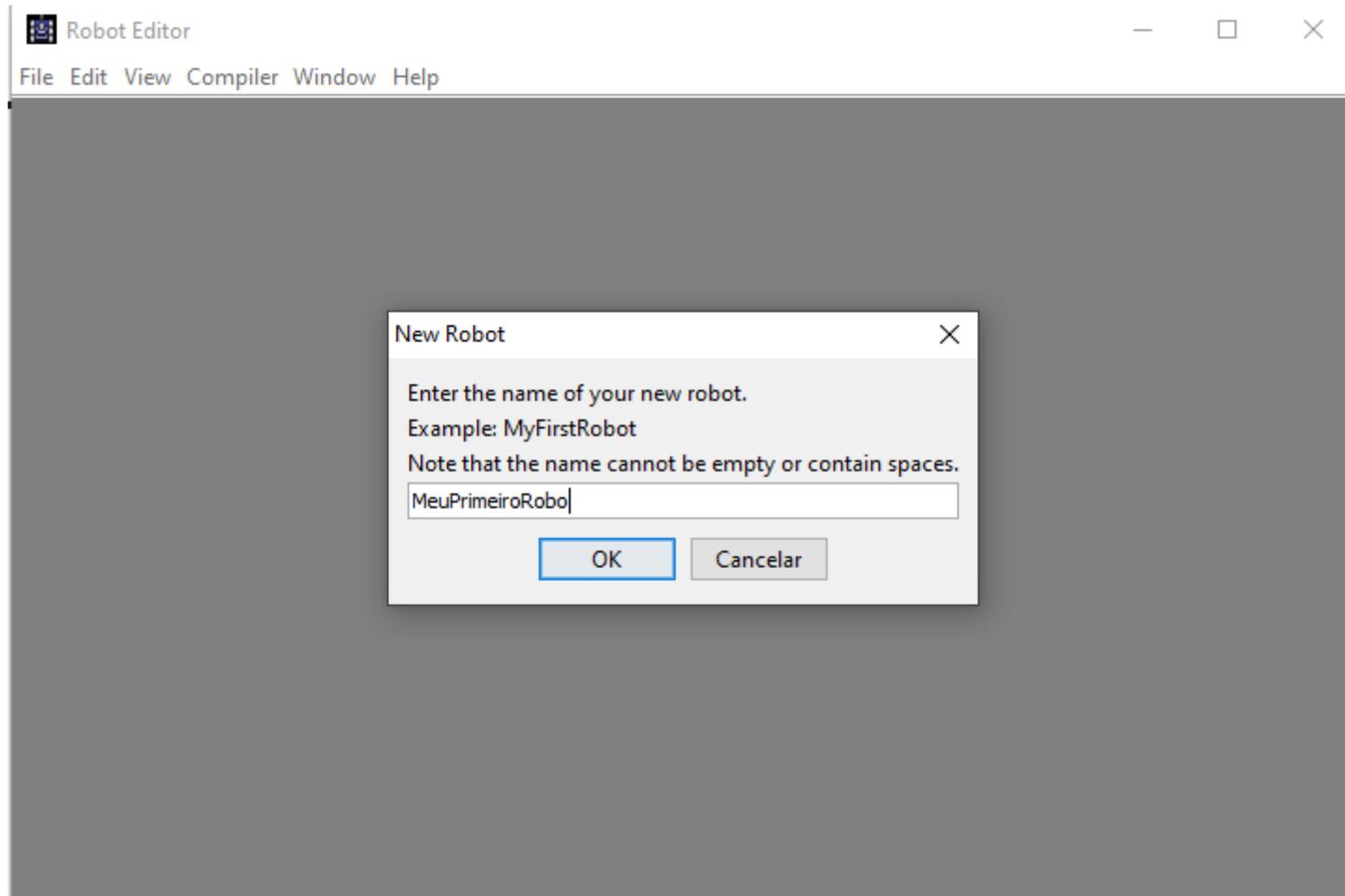
SOURCE EDITOR: FILE



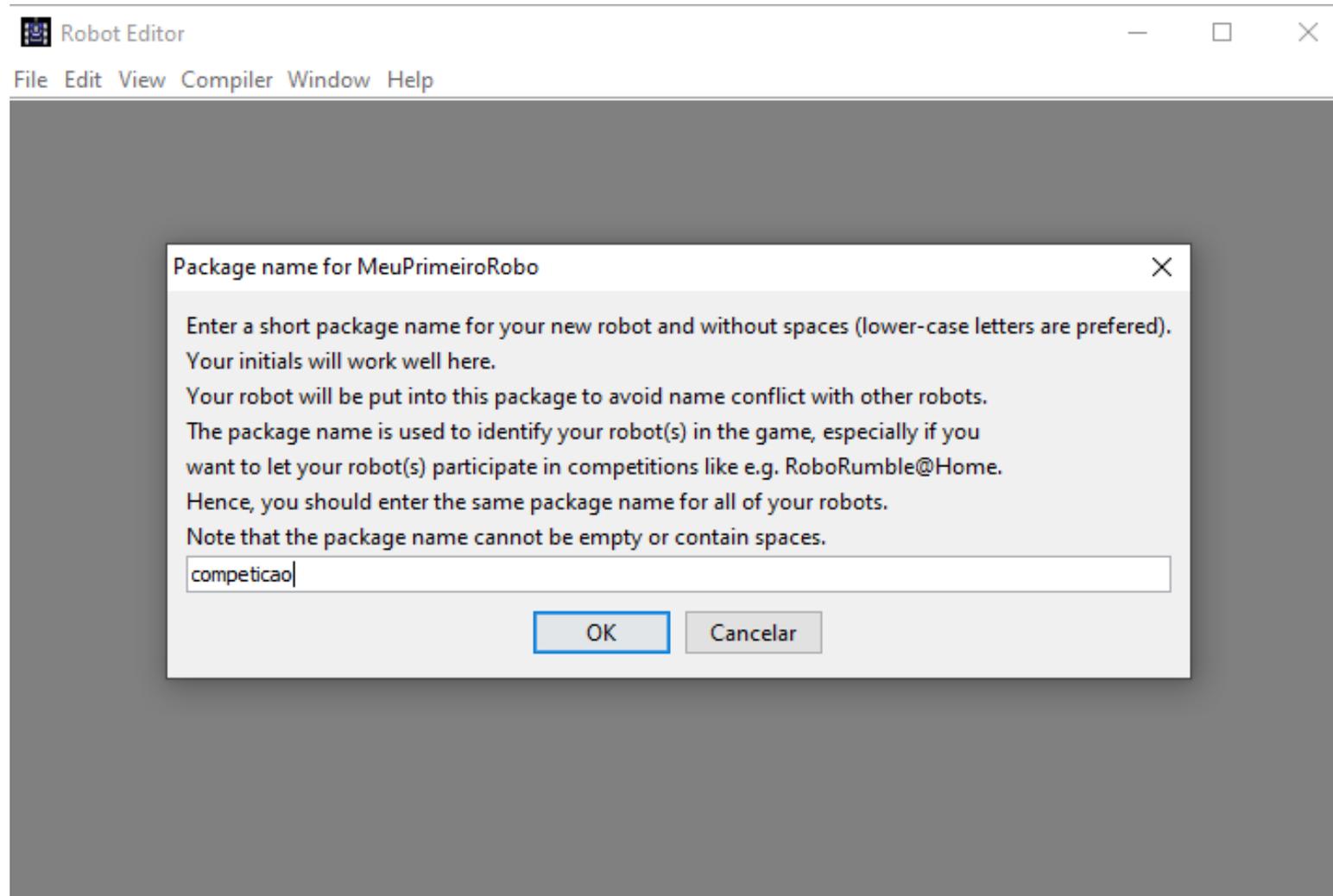
SOURCE EDITOR: FILE → ROBOT



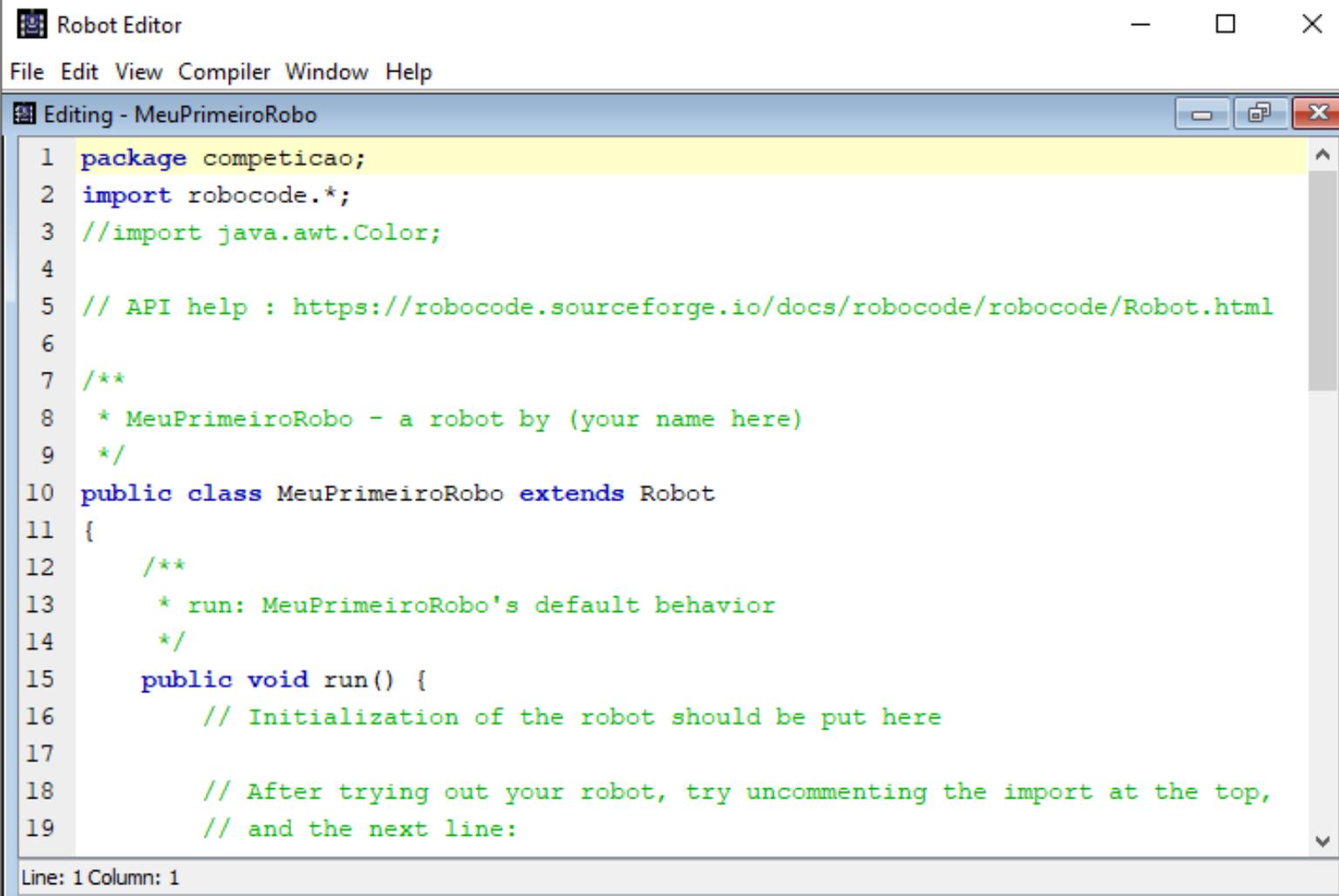
Criando o Robô: Nomeando



Criando o Robô: package



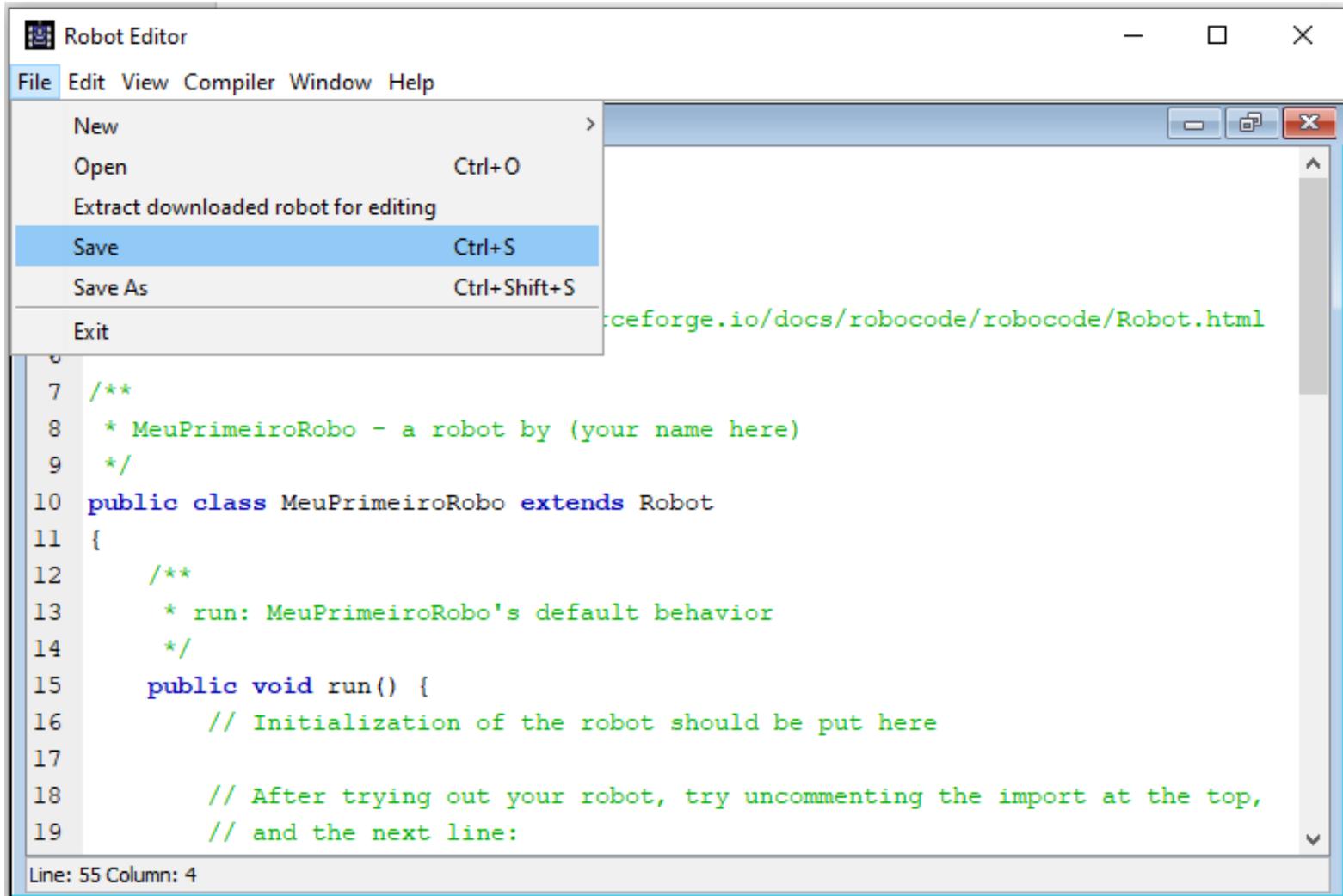
Criando o Robô: Desenvolvendo



```
Robot Editor
File Edit View Compiler Window Help
Editing - MeuPrimeiroRobo
1 package competicao;
2 import robocode.*;
3 //import java.awt.Color;
4
5 // API help : https://robocode.sourceforge.io/docs/robocode/robocode/Robot.html
6
7 /**
8  * MeuPrimeiroRobo - a robot by (your name here)
9  */
10 public class MeuPrimeiroRobo extends Robot
11 {
12     /**
13      * run: MeuPrimeiroRobo's default behavior
14      */
15     public void run() {
16         // Initialization of the robot should be put here
17
18         // After trying out your robot, try uncommenting the import at the top,
19         // and the next line:
```

Line: 1 Column: 1

Criando o Robô: Salvando



```
Robot Editor
File Edit View Compiler Window Help
New
