

**Universidade do Minho**

Escola de Engenharia

Departamento de Informática

# Sistemas de Autónomos

Sistemas Autónomos

Perfil Sistemas Inteligentes @ MEI/MiEI 1º/4º – 2º semestre

Cesar Analide, Bruno Fernandes



# ISLab

Synthetic Intelligence Lab

## Programa

- Sistemas Robóticos e Sistemas Autónomos.
- Arquiteturas, Sensores e Atuadores.
- Perceção, Mobilidade e Navegação.
- Ambientes e Ferramentas de Simulação.
- Sensorização
  - Sensores
  - Sensorização Ambiente
  - Casos de Estudo
- Sensores de imagem e vídeo
  - Análise de imagem

## Sistema de Avaliação

- Componente de Trabalho em Grupo:
  - Plataforma de Sistemas Autónomos (RoboCode)
  - Sensorização Ambiente
- Componente Individual:
  - Análise de imagem

**Universidade do Minho**

Escola de Engenharia

Departamento de Informática

# **RoboCode**

## **Programação de Robôs**

Sistemas Autónomos

Perfil Sistemas Inteligentes @ MEI/MiEI 1º/4º – 2º semestre

Cesar Analide, Bruno Fernandes

## O que é RoboCode?

- O **ROBOCODE** é uma competição de programadores de robôs;
- O RoboCode é um (jogo) simulador de combate, inicialmente desenvolvido por Matthew Nelson (Alphaworks e IBM) desde 2000;

“A programming game that teaches Java in a fun, rewarding manner by letting you create Java ‘Robots’ [...] that battle it out onscreen against other robots.” (<http://www.alphaworks.ibm.com>)

alphaWorks  
is now  
developerWorks®  
**Open**



## Objetivos

- Desafio de programação de (representações de) robôs em JAVA;  
(versão .NET também disponível)
- O objetivo é o de construir/programar um robô para competir com outros, num recinto de batalha!  
(<http://robowiki.net/w/index.php?title=Robocode>)

Build the best – destroy the rest (<http://robocode.sourceforge.io>)



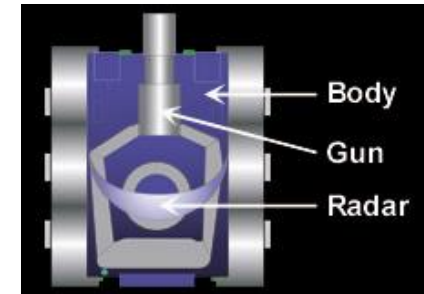


# ISLab

Synthetic Intelligence Lab

- Cada robô é representado por um **tanque de guerra**;
- Cada robô é constituído por:
  - corpo (body) com 6 rodas;
  - arma (gun);
  - radar;
  - sensores (internos e externos);

## Como funciona?





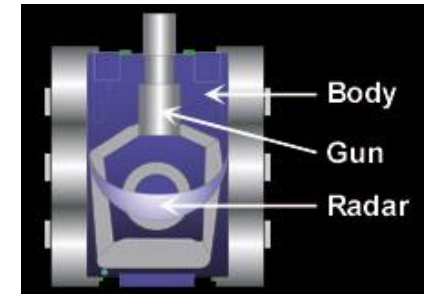


# ISLab

Synthetic Intelligence Lab

## Como funciona?

- Cada robô é representado por um **tanque de guerra**;
- Cada robô é constituído por:
  - corpo (body):
    - o corpo transporta a arma, que ainda suporta o radar;
    - o corpo é usado para mover o tanque para a frente e para trás, ou para o rodar (2 graus de liberdade);
  - arma (gun);
  - radar;
  - sensores (internos e externos);





