Introdução ao Robocode: Utilizando Java para construir e destruir robôs

Darielson A. de Souza
Luis Bruno P. do Nascimento
George Max P. de Souza

IFPI - FMN - UESPI - CEEP
ENUCOMP
ENCONTRO UNIFICADO DE COMPUTAÇÃO EM PARNAÍBA

Quem somos?

- **Luís Bruno Pereira do Nascimento**
- **E-mail:luisbrunu@gmail.com**

- **Darielson Araújo de Souza**
- **E-mail:daryewson@gmail.com**

- **George Max Pereira de Souza**
- **E-mail:georgemaxphb@gmail.com**

Sumário

- Como tudo começou?
- O que é o Robocode?
- Conceitos básicos
- Anatomia do Robô
- Regras
- Como instalar
- A batalha
- Conhecendo o Ambiente
- Introdução à plataforma Java e OO
- Eventos
- Métodos (exemplos de código)
- Sites úteis
- Se der tempo, haverá uma surpresa :)

Motivação

“Parte da motivação para escrever o Robocode foi provar ao mundo que as sentenças 'Java é lento' e 'Você não pode escrever jogos em Java' não são mais verdadeiras. Eu acho que consegui.”

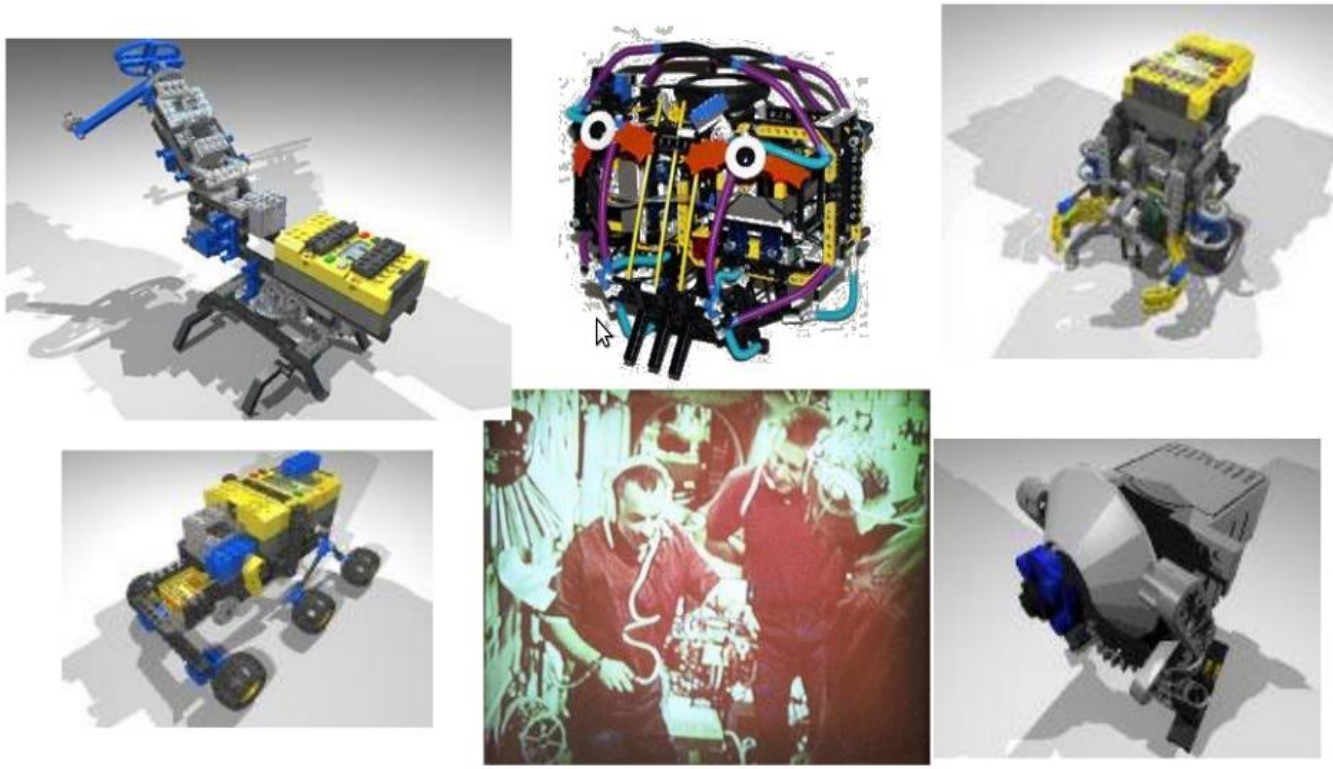
Mathew Nelson – Criador do Robocode

Como tudo começou?
Robôs em Java

Robôs de verdade mesmo, não virtuais :-)

LeJOS: Java em Legos

- Projeto Open-Source que possibilitou o uso de controladores RCX.



Tommy

- Participou do Darpa Grande Challenge 2005
- Usa a tecnologia PRI-MAX que é desenvolvida totalmente em Java pela Perrone Robotics.

Java onde nenhuma outra chegou

- Robôs que estão atualmente em marte como o Spirit que possuem JVM´s embutidos com Real Time Java.

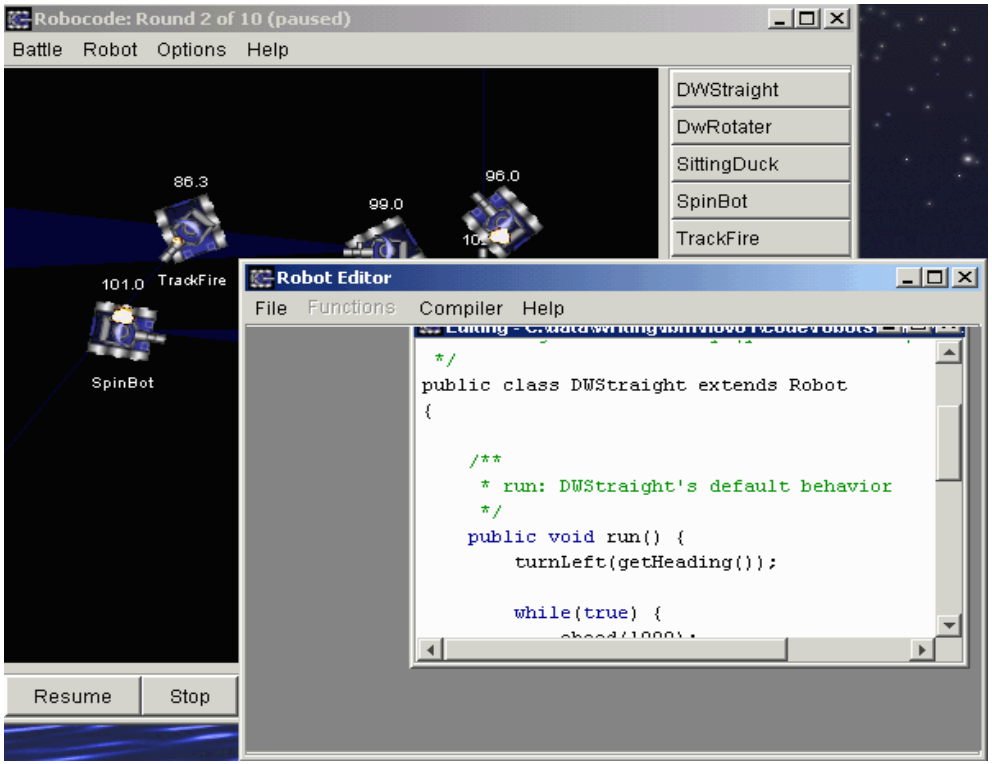
O que é o Robocode?