Open Ctrl+O
Extract downloaded robot for editing
Save Ctrl+S
Save As Ctrl+Shift+S
Exit
ceforge.io/docs/robocode/robocode/Robot.html
7 /**
8  * MeuPrimeiroRobo - a robot by (your name here)
9  */
10 public class MeuPrimeiroRobo extends Robot
11 {
12     /**
13     * run: MeuPrimeiroRobo's default behavior
14     */
15     public void run() {
16         // Initialization of the robot should be put here
17
18         // After trying out your robot, try uncommenting the import at the top,
19         // and the next line:
Line: 55 Column: 4
```

Criando o Robô: Compilando

The image shows the Robot Editor interface with the following elements:

- Robot Editor Window:**
 - Menu: File Edit View **Compiler** Window Help
 - Sub-menu: **Compile Ctrl+B**, Options
 - Code Editor Content:


```

1 package
2 import robocode.*;
3 //import java.awt.Color;
4
5 // API help : https://robocode.sourceforge.net
6
7 /**
8  * MeuPrimeiroRobo - a robot by (your name)
9  */
10 public class MeuPrimeiroRobo extends Robot
11 {
12     /**
13     * run: MeuPrimeiroRobo's default beha
14     */
15     public void run() {
16         // Initialization of the robot sho
17
18         // After trying out your robot, ta
19         // and the next line:
          
```
 - Status Bar: Line: 55 Column: 4
- Compiled successfully. Dialog:**
 - Title: Compiled successfully.
 - Content:


```

[reading java/awt/event/KeyEvent.class]
[reading robocode/RoundEndedEvent.class]
[reading robocode/BattleEndedEvent.class]
[reading robocode/WinEvent.class]
[reading robocode/RobotDeathEvent.class]
[reading robocode/HitRobotEvent.class]
[reading robocode/DeathEvent.class]
[reading robocode/BulletMissedEvent.class]
[reading robocode/BulletHitBulletEvent.class]
[reading robocode/BulletHitEvent.class]
[reading robocode/StatusEvent.class]
[reading java/io/PrintStream.class]
[reading robocode/robotinterfaces/peer/IBasicRobotPeer.class]
[writing competicao\MeuPrimeiroRobo.class - #1]
[completed C:\robocode\robots\competicao\MeuPrimeiroRobo.java - #1/1]
[1 unit compiled]
[1 .class file generated]
          
```
 - Buttons: OK