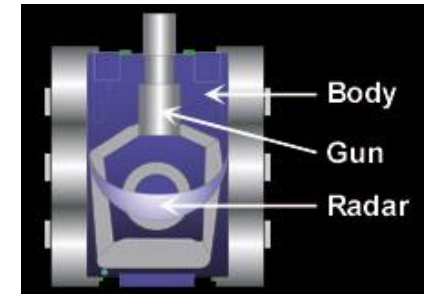


# ISLab

Synthetic Intelligence Lab

## Como funciona?

- Cada robô é representado por um **tanque de guerra**;
- Cada robô é constituído por:
  - corpo (body);
  - arma (gun):
    - a arma está montada em cima do corpo;
    - é usada para disparar balas de energia;
    - a arma pode rodar à direita e à esquerda (1 grau de liberdade);
  - radar;
  - sensores (internos e externos);



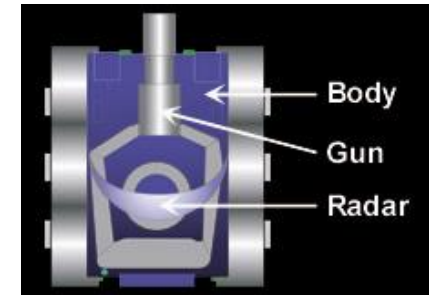


# ISLab

Synthetic Intelligence Lab

## Como funciona?

- Cada robô é representado por um **tanque de guerra**;
- Cada robô é constituído por:
  - corpo (body);
  - arma (gun);
  - radar:
    - o radar está montado em cima da arma;
    - é usado para rastrear outros tanques;
    - o radar pode rodar à direita e à esquerda (1 grau de liberdade);
    - o radar desencadeia um evento “OnScannedRobot” quando deteta robôs;
  - sensores (internos e externos);



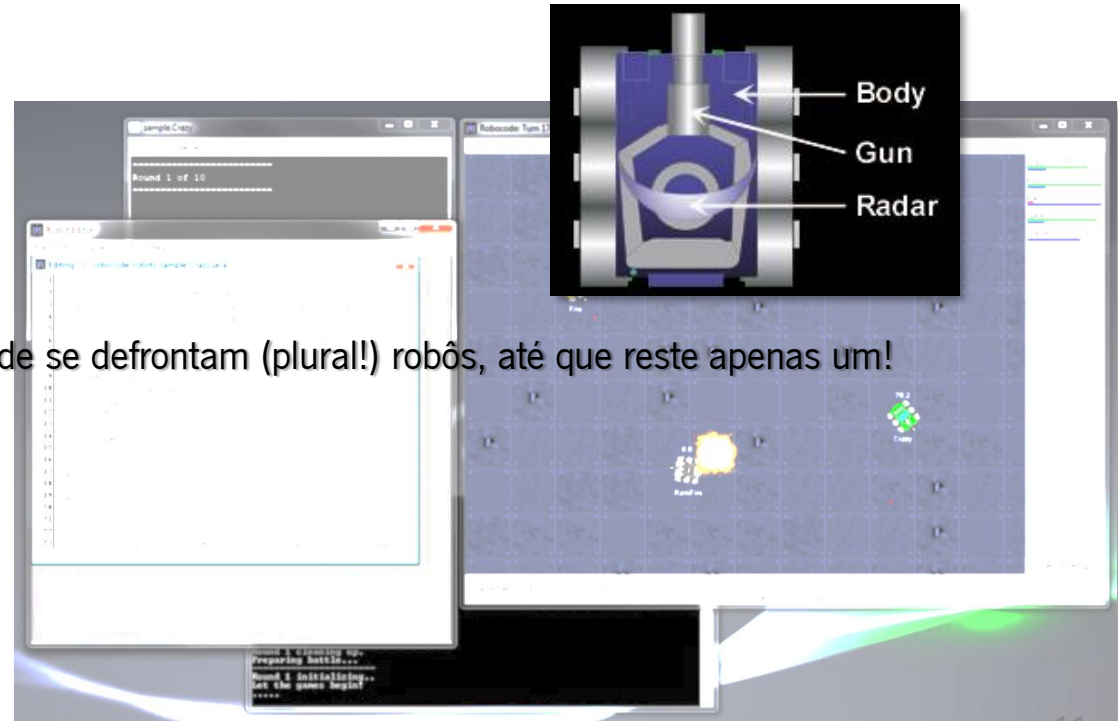


# ISLab

Synthetic Intelligence Lab

## Como funciona?

- Cada robô é representado por um **tanque de guerra**;
- Cada robô é constituído por:
  - corpo (body) com 6 rodas;
  - arma (gun);
  - radar;
  - sensores (internos e externos);
- Cada batalha decorre num recinto, onde se defrontam (plural!) robôs, até que reste apenas um!