- É um jogo com um simulador de batalhas entre tanques de guerra.
- Cada tanque de guerra no simulador robocode é controlado por um programa escrito na linguagem Java.
- O jogo foi iniciado por Matthew A. Nelson no final de 2000 e se tornou profissional quando ele a trouxe à IBM, na forma de um download AlphaWorks, em julho de 2001

Mais sobre Robocode

- O Robocode foi colocado no SourceForge no início de 2005 como uma ferramenta de código aberto para ser continuada, pois estava com o desenvolvimento parado.
- A comunidade começou a desenvolver suas próprias versões para eliminar os erros que haviam na versão existente, e assim foi crescendo com novas funcionalidades.

Mais sobre Robocode



- Atualmente o Robocode está na versão oficial (instavel) 1.8.3.0 de 04 de Outubro de 2013, podendo programar os robôs em .NET além de java, desde a versão 1.7.2.0.

Conceitos Básicos

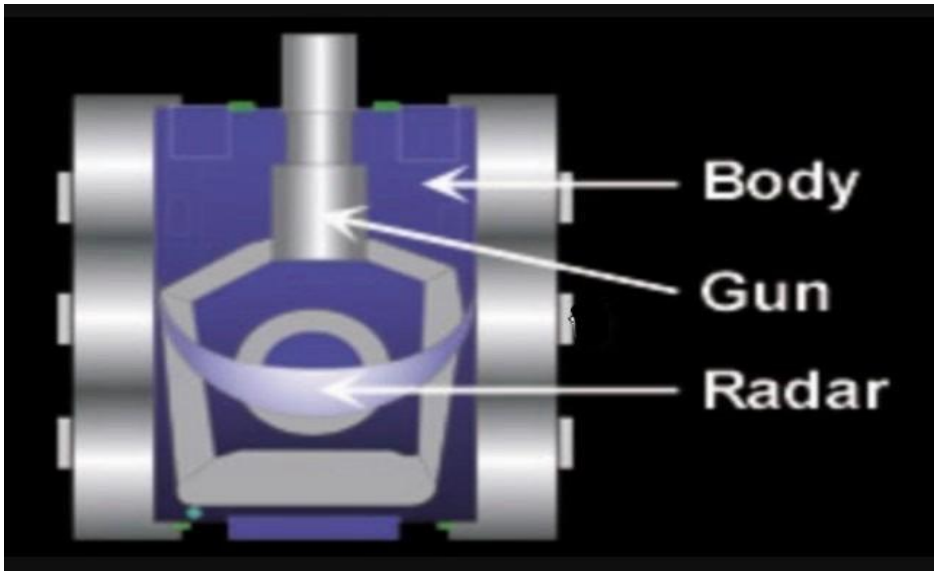
- Antes de o jogador iniciar o programa é importante atentar para alguns conceitos como: método, atributo e evento.

Mais Conceitos básicos

- O método é o elemento que representa uma chamada para algum procedimento de um objeto.
- O atributo é a característica atribuída a um objeto
- evento é o resultado de uma ação, como a ação de receber um tiro ou de acertar o tiro, podendo acionar um novo procedimento (um novo tiro, por exemplo).

Anatomia do Robô

- O robô tem o formato de um taque de guerra, este, dividido em 3 componentes: Corpo, Canhão e Radar.

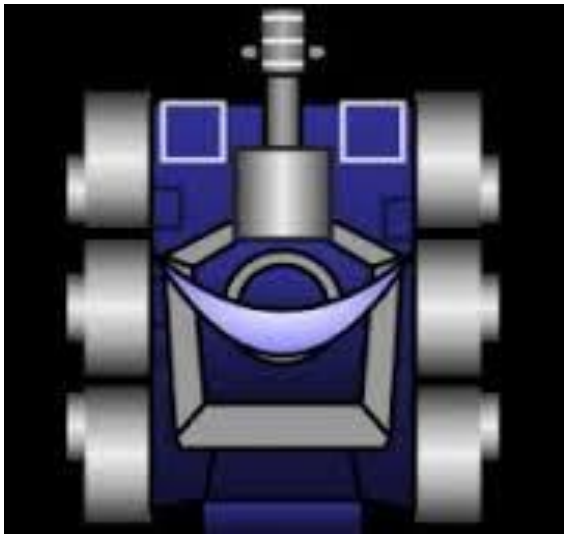


Mais sobre Anatomia do Robô

- Corpo (Body)
  - Responsável pelos movimentos.
    - Mover para frente;
    - Mover para trás;
    - Girar para esquerda;
    - Girar para direita;
    - Parar
- Canhão (Gun)
  - Responsável por realizar os disparos.
  - Gira para esquerda e para a direita
- Radar
  - Localiza os inimigos
  - Girar para esquerda e para a direita,

Mais sobre Anatomia do Robô

- O canhão fica por cima do corpo e o radar está montado em cima do canhão. O giro do corpo é acompanhado ou não pelo canhão e o radar.



Regras

- O jogador deve saber que o robô pode se movimentar somente para frente, para trás e fazer curvas.
- O canhão (gun) pode virar no sentido horário e anti-horário em 360 graus, e dar tiros(bullet) de força maior que zero e menor ou igual a 3.
- Se o robô ficar sem energia, ou seja, energia igual a zero, ele fica desabilitado (disabled) e perde os seus movimentos.
- Deve-se saber também que robô é "cego", a única coisa que ele vê são os robôs adversários, capturados pelo radar(radar), o qual não vê os tiros.

Mais sobre Regras

- Todos os robôs começam cada round com energia 100, e os que ficarem com energia abaixo de zero vão sendo eliminados (explodem) restando apenas um, e então começa um novo round.

- No fim de todos os rounds a batalha acaba e aparece uma tabela mostrando a pontuação e a colocação.

Mais sobre Regras

- O robô perde energia quando:
  - Atingido por um tiro:
    - perde uma quantidade de energia equivalente a 4 vezes a potência do tiro.

Se a potência for maior ou igual a 2, é realizado um dano adicional de 2 vezes a (potência -1).

  - Realizando um tiro:
    - Perde a quantidade equivalente à potência do tiro realizado;
  - Se bater em outro robô, os dois perdem (energia – 1).

O robô só ganha energia se atingir outro robô com um tiro, onde ganha 3 vezes a potência do tiro realizado.

Instalação

- Antes de tudo baixe o java: http://www.oracle.com/technetwork/pt/java/javase/downloads/index.html

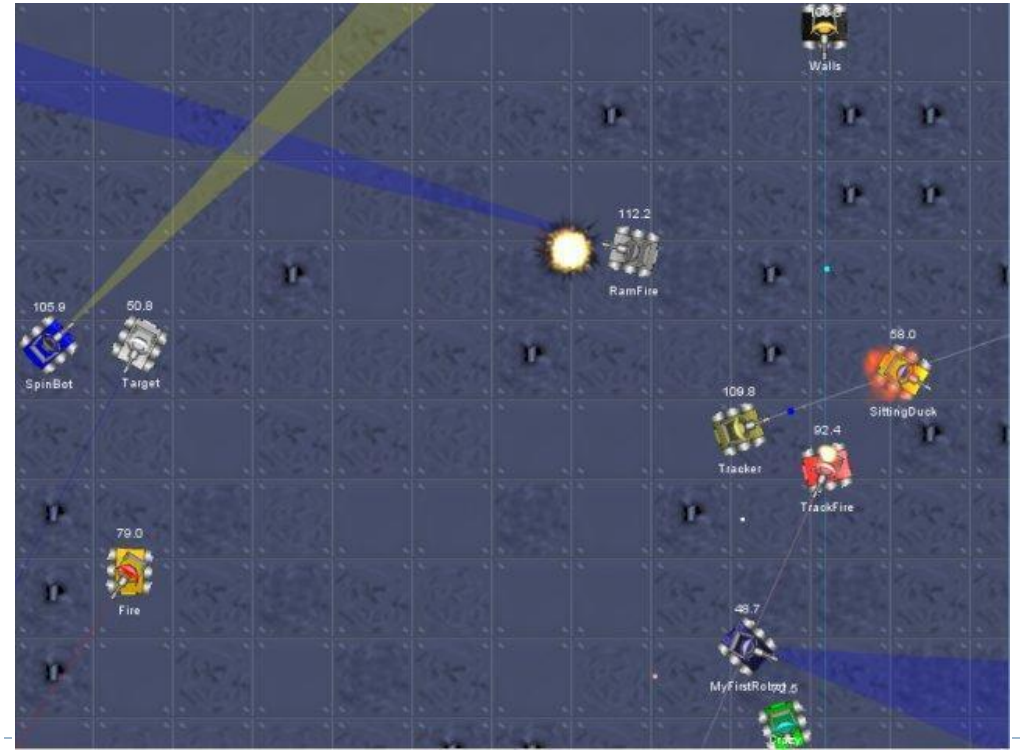
- Acesse a página: http://sourceforge.net/projects/robocode/files/

- Faça setup.jar

- Siga

Campo de Batalha

- batalhas são realizadas em um campo virtual, onde as dimensões desses campos são indicadas pelo jogador, na qual são medidas em pixels.