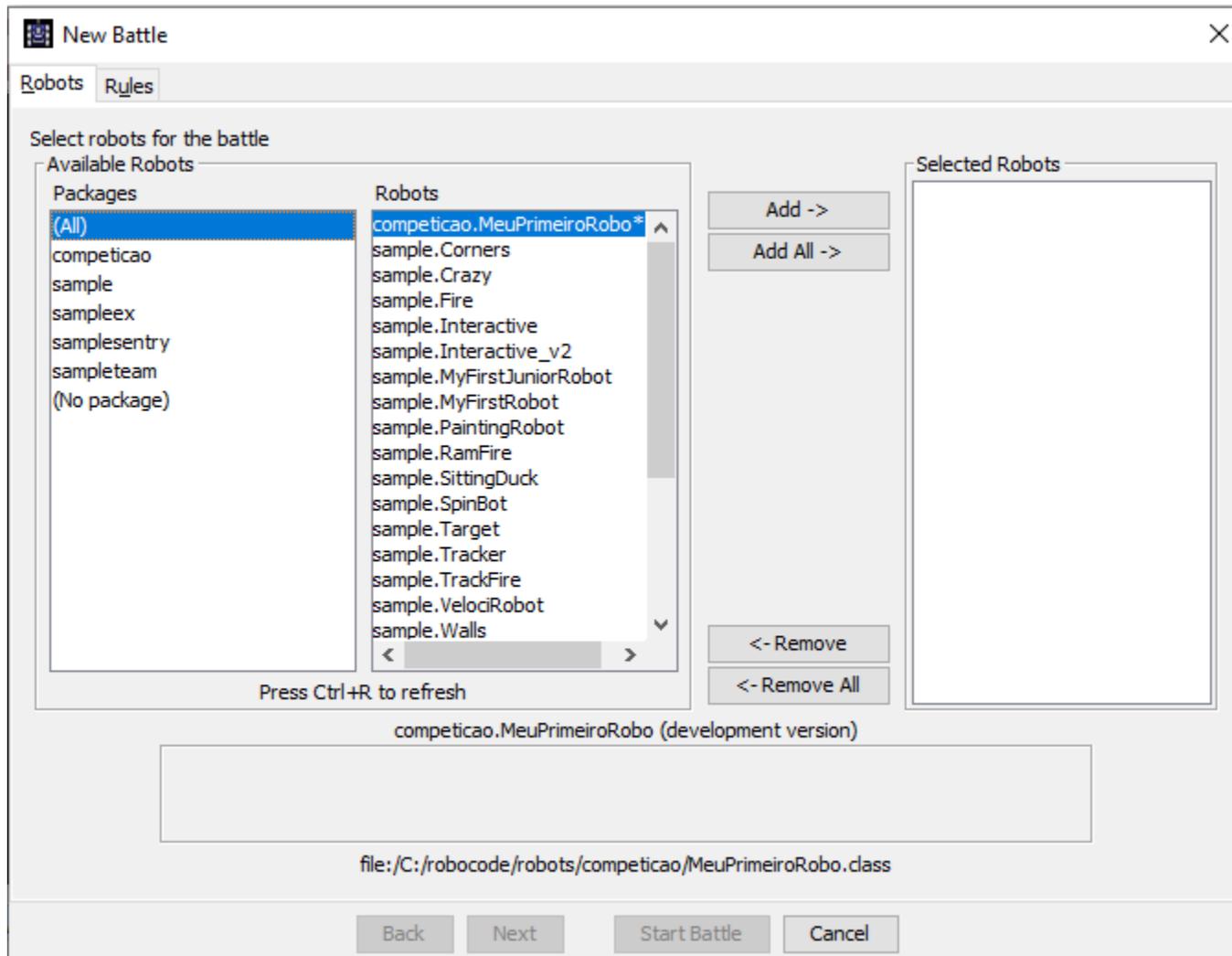
Criando a Batalha

The screenshot shows the Robocode application window titled "Robocode (paused), Used mem: 36 of 455 MB". The "Battle" menu is open, displaying the following options:

- New (Ctrl+N)
- Open (Ctrl+O)
- Save (Ctrl+S)
- Save As (Ctrl+Shift+S)
- Open Record (Ctrl+Shift+O)
- Save Record (Ctrl+R)
- Import XML Record (Ctrl+I)
- Export XML Record (Ctrl+X)
- Take Screenshot (Ctrl+T)
- Exit

The main window area is black with the Robocode logo and the slogan "Build the best, destroy the rest!". At the bottom, there is a control bar with buttons for "Pause/Debug", "Next Turn", "Stop", and "Restart", along with a progress slider ranging from 0 to 75. A "Main battle log" button is visible in the bottom right corner.

Criando a Batalha: Robôs



Criando a Batalha: Seleção de Robôs

New Battle

Robots Rules

Select robots for the battle

Available Robots

Packages

- (All)
- competicao
- sample
- sampleex
- samplesentry
- sampleteam
- (No package)

Robots

- Corners
- Crazy
- Fire
- Interactive
- Interactive_v2
- MyFirstJuniorRobot
- MyFirstRobot
- PaintingRobot
- RamFire
- SittingDuck
- SpinBot
- Target
- Tracker
- TrackFire
- VelociRobot
- Walls

Add ->

Add All ->

Selected Robots

- MeuPrimeiroRobo*
- Crazy

<- Remove

<- Remove All

Press Ctrl+R to refresh

sample.Crazy by Mathew Nelson and Flemming N. Larsen

Built for 1.1.2

A sample robot

Moves around in a crazy pattern

file:/C:/robocode/robots/sample/Crazy.dass

Back Next Start Battle Cancel

Arena de Batalha

Robocode: Turn 116, Round 1 of 3, 74 TPS, 46 FPS, Used mem: 31 of 455 MB

Battle Robot Options Help

MeuPrimeiroRobo* (1)

Crazy (2)

97.0

88.0

MeuPrimeiroR...* (1)