# ISLab

Synthetic Intelligence Lab

## Quais os requisitos?

- Qualquer plataforma que suporte JAVA (ou .NET):
  - Windows, macOS, Linux;
- O conjunto de ferramentas RoboCode:
  - [http://sourceforge.net/projects/robocode/files/robocode sources](http://sourceforge.net/projects/robocode/files/robocode%20sources)
  - inclui JIKES, um compilador JAVA para RoboCode;



# ISLab

Synthetic Intelligence Lab

## Como começar?

- Existem 5 tipos de robôs diferentes (classes):
  - JuniorRobot:  
(<https://robocode.sourceforge.io/docs/robocode/robocode/JuniorRobot.html>)
    - versão simples de robô, em que cada invocação demora uma iteração (turn) e onde a execução das ações não retorna valores até que a iteração (turn) esteja concluída (blocking calls);
  - Robot;
  - AdvancedRobot;
  - TeamRobot;
  - Droid.



# ISLab

Synthetic Intelligence Lab

## Como começar?

- Existem 5 tipos de robôs diferentes (classes):

- JuniorRobot:

- (<https://robocode.sourceforge.io/docs/robocode/robocode/JuniorRobot.html>)

- versão simples de robô, em que cada invocação demora uma iteração (turn) e onde a execução das ações não retorna valores até que a iteração (turn) esteja concluída (blocking calls);

- Robot;

- AdvancedRobot;

- TeamRobot;

- Droid.

**Utilização não permitida!**





# ISLab

Synthetic Intelligence Lab

## Como começar?

- Existem 5 tipos de robôs diferentes (classes):
  - JuniorRobot;
  - Robot:  
<https://robocode.sourceforge.io/docs/robocode/robocode/Robot.html>
    - tipo básico de robô que deve ser estendido para o desenvolvimento de outros robôs;
  - AdvancedRobot;
  - TeamRobot;
  - Droid.





# ISLab

Synthetic Intelligence Lab

## Como começar?

- Existem 5 tipos de robôs diferentes (classes):
  - JuniorRobot;
  - Robot;
  - AdvancedRobot:  
<https://robocode.sourceforge.io/docs/robocode/robocode/AdvancedRobot.html>
    - extensão do Robot que suporta non-blocking calls;
    - permite reescrita de eventos;
  - TeamRobot;
  - Droid.



# ISLab

Synthetic Intelligence Lab

## Como começar?

- Existem 5 tipos de robôs diferentes (classes):
  - JuniorRobot;
  - Robot;
  - AdvancedRobot;
  - TeamRobot:  
<https://robocode.sourceforge.io/docs/robocode/robocode/TeamRobot.html>
    - suporta o envio de mensagens entre robôs de uma mesma equipa;
  - Droid.



# ISLab

Synthetic Intelligence Lab

## Como começar?

- Existem 5 tipos de robôs diferentes (classes):
  - JuniorRobot;
  - Robot;
  - AdvancedRobot;
  - TeamRobot;
  - Droid:  
<https://robocode.sourceforge.io/docs/robocode/robocode/Droid.html>
    - classe para robôs sem radar;
    - iniciam com maior pontuação (energia);
    - a principal motivação é a de serem utilizados na construção de equipas de robôs.