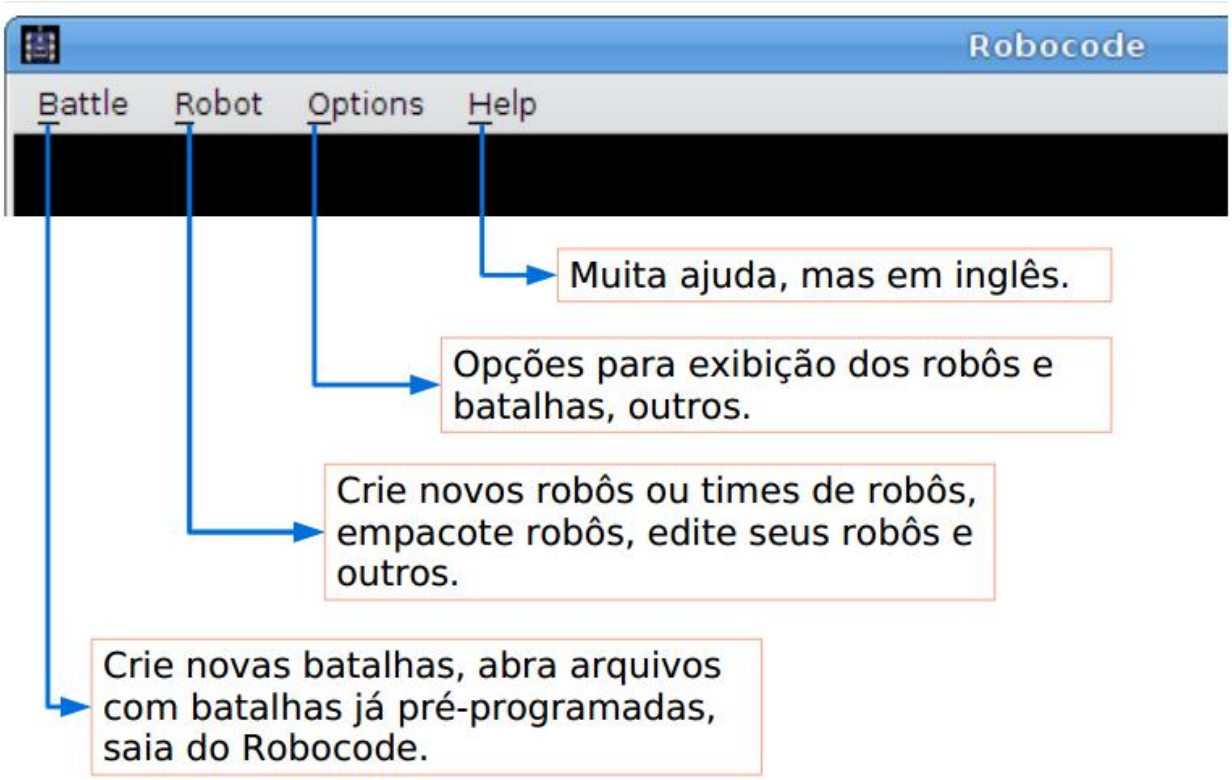
Mais sobre o Campo de Batalha

O sistema de pontuação diz quem é o vencedor. Os pontos são atribuídos da seguinte maneira:

- Sobrevivência:
  - Sempre que um tanque é destruído, todos os tanques que ainda estão na batalha recebem 50 pontos;
- Sobrevivente
  - O último tanque restante recebe um bônus de 10 pontos para cada tanque que foi destruído, independente se tais tanques tenha sido destruído pelo sobrevivente ou não;
- Danos por tiro:
  - Cada tanque recebe um ponto para cada disparo acertado.
- Danos por colisão
  - A cada 1 ponto de dano causado em outro robô através de uma colisão, é recebido 2 pontos;