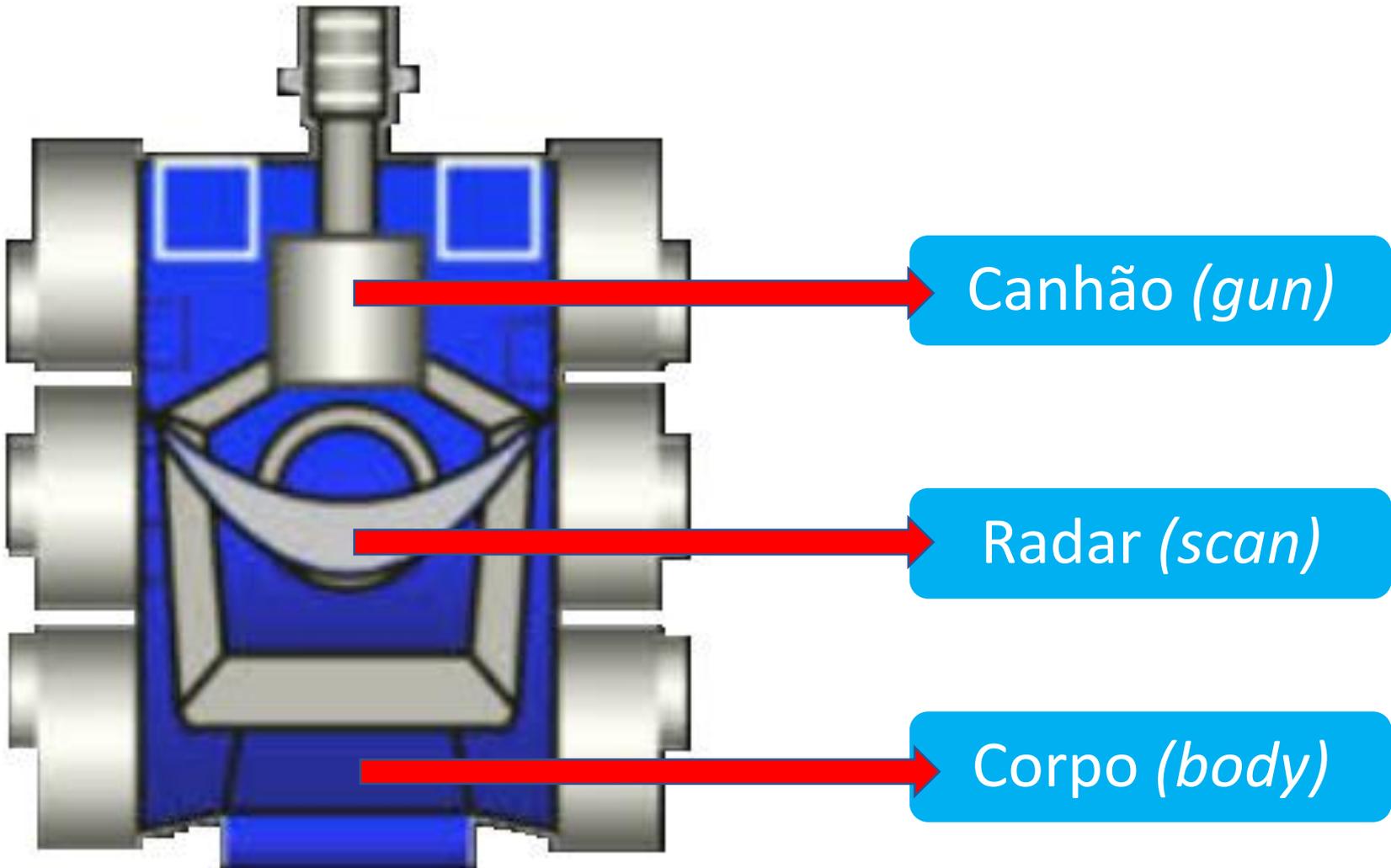
Crazy (2)

