FURBOT

Prof. Maurício Capobianco Lopes

Profª. Luciana Pereira de Araújo

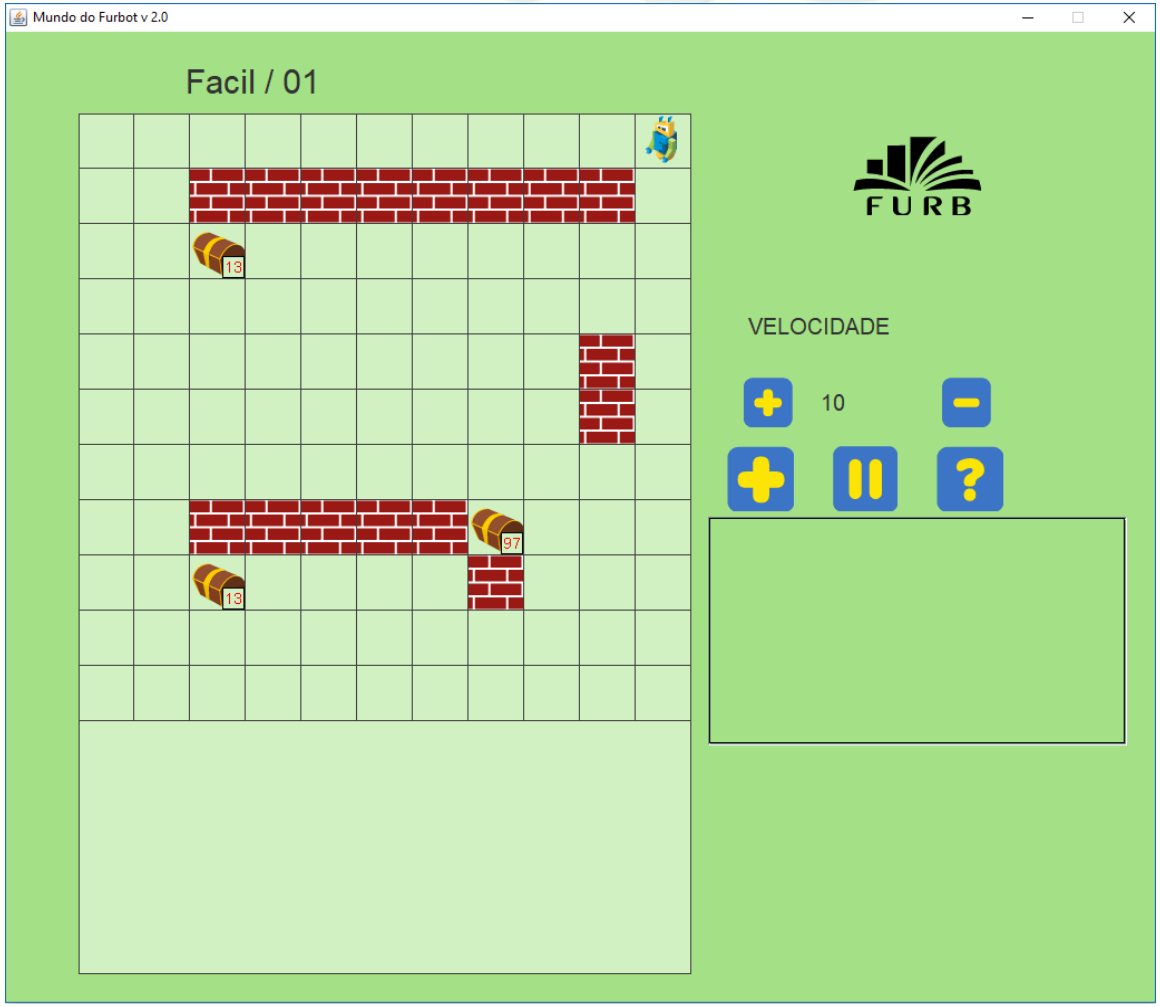
(Material de Apoio)