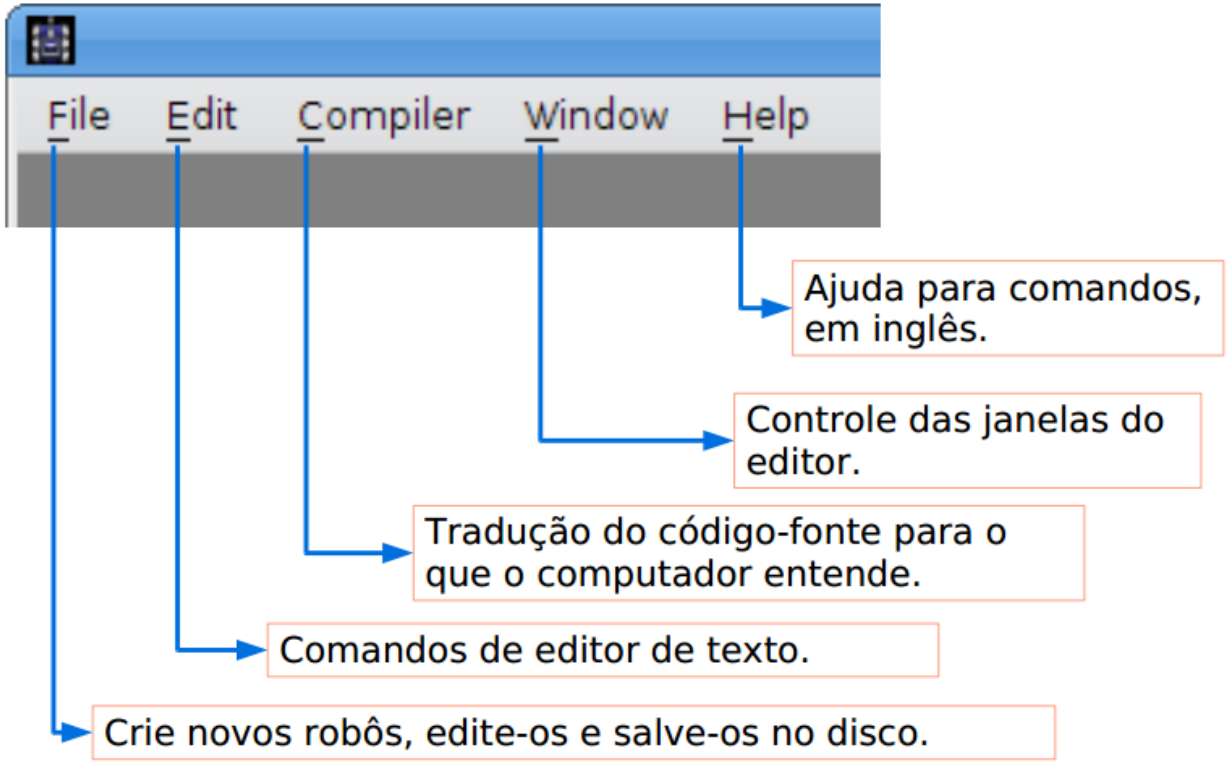
Conhecendo o Ambiente

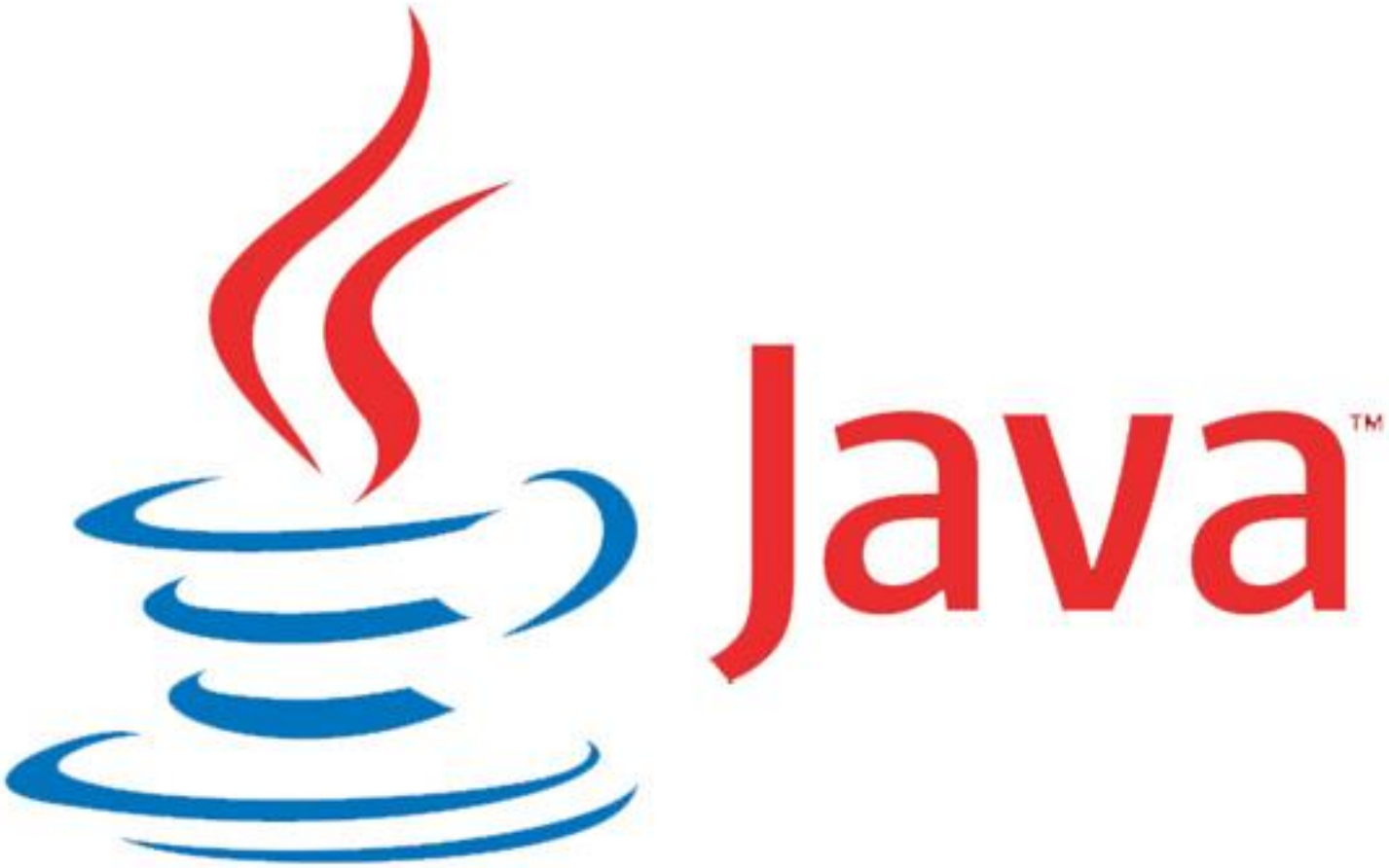
- Robocode ferramenta



Conhecendo o ambiente

- Robocode-Editor




Java™

Java

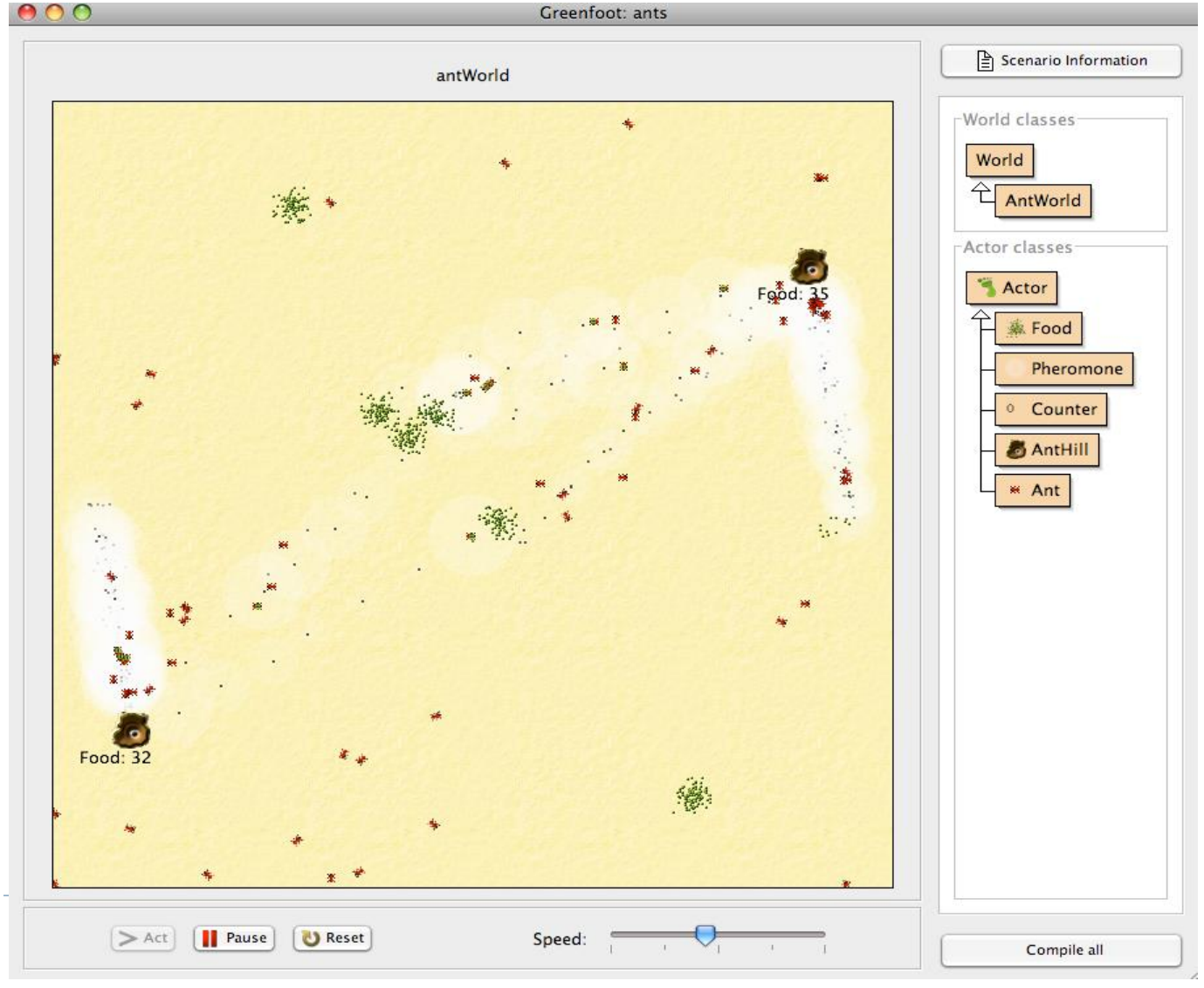
- Java é executado em mais de 850 milhões de computadores;
- Utilizado como linguagem inicial no processo de aprendizagem de lógica de programação;
- A comunidade de usuários Java têm criado muitas ferramentas para auxiliar nesse processo de aprendizado;