Main battle log

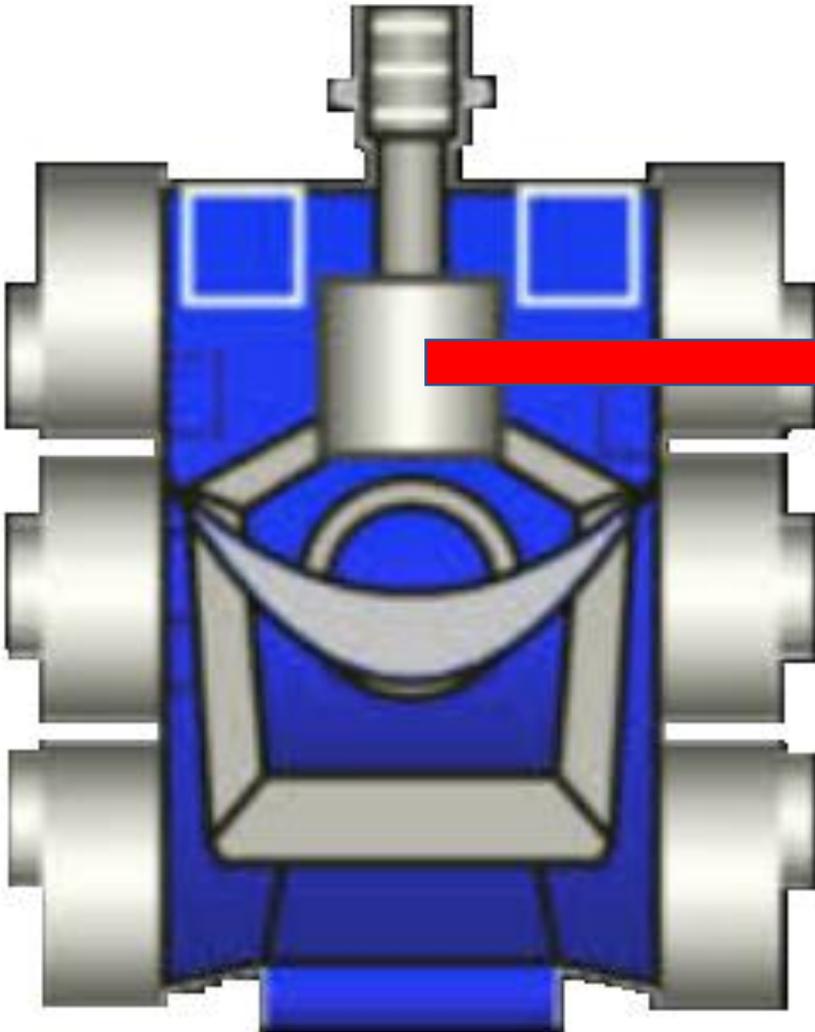
Pause/Debug Next Turn Stop Restart

0 5 10 15 20 25 30 40 50 65 90 150 1000 75

O ROBÔ: Anatomia



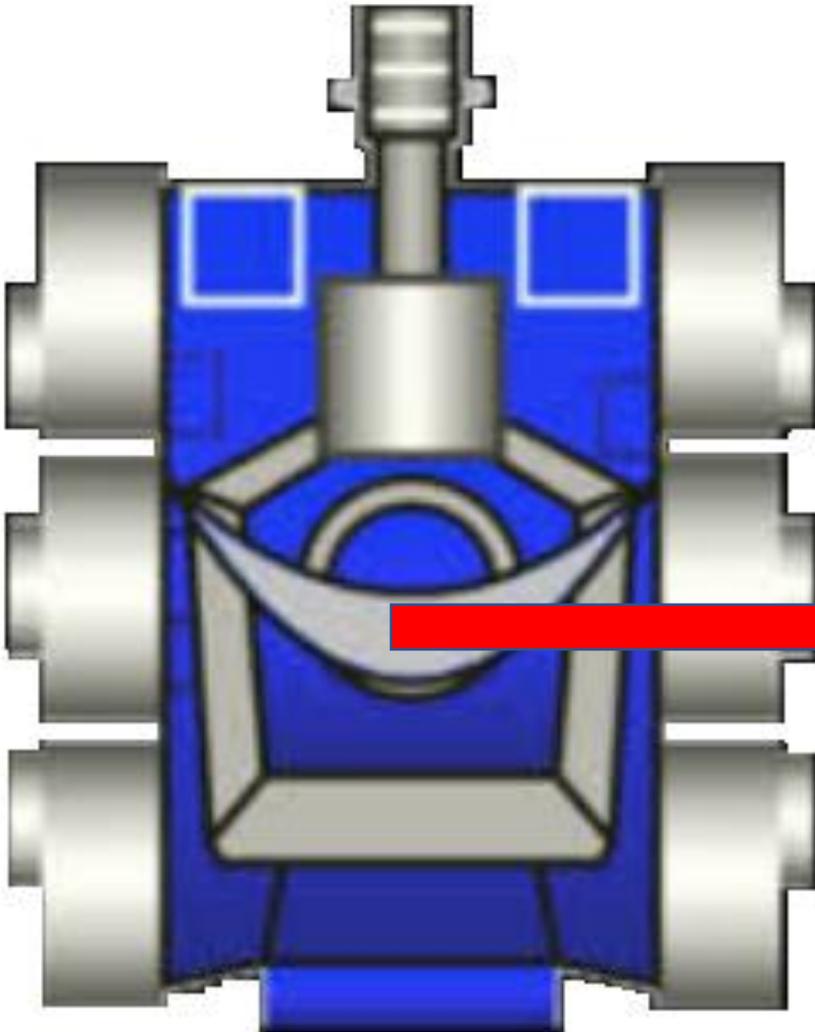
O ROBÔ: Canhão



Canhão (*gun*)

- atira nos inimigos
- é controlado separadamente do corpo
- pode girar em 360 graus
- atira em diferentes ângulos e distâncias

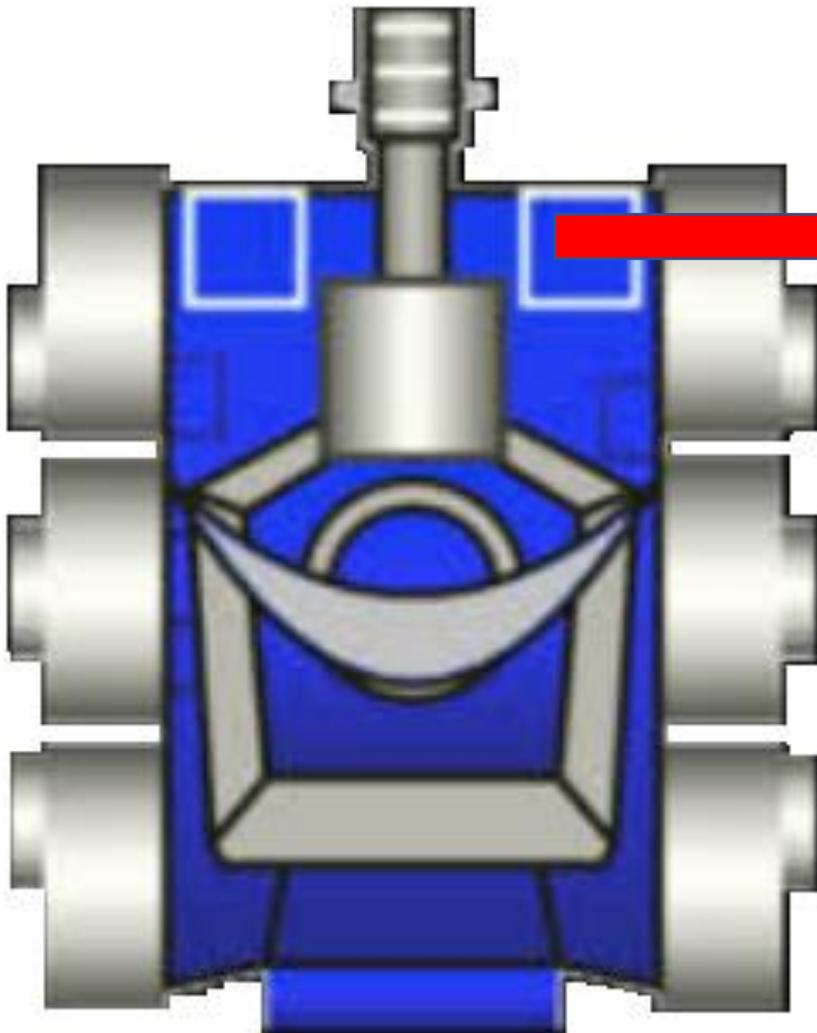
O ROBÔ: Radar



Radar (*scan*)

- usado para identificar os inimigos
- pode girar em 360 graus
- coleta informações de seus inimigos

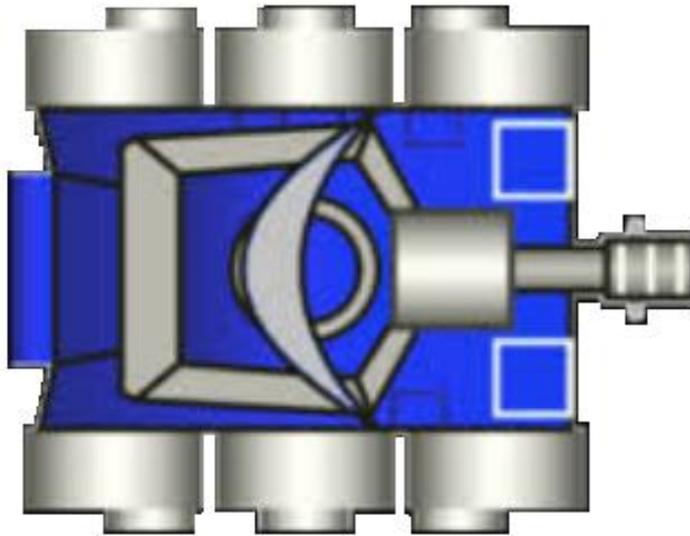
O ROBÔ: Corpo



Corpo (*body*)

- controla o movimento do robô e a direção em que ele está virado.
- movimenta-se para frente e para trás
- movimenta-se para esquerda e direita
- contém informações importantes sobre o estado do robô, como sua energia e velocidade