1. O FURBOT

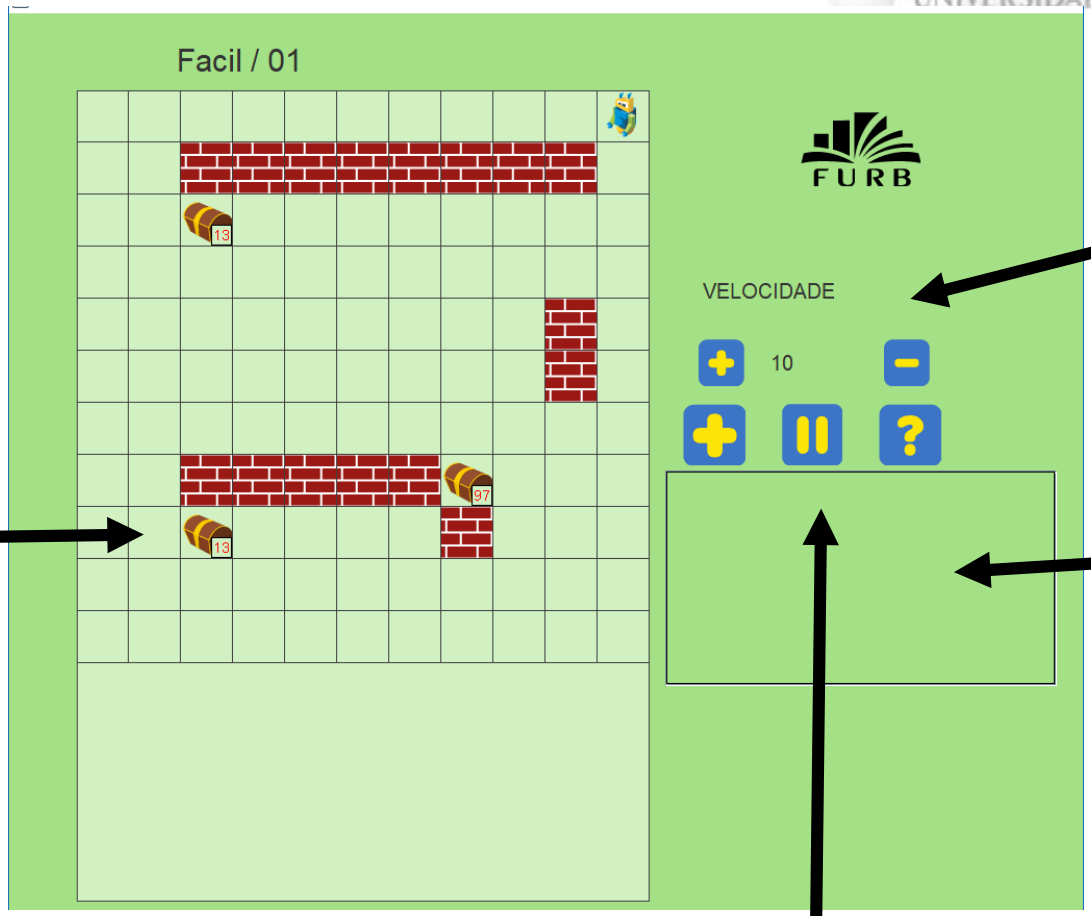
- Ambiente de apoio ao ensino de lógica de programação desenvolvido na FURB em 2008
- Disponibiliza um ambiente que possibilita o desenvolvimento de algoritmos de controle de personagens de tal forma a criar uma atmosfera facilitadora ao aprendizado

1.1 FURBOT – CONCEITOS INICIAIS

- O Furbot é formado por um ambiente composto por linhas e colunas no qual habita um robô e demais personagens.



1.1 FURBOT – CONCEITOS INICIAIS



Velocidade

Cenário/
Mundo

Saídas

Iniciar

1.1 FURBOT – CONCEITOS INICIAIS

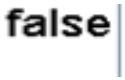
- O cenário é criado a partir de um arquivo XML

```
<furbot>
      <enunciado>
         Faça o robô andar até a posição (7,4). &lt;br&gt;
         Lembre-se de que as coordenadas sempre serão
         fornecidas como (x, y).&lt;br&gt;
         A primeira coluna e linhas são a de número ZERO.
      </enunciado>

      <mundo>
            <qtdadeLin>8</qtdadeLin>
            <qtdadeCol>8</qtdadeCol>
            <explodir>true</explodir>
      </mundo>

      <robo>
            <x>0</x>
            <y>0</y>
      </robo>
</furbot>
```

1.2 FURBOT – PERSONAGENS



- Furbot: é o robô que recebe as missões a serem executadas no mundo;
- Alien: inimigo do robô, está no mundo para impedi-lo de realizar as missões;
- Booleano: identificadores lógicos (true/false);
- Numero: números utilizados durante a realização das missões;
- Tesouro: baú que guarda tesouros, os quais devem ser encontrados pelo Furbot;
- Parede: obstáculo que impede a passagem.