Softwares Educacionais para Java

Greenfoot

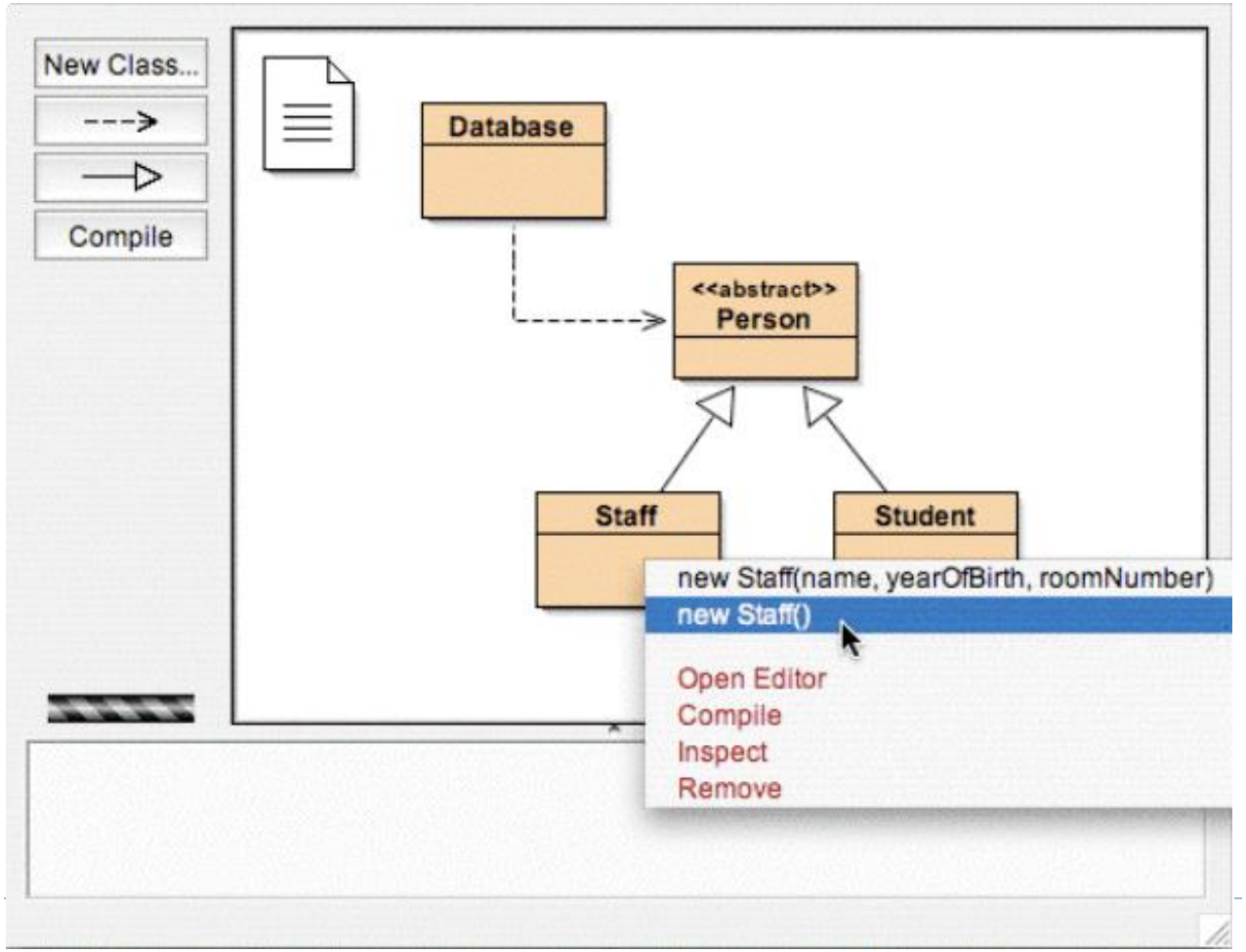
Softwares Educacionais para Java

- Greenfoot



Softwares educacionais para Java

- BlueJ



Mitos

- **Java é lento**
  - A distancia entre Java e C/C++ está diminuindo graças às melhorias em suas novas versões, melhores compiladores JIT(just-in-time) que convertem bytecode para código nativo em tempo de execução.
  - http://regispires.wordpress.com/2010/10/23/comparacao-de - performance/http://regispires.wordpress.com/2010/ 10/23/comparacao-de-performance/
- **Ninguém mais usa Java**
  - Segundo o índice do TIOBE, a Linguagem Java é a segunda mais usada, depois de C. (Novembro, 2013)
  - *http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html*
- **Java e JavaScript são a mesma coisa**
  - São linguagens bem diferentes. JS é uma Linguagem de scripts que pode ser usada em páginas WEB e tem uma linguagem baseada na sintaxe de Java.

Uma Visão Geral

Características

- **Orientada a Objetos**
  - Foco nos dados e métodos utilizados para manipula-los.
- **Simples**
  - A linguagem possui apenas os mecanismos necessários para implementar seu conjunto de funcionalidades.
- **Robusta**
  - Tipagem de dados é forte, ou seja, tipos checados em tempo de execução.
  - Não possui ponteiros, evitando a corrupção de dados na memória.
  - Tem mecanismos para tratamento de exceções