## Constituição do robô

### ■ Corpo:

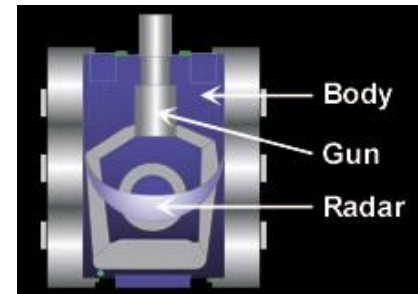
- As 6 rodas são “adorno”: não têm qualquer influência na deslocação;
- Deslocação do robô:

- avançar/recuar: ahead( Pixels ) / back( Pixels )
- velocidade máxima: 8 pixels/turn
- aceleração/travão: 1 / 2 pixels/turn;
- rodar: turnLeft( Graus ) / turnRight( Graus )
- taxa de rotação: dependente da velocidade ( 10-75%velocidade) graus/turn)  
(quanto mais rápido se desloca, mais demora a rotação)

### ■ Arma;

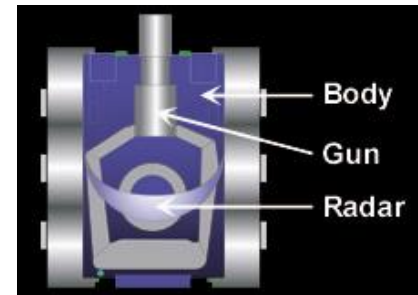
### ■ Radar;

### ■ Sensores.



## Constituição do robô

- Corpo;
- Arma:
  - Utilização essencial, tendo em conta o objetivo do jogo!
  - (... ou não!)
  - Utilização:
    - disparar: `fire( Power )`
    - rodar: `turnGunLeft( Graus ) / turnGunRight( Graus )`
  - Rotação: 20 graus/turn  
(adicionados à rotação do corpo)
- Radar;
- Sensores.



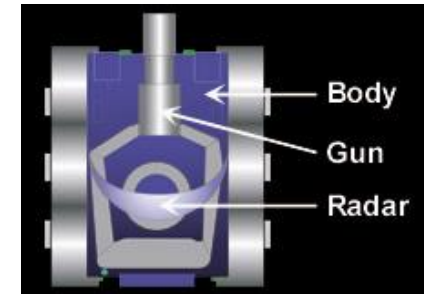


# ISLab

Synthetic Intelligence Lab

- Corpo;
- Arma;
- Radar:
  - Identificação do cenário e dos adversários:
    - movimento: `turnRadarLeft( Graus ) / turnRadarRight( Graus )`
    - rotação: `getRadarRotationRate( )`
  - Rotação: 45 graus/turn  
(adicionados à rotação da arma)
- Sensores.

## Constituição do robô



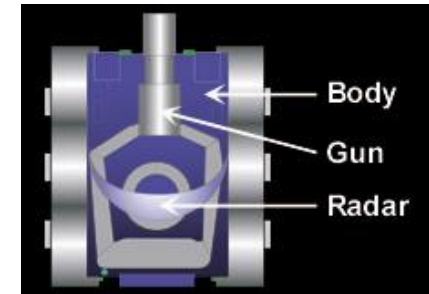


# ISLab

Synthetic Intelligence Lab

## Constituição do robô

- Corpo;
- Arma;
- Radar;
- Sensores: detecção de atividade;
  - Sensores internos:
    - energia: `getEnergy( )`
    - direção: `getHeading( ) / getRadarHeading( ) / getGunHeading( )`
    - posição: `getX( ) / getY( )`
    - velocidade: `getVelocity( )`
    - ...
  - Sensores externos.