UTILIZANDO O FURBOT

2. FURBOT – PRIMEIROS PASSOS

```
1  import br.furb.furbot.Furbot;
2  import br.furb.furbot.MundoVisual;
3
4  public class Exemplo extends Furbot {
5
6      @Override
7      public void inteligencia() throws Exception {
8          //código-fonte
9      }
.0
.1     public static void main(String[] args) {
.2         MundoVisual.iniciar("Nome.xml");
.3     }
.4
.5 }
.6
```

Método de inteligência do Robô

Método de inicialização

2. FURBOT – PRIMEIROS PASSOS


```
1  import br.furb.furbot.Furbot;
2  import br.furb.furbot.MundoVisual;
3
4  public class Exemplo extends Furbot {
5
6      @Override
7      public void inteligencia() throws Exception {
8          //código-fonte
9      }
.0
.1     public static void main(String[] args) {
.2         MundoVisual.iniciar("Nome.xml");
.3     }
.4
.5 }
.6
```

Importações

Definição da Classe

Nome do XML

COMANDOS DO FURBOT

3.1 FURBOT – COMANDOS INICIAIS

- andarDireita():void
  - Faz com que o robô ande uma célula à direita. Se houver um obstáculo ou for o fim do mundo, o robô explode;
- andarEsquerda():void
  - Faz com que o robô ande uma célula à esquerda. Se houver um obstáculo ou for o fim do mundo, o robô explode;
- andarAcima():void
  - Faz com que o robô ande uma célula para cima. Se houver um obstáculo ou for o fim do mundo, o robô explode;
- andarAbaixo():void
  - Faz com que o robô ande uma célula para baixo. Se houver um obstáculo ou for o fim do mundo, o robô explode.

3.1 FURBOT – COMANDOS INICIAIS

- diga (String):void
  - Faz com que a mensagem enviada como parâmetro seja apresentada na tela. Toda mensagem deve ser escrita entre os parêntesis e grafada entre aspas duplas;
- diga (Object):void
  - Permite que seja passado como parâmetro um objeto que contém a mensagem a ser apresentada na tela. Este recurso é avançado e você só vai utilizá-lo mais adiante;
- limparConsole():void
  - Limpa a área de mensagens do mundo. Geralmente utilizamos para facilitar a visualização de mensagens importantes.

3.2 FURBOT – COMANDOS DE AVALIAÇÃO DA SITUAÇÃO

- ehVazio (Direcao direcao) : boolean
  - Método que verifique se a célula indicada por **direção** é vazia
  - Neste caso o furbot retorna um valor verdadeiro (true)
  - Se está ocupada o furbot retorna um valor falso (false)
  - O método não indica quem está ocupando aquela posição
  - Se a célula na direção apontada fica além dos limites do mundo, o método retorna verdadeiro
- ehFim (Direcao direcao) : boolean
  - Método que verifica se a célula na direção passada como parâmetro fica além do mundo.
  - Se ela for além do mundo retorna verdadeiro (true) senão retorna falso (false)

3.2 FURBOT – COMANDOS DE AVALIAÇÃO DA SITUAÇÃO

- ehObjetoDoMundoTipo (String classe, Direcao direcao) : boolean
  - Método que retorna verdadeiro caso existe um objeto do tipo passado como parâmetro na direção informada
  - Ex. ehObjetoDoMundoTipo("Alien", ACIMA)

3.2 FURBOT – COMANDOS DE AVALIAÇÃO DA SITUAÇÃO

- Direcao
  - Indica o caminho a ser testado pelo robô
    - DIREITA: refere-se à posição à direita do robô
    - ESQUERDA: refere-se à posição à esquerda do robô
    - AQUIMESMO: refere-se à posição onde está o robô
    - ABAIXO: refere-se à posição abaixo do robô
    - ACIMA: refere-se à posição acima do robô

3.2 FURBOT – COMANDOS DE AVALIAÇÃO DA SITUAÇÃO

- ehDependenteEnergia(): boolean
  - Verifica se o objeto em questão depende de energia
- ehBloqueado(): boolean
  - Verifica se o objeto é bloqueado
- jahExplodiu():
  - Verifica se o objeto já explodiu
- chegouUmObjetoNaPosicao (ObjetoDoMundo objetoDoMundo): void
  - Verifica se algum outro objeto passou na posição atual

3.3 FURBOT – COMANDOS DE POSIÇÃO

- getX() : int
  - Método que retorna um valor inteiro contendo o número da coluna em que está o robô
  - A primeira coluna é ZERO
- getY() : int
  - Método que retorna um valor inteiro contendo o número da linha em que está o robô
  - A primeira linha é ZERO

3.4 COMANDOS DE ESTADO

- getEnergia(): int
  - Retorna a energia atual do objeto
- getMaxEnergia(): int
  - Retorna a quantidade máxima de energia que o objeto pode ter
- getSouDoTipo(): String
  - Retorna uma String indicando qual é o tipo do objeto

3.4 FURBOT – COMANDOS DE MANIPULAÇÃO DE OBJETOS

- getObjeto(Direcao direcao) : ObjetoMundo
  - Método que retorna o objeto da célula indicada em direção
  - Se não tiver nada nessa célula, ou for além dos limites do mundo, retorna null