O ROBÔ: Restrições

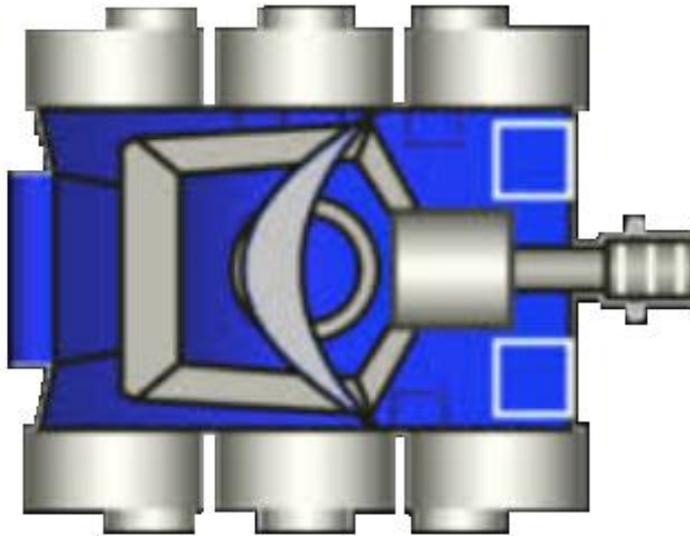


Energia



Calor

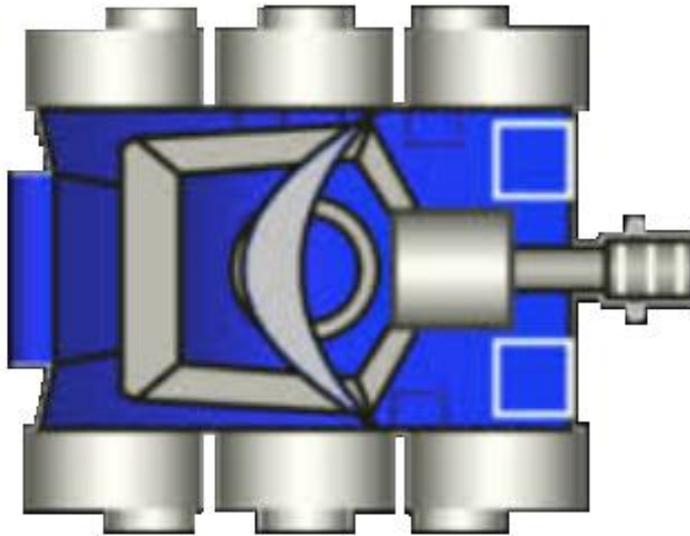
O ROBÔ: Energia



Energia

- Utilizada nas ações do robô
- Diminui, se: colidir com robô inimigo ou na parede
- Aumenta, se: quando causar danos no inimigo

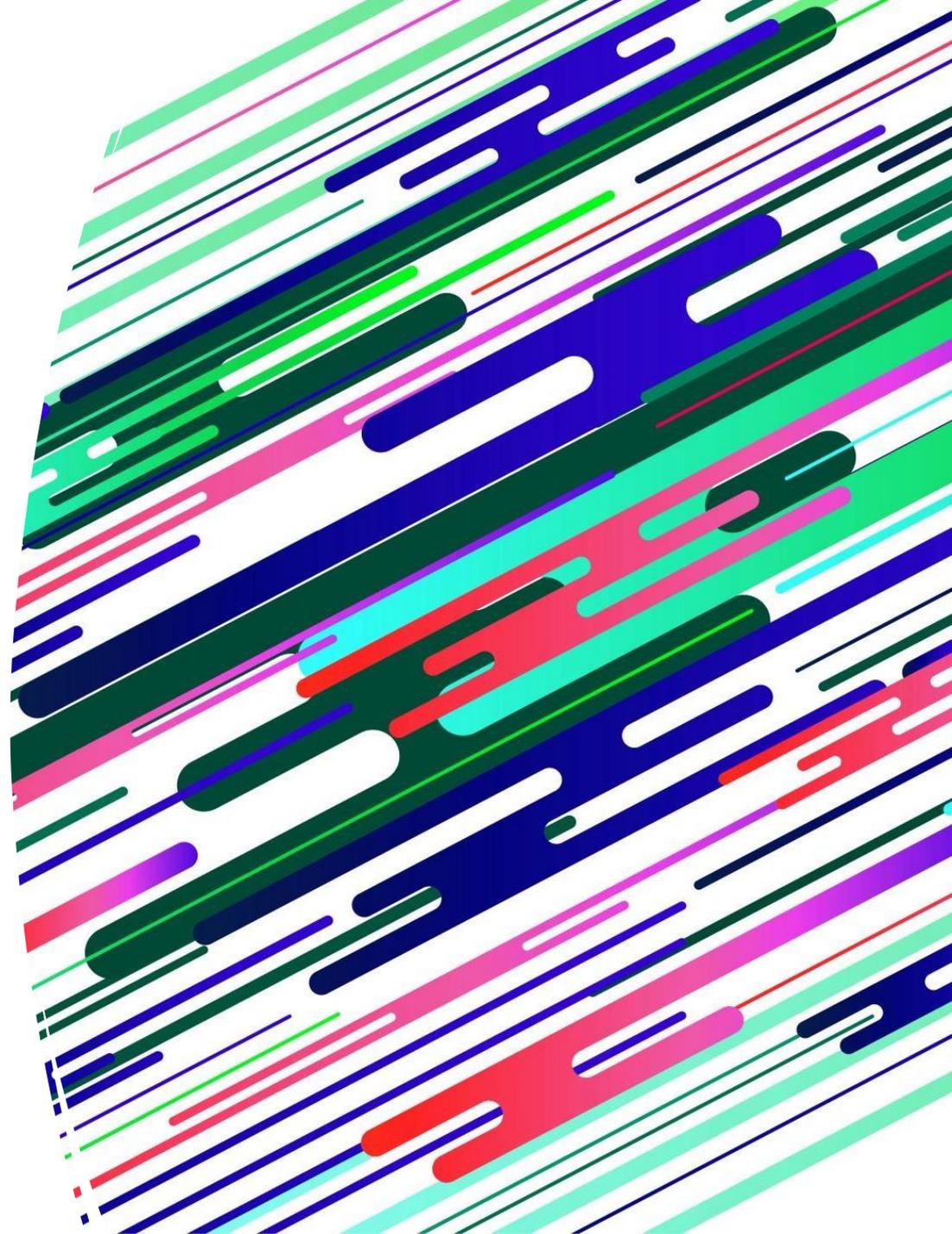
O ROBÔ: Calor



Calor

- O canhão só funciona se o calor tiver em zero
- O calor gerado é proporcional a potência usada do disparo da bala

Métodos



Métodos: Movimentação

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).
turnGunRighth(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.
turnRadarRighth(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.

Métodos: Tiro

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 <i>Bullet</i> , além disso, manda outro tiro em seguida, este com mais velocidade, se o primeiro tiro tiver boas possibilidades de acertar.