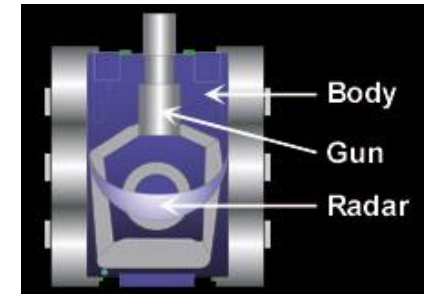


# ISLab

Synthetic Intelligence Lab

- Corpo;
- Arma;
- Radar;
- Sensores: detecção de atividade;
  - Sensores internos;
  - Sensores externos:
    - choque: `onHitWall( ) / onHitRobot( ) / ...`
    - tiro: `onHitByBullet( )`
    - radar: `onScannedRobot( )`
    - (identificados por eventos)

## Constituição do robô





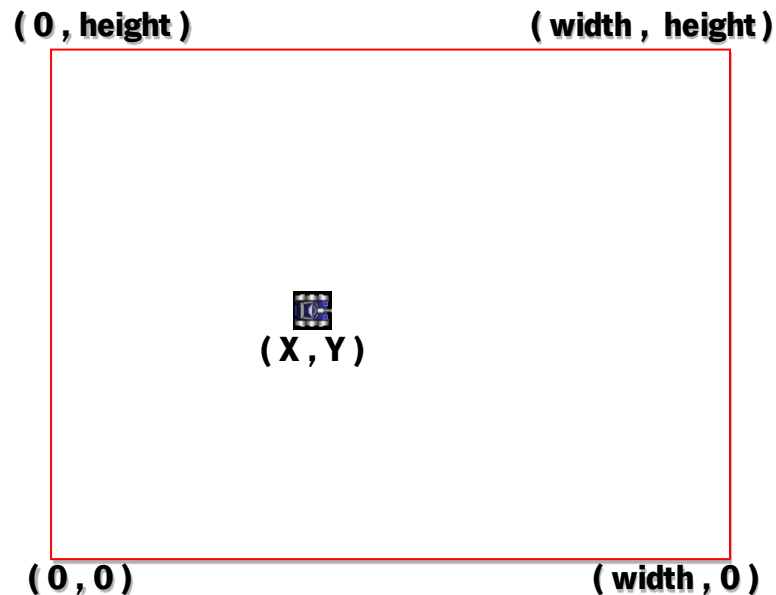
# ISLab

Synthetic Intelligence Lab

## ■ O recinto de batalha:

- É medido em píxeis;
- Por defeito, a dimensão é 800x600 píxeis;
- As coordenadas são dadas por ( x,y );
- Usa o sistema cartesiano de coordenadas, o que significa que o ponto **( 0,0 )** corresponde ao canto inferior esquerdo;
- A posição do robô é obtida por `getX( )` e `getY( )`;
- As dimensões do campo de batalha podem ser obtidas através de `getBattleFieldWidth( )` e `getBattleFieldHeight( )`;