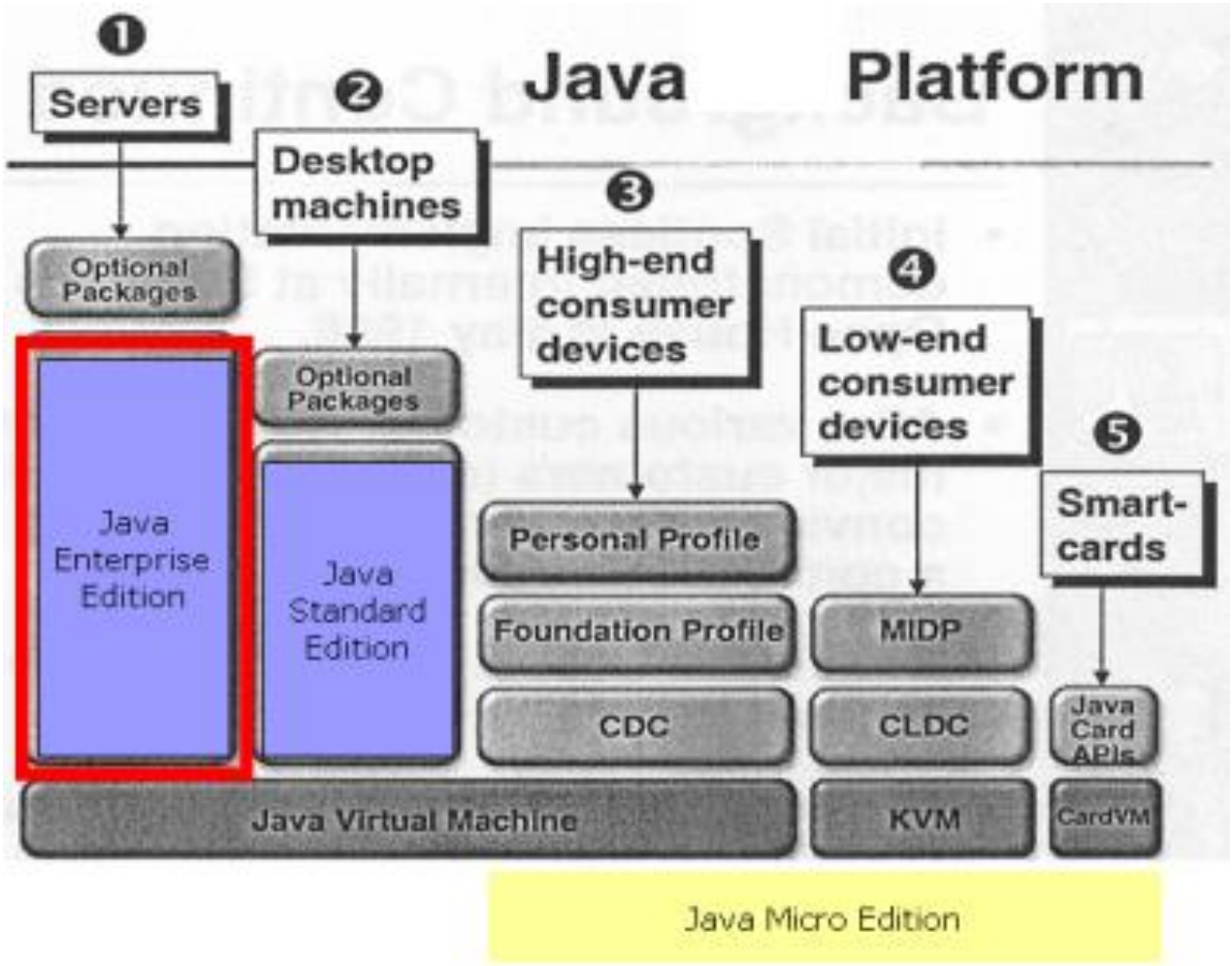
Características

- **Independente de Arquitetura**
  - O bytecode gerado, independe da arquitetura de hardware e software.
  - " write once run anywhere“
- **Multi-Thread**
  - As Threads permitem a execução concorrente de código.

Características

- **Coleta de Lixo Automática**
  - A coleta de lixo permite que a memória alocada para objetos seja recuperada. Java faz isso automaticamente, aumentando a produtividade e diminuindo o número de *bugs*.
- **Bombril**
  - Abrange os mais diversos tipos de aplicações:
  - Acesso a banco de dados;
  - Interface gráfica;
  - Cliente-Servidor;
  - Robótica;
  - ...
- **MUITA** documentação;

A plataforma Java



A plataforma Java

- **Java Virtual Machine (JVM)**
  - Provê suporte para independência de plataforma;
  - Interpreta o código Java compilado.
  - Java é uma compilada e interpretada
- **Java Application Programming Interface (API)**
  - Provê suporte para programação de aplicações em geral.
  - Vasta gama de componentes de software prontos para uso para as mais diversas funcionalidades;

A Orientação a Objetos

- **Objeto**
  - Representa um elemento do Domínio.
  - Têm características e pode realizar ações.
- **Classe**
  - Funciona como uma “forma” para criar objetos.
- **Métodos**
  - Representam as ações que um objeto pode realizar.
- **Instanciar**
  - Ato de criar um objeto a partir dessa “forma”.

Orientação a Objetos

- **Herança**
  - Conceito da OO que prega a ideia "**don't repeat yourself**", ou seja, a reutilização de código através da herança de código.

- **Polimorfismo**
  - É a mesma ação executada

de forma diferente pelos objetos.

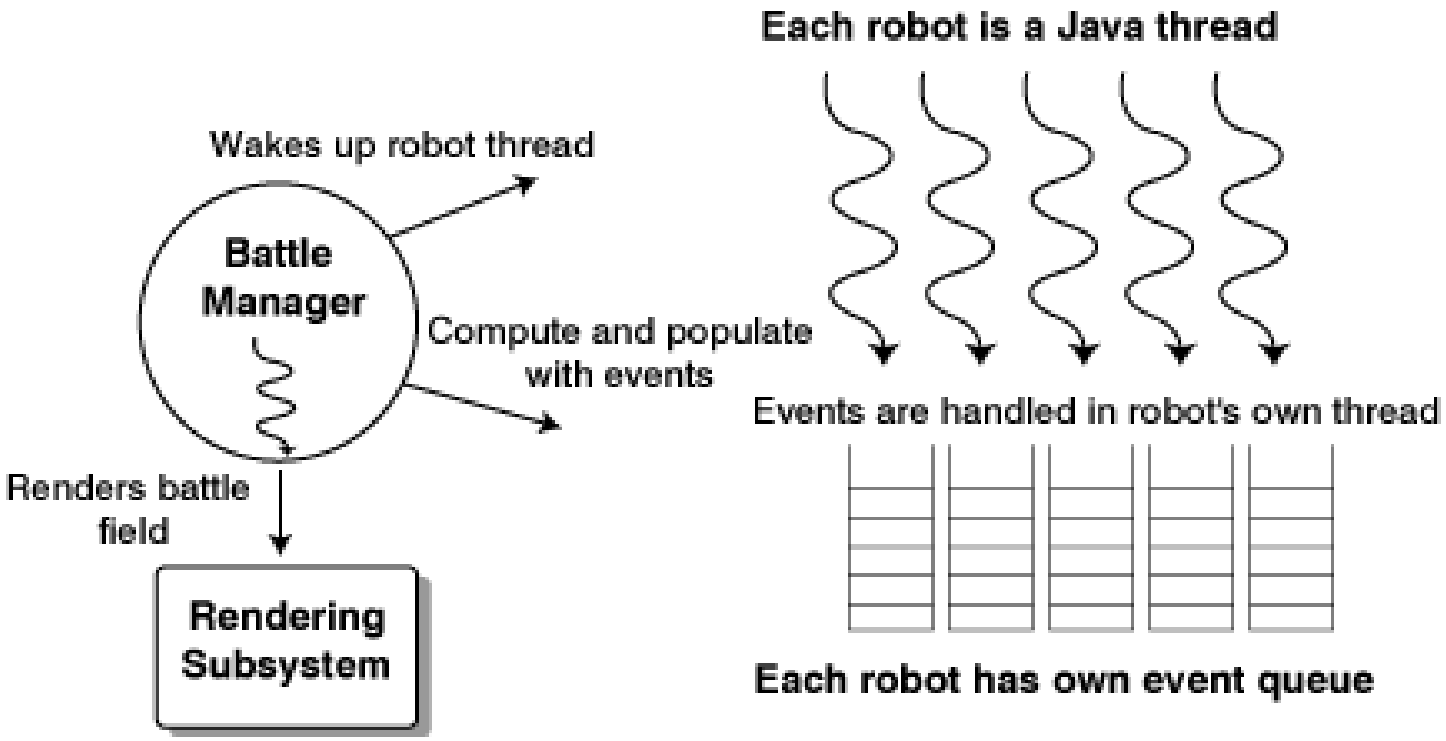
```
politico.roubar();
assaultante.roubar();
```

Classe MeuRobo

```
package novo;
import robocode.*;

public class MeuRobo extends Robot {

	public void run() {
		while (true) { //Enquanto o robô estiver ativo, o código será executado.

			ahead(100); //caminha 100 pixels para frente
			turnGunRight(360); //gira canhão em 360º para a direita

			back(100); //volta 100 pixels
			turnGunRight(360); //gira canhão em 360º para a direita
		}
	}

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

Funcionamento do Robocode

- Cada robô é executado em sua própria thread.
- Quem controla todo o sistema é a thread Battle Manager (gerenciador de batalha).



Eventos

- É fundamental conhecer todos os eventos disponíveis do Robocode, para que se possa pensar nas estratégias e inteligência que o seu robô irá possuir para competir com os inimigos.

- Os eventos são chamados quando acontece algo específico no decorrer do combate. Alguns deles te enviam, por parâmetro, dados do robô adversário em questão para você trabalhar com esses valores dentro do evento.

Eventos

INDICE:

- **Run**
- **onScannedRobot**
- **onWin**
- **onHitRobot**
- **onHitWall**
- **onHitByBullet**
- **onBulletHit**
- **onBulletMissed**
- **onBulletHitBullet**
- **onDeath**
- **onRobotDeath**
- **onSkippedTurn**

Eventos

Run

- É executado quando o round for iniciado. Diferente do que muitos pensam, esse evento só será chamado novamente quando iniciar outro round. Por isso é muito comum e recomendado usar um loop infinito dentro dele, para que seu robô nunca fique parado quando não tiver sendo executado outro evento.