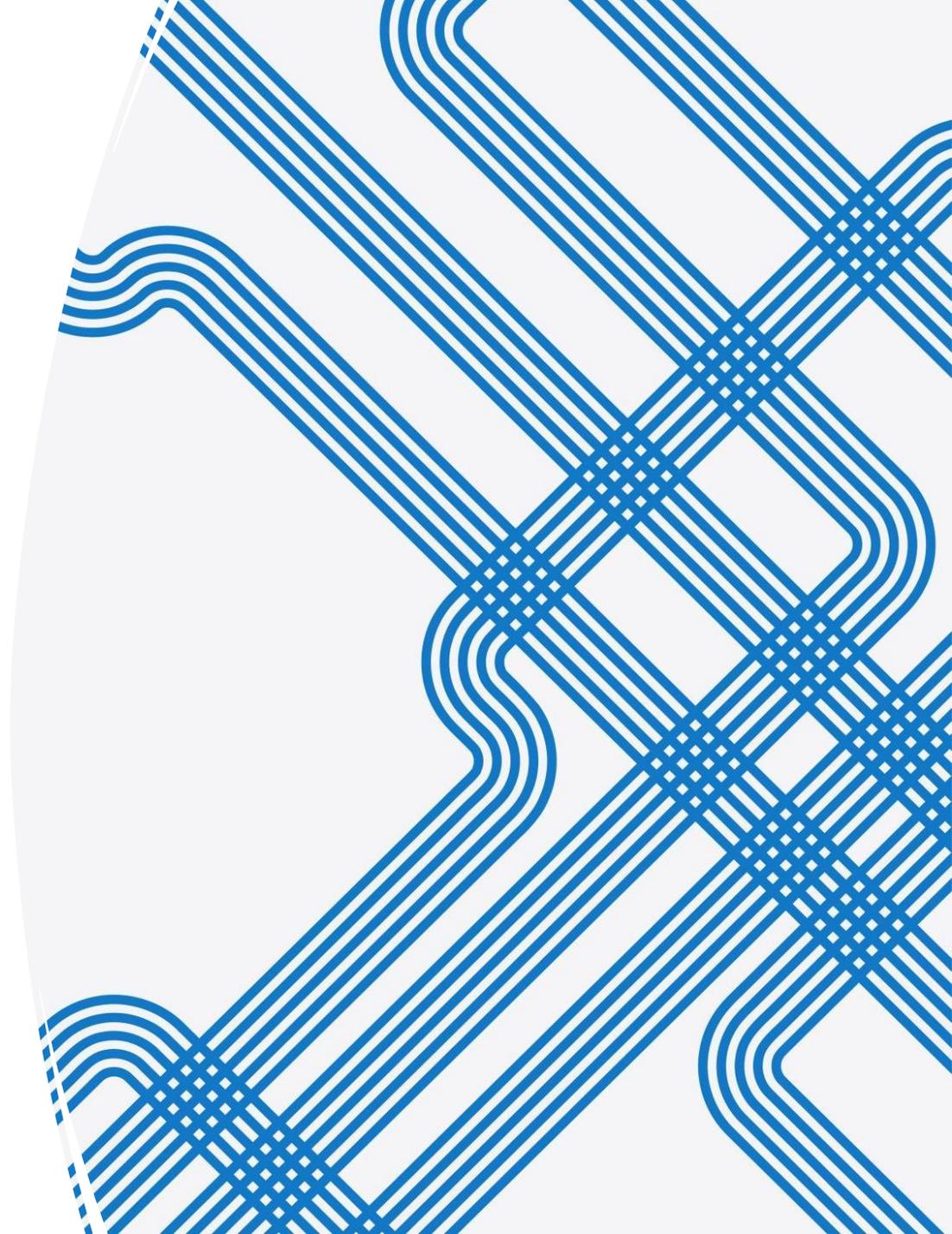
Outros Métodos

Enviar dados para o robô

Retorna dados do robô

Retorna dados da batalha

Eventos



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.

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);  
}
```

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);  
}
```

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);  
}
```

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");  
}
```

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");  
}
```

Evento onBulletHitBullet()

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");  
}
```

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");  
}
```

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");  
}
```

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");  
}
```

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");  
}
```

Evento onWin()

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

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

Pontuação: Quem Vence?

Vitória: os robôs recebem pontos por vencerem uma batalha na arena.

Danos causados: os robôs recebem pontos por causar danos em outros robôs durante a batalha.

Sobrevivência: os robôs recebem pontos por sobreviverem até o final da batalha, mesmo que não tenham causado nenhum dano.

Estratégia: os robôs recebem pontos por terem uma estratégia eficaz na batalha, como evitar os ataques dos outros robôs ou buscar os pontos falhos dos adversários.

Bora Praticar!!!!

1. Crie um robô que se mova em linha reta e gire em círculos.

2. Faça o robô detectar outros robôs e evitar colisões.

3. Faça o robô atirar em outros robôs quando eles estiverem a uma certa distância.

4. Faça o robô detectar as paredes e evitar colidir com elas.

5. Faça o robô detectar a posição de outros robôs e seguir o robô mais próximo.

REGULAMENTO



Inscrições:

Todas as informações referentes a inscrição da equipe (unidade, professor e alunos) estão disponíveis no site www.robotica.cpscetec.com.br/robocode2023.

REGULAMENTO

Fases do torneio:



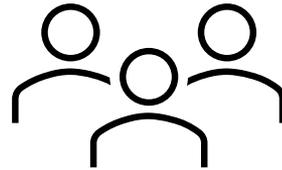
Torneio Local: Unidade
30/08/2023



Final: Live
20 a 22/09/2023

REGULAMENTO

Formação das equipes



até 3 alunos por equipe

nome da equipe, que deverá ser o mesmo
nome do robô (nome_do_robô.java)

REGULAMENTO

Parte Técnica



A implementação do Robô deve ser feita em um único arquivo **.java**.



O nome do robô, deverá ser o mesmo nome da equipe (*nome_do_roboto.java*)



No início do código-fonte do robô, **deverá conter o nome da equipe e nome dos integrantes**, como comentário.



Deverá obrigatoriamente constar trechos de comentários de forma clara e simples, sobre a lógica aplicada para o desenvolvimento.



Plagiar robôs caracteriza desclassificação da equipe.

REGULAMENTO

Premiações:

Na 1ª Fase, fica a critério da escola/professores, providenciar premiações para os 3 primeiros colocados.

Na 2ª Fase, os 3 primeiros colocados receberão certificados e brindes fornecidos por empresas apoiadoras da iniciativa.

Todos os participantes inscritos (alunos e professores) com robôs submetidos e relatório de evidências da realização da 1ª fase, receberão o certificado de participação.

REGULAMENTO



CONFIGURAÇÃO DAS BATALHAS

- arena
- 1x1



EVIDÊNCIAS DAS BATALHAS



SUBMETER ARQUIVOS



ROBÓTICA

Paula Souza

Equipe Robótica Paula Souza

CETEC Capacitações

Centro Paula Souza

www.cps.sp.gov.br

www.robotica.cpscetec.com.br

robotica@cps.sp.gov.br

Premiação



MAMUTE
Eletrônica