## Características físicas do recinto



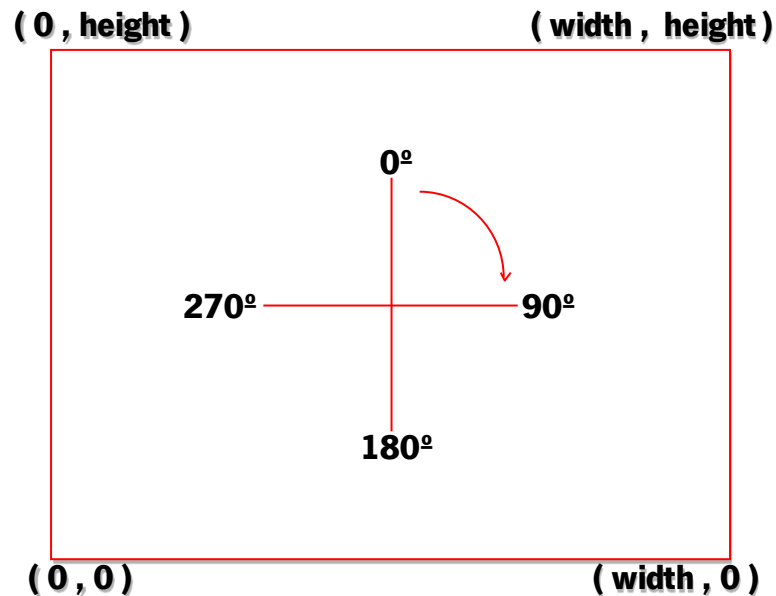


# ISLab

Synthetic Intelligence Lab

- No recinto de batalha:
  - Os ângulos são medidos em graus, evoluindo no sentido dos ponteiros do relógio;
  - $0^\circ$  ou  $360^\circ$  indicam o norte;
  - $90^\circ$  indicam o este;
  - $180^\circ$  indicam o sul;
  - $270^\circ$  indicam o oeste;
  - `getHeading()` devolve a posição, em graus, para onde a frente do robô está orientada;
  - `getGunHeading()` / `turnGunLeft()` / `turnGunRight()`;
  - `getRadarHeading()` / `turnRadarLeft()` / `turnRadarRight()`.

## Características físicas do recinto





## Características físicas

### ■ Energia:

- O robô utiliza (consome) energia durante a sua operação;
  - Inicia com 100 unidades de energia (120 para os droids);
  - Considera-se abatido quando a energia se esgota;
  - A sequência de iterações (turns) consome energia;
  - O disparo da arma consome energia;
  - Um tiro com sucesso é recompensado com recuperação de energia;
- `getEnergy( )` obtém a energia atual do robô (método na classe `robocode.Robot`);
- `getEnergy( )` obtém a energia atual do robô atingido (método na classe `robocode.HitRobotEvent`);
- ...



# ISLab

Synthetic Intelligence Lab

## Características físicas da arma

- A arma é caracterizada pela capacidade de disparo:
  - Depende da energia do tiro: fire( Power ), com  $0,1 \leq \text{Power} \leq 3$ ;
  - O dano infligido no outro robô é de  $4 \times \text{Power}$ ; se  $\text{Power} > 1$ , ainda adiciona  $(\text{Power} - 1) \times 2$ ;
  - Velocidade do disparo é  $20 - 3 \times \text{Power}$ ;
  - Se o disparo atinge o alvo, o robô incrementa a sua energia em  $3 \times \text{Power}$ ;



## Características físicas da arma

- A arma é caracterizada pela capacidade de disparo:
  - Depende da energia do tiro: `fire( Power )`, com  $0,1 \leq \text{Power} \leq 3$ ;
  - O dano infligido no outro robô é de  $4 \times \text{Power}$ ; se  $\text{Power} > 1$ , ainda adiciona  $(\text{Power} - 1) \times 2$ ;
  - Velocidade do disparo é  $20 - 3 \times \text{Power}$ ;
  - Se o disparo atinge o alvo, o robô incrementa a sua energia em  $3 \times \text{Power}$ ;
- ... e pela temperatura:
  - A temperatura da arma (`getGunHeat()`) é dependente da potência do disparo;
  - Após disparo, a temperatura sobe para:  $1 + (\text{Power} / 5)$ ;
  - A arma só volta a estar em condições de disparo quando a temperatura descer a zero; (no início de cada round, todas as armas estão “quentes”)
  - Em cada iteração (turn) a temperatura baixa 0,1 (ou valor diferente se definido no início da batalha - `getGunCoolingRate()`).





## Características físicas do movimento

- O movimento dos tanques é caracterizado por:
  - Aceleração e desaceleração:
    - é o sistema RoboCode que determina a aceleração com base na distância do movimento indicado;
    - os tanques aceleram a uma taxa de 1 pixel/turn/turn;
    - os tanques desaceleram a uma taxa de 2 pixels/turn/turn;
  - Velocidade:
    - a velocidade nunca poderá exceder 8 pixels/turn;
    - velocidade = aceleração x tempo;
  - Distância:
    - distância = velocidade x tempo;





## Características físicas do movimento de rotação

- A capacidade de rotação do corpo, da arma e do radar é dada por:
  - Rotação do corpo:
    - $(10 - 75\% \times |\text{velocidade}|)$  graus/turn;
    - quanto mais rápido se move, mais lenta é a rotação;
  - Rotação da arma:
    - 20 graus/turn;
    - este valor é adicionado à rotação do corpo;
  - Rotação do radar:
    - 45 graus/turn;
    - este valor é adicionado à rotação da arma;