```
public void run(){
    do {
        turnRadarLeft(360);
        setAhead(200);
        turnRight(300);
    }while(true)
}
```

onScannedRobot

Executado quando o radar do seu robô encontra um adversário.

É um dos eventos mais importantes pois é uma forma de saber a energia, distância,

o ângulo dos seus inimigos.

onScannedRobot

- **Metodos da classe *ScannedRobotEvent*:**

Comandos	Tipo Retorno	Descrição Retorno
getName()	String	Retorna o nome do robô adversário scaneado.
getBearing()	double	Retorna o ângulo do robô adversário em relação ao seu robô
getBearingRadians()	double	Ângulo em radianos do robô adversário em relação ao seu.
getDistance()	double	Retorna a distância do robô adversário em relação ao seu robô.
getEnergy()	double	Retorna a energia do robô adversário.
getHeading()	double	Retorna o ângulo em graus do adversário em relação a tela.
getHeadingRadians()	double	Retorna o ângulo em radianos do adversário em relação a tela.
getVelocity()	double	Retorna a velocidade do robô scaneado.

```
public void onScannedRobot(ScannedRobotEvent Inimigo) {
      double ângulo = Inimigo.getBearing();
      double distancia = Inimigo.getDistance();
      if ( distancia < 200 ) {
          TurnGunRight(angulo);
          fire(2);
         }
}
```

Obs: Não confunda "getEnergy()" com " Inimigo.getEnergy()", pois o primeiro é a energia de seu robô e o outro a energia do robô que foi escaneado.

onWin

- É executado quando seu robô ganha o round.

```
public void onWin(WinEvent e) {
    turnRight(36000);
}
```

onHitRobot

- É executado quando seu robô bate em outro robô. Aproveite que você está bem perto do inimigo, vire o canhão para ele e mande um tiro de força máxima, porque dificilmente errará.

Métodos da classe *HitRobotEvent*:

Comandos	Tipo Retorno	Descrição Retorno
getName()	String	Retorna o nome do robô adversário colidido.
getBearing()	double	Ângulo em graus do robô adversário em relação ao seu robô
getBearingRadians()	double	Ângulo em radianos do robô adversário em relação ao seu robô.
getEnergy()	double	Retorna a energia do robô adversário.
isMyFault()	boolean	Retorna true se foi seu robô quem originou o evento, e false se foi o adversário que bateu em seu robô.

Exemplo:

```
public void onHitRobot(HitRobotEvent INI) {
    turnRight(INI.getBearing());
    fire(3);
}
```

onHitWall

- É executado quando seu robô colide com a parede. Quando seu robô bate na parede, perde energia.
- É bacana mudar o robô de posição antes que sua energia = -1;

Comandos	Tipo Retorno	Descrição Retorno
getBearing()	double	Ângulo em graus da parede batida em relação ao seu robô.
getBearingRadians()	double	Ângulo em radianos da parede batida em relação ao seu robô.

```
public void onHitWall(HitWallEvent e) {
    turnLeft(180);
}
```

onHitByBullet

- É executado quando seu robô leva um tiro.

Comandos	Tipo Retorno	Descrição Retorno
getName()	String	Retorna o nome do robô adversário que te acertou um tiro.
getBearing()	double	Retorna o ângulo do robô adversário em relação ao seu robô.
getBearingRadians()	double	Ângulo em radianos do robô adversário em relação ao seu robô.
getHeading()	double	Retorna o ângulo em graus do robô adversário em relação a tela.
getHeadingRadians()	double	Ângulo em radians do robô adversário em relação a tela.
getBullet()	Bullet	Retorna o Bullet (tiro) do robô adversário que atingiu seu robô.
getPower()	double	Retorna a força do tiro.
getVelocity()	double	Retorna a velocidade do tiro.

```
public void onHitByBullet(HitByBulletEvent e) {
    ahead(100);
}
```

onBulletHit

- É executado quando seu tiro acerta um adversário.

Métodos da classe *BulletHitEvent*:

Comandos	Tipo Retorno	Descrição Retorno
getName()	String	Retorna o nome do robô adversário que você acertou um tiro.
getBullet()	Bullet	Retorna o Bullet (dados do tiro) de seu robô que atingiu o adversário.
getEnergy()	double	Retorna a energia do robô adversário após levar o tiro.


```
public void onBulletHit(BulletHitEvent e) {
    Acertos++;
}
```

onBulletMissed

- Chamado quando uma de seus tiros colide com a parede(erra o tiro).

Métodos da classe *BulletMissedEvent*:

Comandos	Tipo Retorno	Descrição Retorno
getBullet()	Bullet	Retorna o Bullet (dados do tiro), de seu robô, que bateu na parede.

```
public void onBulletMissed(BulletMissedEvent e) {
    vacilos++;
}
```

onBulletHitBullet

- É executado quando uma de suas balas colide com outra bala.

Métodos da classe *BulletHitBulletEvent*:

Comandos	Tipo Retorno	Descrição Retorno
getBullet()	Bullet	Retorna o Bullet (dados do tiro) de seu robô.
getHitBullet()	Bullet	Retorna o Bullet do robô adversário.

```
public void onBulletHitBullet(BulletHitBulletEvent e)
{
}
```

onDeath

- Executado se seu robô morrer.

```
public void onDeath(DeathEvent e) {
    System.out.println(getName()+" morreu!");
    System.out.println("Quantidade de inimigos ainda vivo: "+getOthers());
}
```

onRobotDeath

- É chamado quando morre um adversário.

Comandos	Tipo Retorno	Descrição Retorno
getName()	String	Retorna o nome do robô que morreu.

```
public void onRobotDeath(RobotDeathEvent e) {
      if(nome==e.getName())
            nome=null;
}
```

Métodos

- Movimentação – Robot

Comando	Parâmetro	Descrição
ahead(double)	a distância que o robô deverá percorrer.	Movimenta o robô para frente, uma distância x dada por parâmentro. Se o robô bater em outro, ou na parede antes de completar a distancia 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.

Mais sobre Métodos

- Movimentação - 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:

Mais sobre Métodos

```
ahead(100);
turnRight(90);
```

O robô irá andar para frente e depois que tiver terminado de percorrer a distância 100, ele girará 90º.

Mas se tiver:

```
setAhead(100);
setTurnRight(90);
```

Mais sobre Métodos

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

Comando	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.

Mais sobre Métodos

- Movimentação - AdvancedRadiansRobot

Esses métodos "Radians" são usados quando vai se trabalhar com PI, seno, cosseno, tangente.
Os métodos que começam com "set" são como aqueles visto acima, que continuam lendo as linhas de comando abaixo, misturando movimentos.

Mais sobre Métodos

Comando	Parâmetro	Descrição
turnRightRadians(double)	o ângulo em radianos	Gira o robô para a direita.
turnLeftRadians(double)	o ângulo em radianos	Gira o robô para a esquerda.
turnGunRightRadians(double)	o ângulo em radianos	Gira o canhão para a direita.
turnGunLeftRadians(double)	o ângulo em radianos	Gira o canhão para a esquerda.
turnRadarRigthRadians(double)	o ângulo em radianos	Gira o radar para a direita.
turnRadarLeftRadians(double)	o ângulo em radianos	Gira o radar para a esquerda.
setTurnRightRadians(double)	o ângulo em radianos	Herdado do método turnRightRadians.
setTurnLeftRadians(double)	o ângulo em radianos	Herdado do método turnLeftRadians.
setTurnGunRightRadians(double)	o ângulo em radianos	Herdado do método turnGunRightRadians.
setTurnGunLeftRadians(double)	o ângulo em radianos	Herdado do método turnGunLeftRadians.
setTurnRadarRigthRadians(double)	o ângulo em radianos	Herdado do método turnRadarRightRadians.
setTurnRadarLeftRadians(double)	o ângulo em radianos	Herdado do método turnRadarLeftRadians.

Mais Métodos

- Tipo – Robot

Comando	Parâmetro	Descrição
fire(double)	a força do tiro, e subtraido da energia de seu robô.	Atira imediatamente na força mandada por parâmetro, de 1 até 3. Se mandar um tiro maior que 3 ele considera força 3.
fireBullet(double)	a força do tiro, e subtraido 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 da acertar.

- Tipo – AdvancedRobot

Comandos	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.

Mais sobre Métodos

- Envia Dados Para O Robô

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

Mais sobre Métodos

- Retorna Dados do Robô

Comando	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	Retorna a taxa de resfriamento do canhão.
getGunHeat()	double	Retorna quanto o canhão está virando no momento corrente.
getVelocity()	double	Retorna a velocidade do robô.

Mais sobre Métodos

- Retorna Dados do Robô – AdvancedRadiansRobot

Comandos	Tipo do Retorno	Retorno
getHeadingRadians()	double	Retorna a direção que o robô está voltado, em radianos (de 0 até 2*PI).
getGunHeadingRadians()	double	Retorna o ângulo em radianos do canhão está apontado em relação a tela
getRadarHeadingRadians()	double	Retorna o ângulo em radianos do radar está voltado em relação a tela
getTurnRemainingRadians()	double	
getGunTurnRemainingRadians()	double	
getRadarTurnRemainingRadians()	double	

Mais sobre Métodos

- Retorna Dados da Batalha

Comandos	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. Quando inicia outro round o tempo volta a 0(zero). O é tempo equivale ao número de quadros mostrados.

Mais sobre Métodos

- Outros

Comando	Parâmetro	Descrição
doNothing()	nenhum parâmetro	
scan()	nenhum parâmetro	
stop()	nenhum parâmetro	
stop(boolean)		
resume()	nenhum parâmetro	
setResume()	nenhum parâmetro	
setStop()	nenhum parâmetro	
setStop(boolean)		
finalize()	nenhum parâmetro	

Mais sobre Métodos

Você pode digitar um método em qualquer lugar dentro do "public class", no início ou no fim do código, a linguagem Java permite.

INDICE:

- **anguloRelativo**
- **mira**
- **dancinha**
- **dancinha2**
- **dancinha3**
- **risadinha**
- **fogo**
- **tiroFatal**
- **pertoParede**

Mais sobre Métodos

- ***anguloRelativo***

Essa função é muito utilizada, até pelos robôs do diretório samples. Nela você manda por parâmetro o ângulo(valor do tipo double) a ser deslocado pelo radar, ou pelo canhão, ou até mesmo do seu robô se você querer ir em direção ao inimigo, a função retornará se a volta menor será no sentido horário ou anti-horário.

```
public double anguloRelativo(double ANG) {
     if (ANG> -180 && ANG<= 180)
        return ANG;
     double REL = ANG;
     while (REL<= -180)
        REL += 360;
        while (ANG> 180)
             REL -= 360;
  return REL;
}
```

- No evento onScanedRobot ou onHitRobot digite:

  turnRadarRight(anguloRelativo(e.getBearing()+getHeading()-getRadarHeading()));//para mirar o radar no adversário.

  turnGunRight(anguloRelativo(e.getBearing()+getHeading()-getGunHeading())); //para mirar o canhão no adversário.

  turnRight(anguloRelativo(e.getBearing)); //para virar seu robô em direção do adversário

- Obs: Sempre "Right", nunca use "Left" para essa função porque não da certo.

Mais sobre Métodos

- ***mira***

  Isso na verdade é um procedimento e não um método, criado em base do método anterior. Ela mira o canhão com mais rapidez e o código mais limpo. O método anterior já está embutido, ela não será mais necessária.

```
public void mira(double Adv) {
    double A=getHeading()+Adv-getGunHeading();
    if (!(A > -180 && A <= 180))      {
        while (A <= -180)
            A += 360;
        while (A > 180)
            A -= 360;
    }
    turnGunRight(A);
}
```

- No evento onScanedRobot ou onHitRobot digite:

  mira(e.getBearing());//mira o canhão contra o adversário
  fire(2);

Mais sobre Métodos

- ***dancinha***

  Procedimento que você usa para gozar do adversário ou comemorar quando ganhar um round.

```
public void dancinha() {
        setAhead(5);
        setTurnRight(360D);
        setTurnGunLeft(360D);
        setTurnRadarRight(30D);
}
```

  - No evento onWin digite:

    dancinha();

Mais sobre Métodos

- ***dancinha2***

  Aqui outro tipo de dancinha.

```
public void dancinha2() {
    setMaxVelocity(8D);
    setTurnRight(dir * 30);
    setTurnGunLeft(dir * 30);
    setTurnRadarRight(dir * 30);
    if(getTime() % (long)2 == 0L)
        dir *= -1;
}
```

  - No evento onWin digite:

    dancinha2();

Mais sobre Métodos

- ***risadinha***

  Malhando dos perdedores. Treme o robô como se estivesse rindo.

```
public void risadinha() {
        for (int i = 0; i < 50; i++)        {
                turnRight(30);
                turnLeft(30);
        }
}
```

Mais sobre Métodos

- ***fogo***

  Esse procedimento melhora o tiro do seu robô, você não desperdiçará energia e o robô parará de atirar de muito longe quando a energia dele for menor que 15.

```
public void fogo(double Distancia) {
        if (Distancia> 200 || getEnergy() < 15)
            fire(1);
        else if (Distancia > 50)
            fire(2);
        else
            fire(3);
}
```

  - Para usar:

    fogo(e.getDistance());

Mais sobre Métodos

- ***tiroFatal***

  Esse procedimento da um tiro baseado na energia inimiga, ela é muito boa para dar o último quando o adversário estiver com um tiro para morrer(energia <12), porque seu robô não desperdiçará energia. Por exemplo, se o adversário estiver com energia=3, e você der um tiro 3 você vai matá-lo e vai ganhar 3 de energia (3*2-3 =3), mas com essa função você também matará e ganhará 5.15 de energia (3*2-0.85=5.15)

```
public void tiroFatal(double EnergiaIni)
{
        double Tiro = (EnergiaIni / 4) + .1;
        fire(Tiro);
}
```

Para usar:

```
if(e.getEnergy<12)
    tiroFatal(e.getEnergy);
else
   fire(2);
```

Mais sobre Métodos

- ***pertoParede***

  Função que retorna true(verdadeiro) se seu robô está perto da parede

```
public boolean pertoParede() {
        return

       (getX()<50 || getX()>getBattleFieldWidth()-50 ||
        getY()<50 ||
          getY()>getBattleFieldHeight()-50);
}
```

antes de mandar seu robô andar escreva:

```
if(pertoParede())
          back(100);
else
          ahead(100);
```

Sites Úteis

- http://robowiki.net/wiki/
- http://www.slideshare.net/gscheibel/batalhas-com-robocode
- http://www.gsigma.ufsc.br/~popov/aulas/robocode/funcoes.html
- http://www.gsigma.ufsc.br/~popov/aulas/robocode/eventos.html
- http://robocode.cs.cityu.edu.hk/[Site de Competição].

Referências

- Larsen, F. N. “ReadMe for Robocode”, http://robocode.sourceforge.net/docs/ReadMe.html
- Souza, R. F. “Robocode”, http://www.slideshare.net/rosicleiafrasson/robocode-22009167
- Loadholtes, N. “IBM’s Robocode: A Platform for Learning AI”, http://ai-depot.com/articles/ibms-robocode-a-platform-for-learning-ai/
- Slides de aula (Regis Pires) http://www.slideshare.net/regispires/java-01-java-visao-geral-resumo
- O que é a tecnologia Java e por que é necessária? http://www.java.com/pt_BR/download/faq/whatis_java.xml
- Tutorial Java 3: Orientação a Objetos http://javafree.uol.com.br/artigo/871497/
- Apostila de Java: Orientação a Objetos http://www.slideshare.net/k19treinamentos/apostila-de-java-orientacao-a-objetos


That's all Folks!