3.4 FURBOT – COMANDOS DE MANIPULAÇÃO DE OBJETOS

- adicionarObjetoNoMundo (ObjetoDoMundo objeto, Direcao direcao) : void
  - adiciona um objeto no mundo na posição relativa à direção informada.
- adicionarObjetoNoMundoXY(ObjetoDoMundo objeto, int X, int Y):
  - adiciona um objeto no mundo na coordenada x,y.
- getObjetoXY(int x, int y): T
  - Obtém um objeto genérico contido na posição (x, y)

3.4 FURBOT – COMANDOS DE MANIPULAÇÃO DE OBJETOS

- removerObjetoDoMundo(ObjetoDoMundo objeto) : void
  - remove um objeto do mundo passado como parâmetro
- removerMe(): void
  - remove o objeto que chamou o método

3.4 FURBOT – COMANDOS DE MANIPULAÇÃO DE OBJETOS

- bloquear(): void
  - Faz com que o estado do objeto seja bloqueado
- desbloquear(): void
  - Faz com que o estado do objeto seja desbloqueado

3.4 FURBOT – COMANDOS DE MANIPULAÇÃO DE OBJETOS

- getTempoEspera(): int
  - Retorna o tempo de espera do objeto
- setTempoEspera(int milisegundos): void
  - Define o tempo de espera do objeto
- esperar(int segundos): void
  - Entra em estado de espera pelo tempo definido
- esperarAlguem(): ObjetoDoMundo
  - Espera até que um objeto se posicione em cima do objeto atual

3.4 FURBOT – COMANDOS DE TECLA

- getUltimaTeclaPress(): int
  - Retorna um inteiro referente a última tecla pressionada, podem ser verificadas utilizando a enumeração Teclas.
- Teclas:
  - TECLACIMA
  - TECLADIREITA
  - TECLAESQUERDA
  - TECLAESPACO

CRIAÇÃO DE OBJETOS

4. Criação de Objetos

- O FURBOT permite a criação de objetos que podem ser inseridos em qualquer local do mapa
- Para serem inseridos os objetos precisam ser criados usando o método new()
- O método new() cria um endereço de memória (handle) para o objeto e permite que o mesmo possa receber e informar valores e proceder cálculos ou operações
- O endereço de memória alocado para o objeto deve ser armazenado em uma variável do tipo da classe do objeto que se deseja criar

4. Criação de Objetos

- Sintaxe: `Nome_Classe nome_objeto = new Nome_Classe();`

- A sintaxe é utilizada para a criação de qualquer tipo de objeto, sendo eles do Furbot ou não.

4. Criação de Objetos

- Exemplos:
  - Para criar um numero

```
Numero numero = new Numero();
```

  - Para criar um booleano

```
Booleano logico = new Booleano();
```

  - Para criar um alien

```
Alien alien = new Alien();
```

  - Para criar um tesouro

```
Tesouro tesouro = new Tesouro();
```

  - Para criar uma parede

```
Parede parede = new Parede();
```

4. Criação de Objetos

- Ao criar os objetos Numero, Booleano e Tesouro é necessário definir o seu conteúdo utilizando o método setValor()
- Exemplos:
  - Para criar e definir o valor de um Numero

```
Numero numero = new Numero();
numero.setValor (10);
```

  - Para criar e definir o valor de um Booleano

```
Booleano logico = new Booleano();
logico.setValor (true);
```

4. Criação de Objetos

- Para adicionar ou remover objetos do mundo podem ser utilizados os métodos:
  - adicionarObjetoNoMundo (ObjetoDoMundo objeto, Direcao direcao) : void
    - adiciona um objeto no mundo na posição relativa à direção informada.
  - adicionarObjetoNoMundoXY(ObjetoDoMundo objeto, int X, int Y):
    - adiciona um objeto no mundo na coordenada x,y.
  - removerObjetoDoMundo(ObjetoDoMundo objeto) :
    - remove um objeto do mundo passado como parâmetro

4. Criação de Objetos

- Exemplos:
  - Para criar, definir e adicionar um Numero

```
Numero numero = new Numero();
numero.setValor (10);
adicionarObjetoNoMundo (numero,
AQUIMESMO);
```

  - Para criar e adicionar um Alien

```
Alien alien = new Alien();
adicionarObjetoNoMundoXY (alien, 0, 0);
```

5. OUTROS COMANDOS

- repintar(): void
  - Atualiza a tela / frame
- buildImage(): ImageIcon
  - Muda a imagem do personagem
- sortear(): int
  - Sorteia um número aleatório
- executar(): void
  - Faz o personagem executar o método inteligência