## Exemplo (JAVA)

- Excerto de MyFirstRobot;
- Extensão da classe inicial Robot;
- Ações típicas:
  - avança e recua;
  - dispara quando deteta adversário;
  - muda de direção quando atingido.

```
import robocode.Robot;
...
public class MyFirstRobot extends Robot {
    public void run() {
        while (true) {
            ahead(100);
            turnGunRight(360);
            back(100);
            turnGunRight(360);
        }
    }
    public void onScannedRobot(ScannedRobotEvent e) {
        fire(1);
    }
    public void onHitByBullet(HitByBulletEvent e) {
        turnLeft(90 - e.getBearing());
    }
}
```



## Exemplo (JAVA)

### ■ Construção de um Robot:

#### ○ [AREA 1]:

- Espaço para declaração e inicialização de variáveis;
- Estas variáveis serão visíveis dentro do método run() ou em qualquer outro método usado ou definido;

```
import robocode.Robot;
...
public class MyFirstRobot extends Robot {
    [AREA 1]
    public void run() {
        [AREA 2]
        while (true) {
            [AREA 3]
        }
    }
    [AREA 4]
    public void onScannedRobot(ScannedRobotEvent e) {
        fire(1);
    }
}
```



## Exemplo (JAVA)

### ■ Construção de um Robot:

- [AREA 1];
- **[AREA 2]:**
  - O método run() é invocado pelo sistema para iniciar o ciclo de vida do robot;
  - O código escrito neste área será executado uma só vez, no início do ciclo de vida do robot;

```
import robocode.Robot;
...
public class MyFirstRobot extends Robot {
    [AREA 1]
    public void run() {
        [AREA 2]
        while (true) {
            [AREA 3]
        }
    }
    [AREA 4]
    public void onScannedRobot(ScannedRobotEvent e) {
        fire(1);
    }
}
```



## Exemplo (JAVA)

### ■ Construção de um Robot:

- [AREA 1];
- [AREA 2];
- **[AREA 3]:**
  - Esta é a segunda parte do método run();
  - O ciclo infinito desta área promove a execução contínua do comportamento programado do robot;

```
import robocode.Robot;
...
public class MyFirstRobot extends Robot {
    [AREA 1]
    public void run() {
        [AREA 2]
        while (true) {
            [AREA 3]
        }
        [AREA 4]
        public void onScannedRobot(ScannedRobotEvent e) {
            fire(1);
        }
    }
}
```



## Exemplo (JAVA)

### ■ Construção de um Robot:

- [AREA 1];
- [AREA 2];
- [AREA 3];
- **[AREA 4]:**
  - É nesta área onde se definem outros métodos a utilizar durante o ciclo de vida do método run();
  - Pode-se utilizar eventos do sistema ou programar métodos próprios;
  - No exemplo ao lado, usa-se o evento ScannedRobot para disparar quando o radar deteta outro robot;

```
import robocode.Robot;
...
public class MyFirstRobot extends Robot {
    [AREA 1]
    public void run() {
        [AREA 2]
        while (true) {
            [AREA 3]
        }
    }
    [AREA 4]
    public void onScannedRobot(ScannedRobotEvent e) {
        fire(1);
    }
}
```



# ISLab

Synthetic Intelligence Lab

## Referências eletrônicas

- RoboCode Home na SourceForge:
  - <http://robocode.sourceforge.net>
- RoboWiki:
  - <http://robowiki.net/w/index.php?title=Robocode>
- Download:
  - <http://sourceforge.net/projects/robocode/files>
- RoboCode na WikiPédia:
  - <http://en.wikipedia.org/wiki/Robocode>



**Universidade do Minho**

Escola de Engenharia

Departamento de Informática

## Sistemas Autónomos

Perfil Sistemas Inteligentes @ MEI/MiEI 1º/4º – 2º semestre

Cesar Analide, Bruno Fernandes