Programação Orientada a Objetos

Aula 6 – Introdução ao Java (cont)

Programação Orientada a Objetos
2.o semestre de 2003
Fábio Roberto de Miranda

Escopo de variáveis

```
class Ponto {
    /**
     *  Class Ponto com construtores
     */
    int x;
    int y;
    Ponto(int vx, int vy){
        x = vx;
        y = vy;
    }
    Ponto(){
    }
}
```

Escopo de variáveis

- E se quiséssemos que os argumentos do construtor de Ponto também se chamassem x e y, assim como as variáveis locais?
- Poderíamos querer fazer isso para que os nos Javadocs saiam os nomes dos argumentos como x e y, ficando mais intuitivo para o usuário.

Escopo de variáveis

- Ao mencionarmos uma variável dentro de um método, variáveis são buscadas no escopo local do método antes de entre as propriedades da classe.
- No exemplo abaixo, dentro do método "escondemos" os x e y da classe

```
Ponto(int x, int y){
  // Os atributos x e y da classe
  foram "esquecidos"
}
```

O operador this

- Dentro de uma classe, podemos preceder um atributo do operador this quando quisermos ganhar acesso a uma das propriedades da classe (declarada fora dos métodos, no corpo da classe)
- Podemos também usá-lo para invocar construtores

Uso do operador this

```
class Ponto {
    int x;
    int y;
    Ponto(int x, int y){
    /* this.x corresponde ao x que é
  propriedade da classe */
        this.x = x;
        this.y = y;
    }
    Ponto(){
   /* Invoca o construtor Ponto(int x, int y)*/
        this(0,0);
    }
}
```

Contexto estático x dinâmico

Veja o código de Endereço.java e diga por que a chamada para imprime() não dá certo

```
public static void main(String[] args){
       // Não dá certo
       // imprime();

}
```

Variáveis final

- Uma variável ou referência declarada final em Java não pode ter seu valor posteriormente alterado. Ex (da classe Logradouro.java):

```
static final int PRAÇA = 0;
```

- São usadas para implementar constantes

Comparação de objetos

- Em Java, a comparação de duas variáveis de tipos primitivos (int, double, etc) com o operador == diz se o conteúdo é o mesmo. Ex.:

```
int i = 0;
int j = 1;
if (i == j){
...
}
```

Comparação de objetos

- Quando aplicamos os operadores == a objetos, estamos testando se os objetos são exatamente os mesmos (na memória).
- Cuidado ao comparar duas *String*s. Uma comparação correta deve utilizar o método .equals().

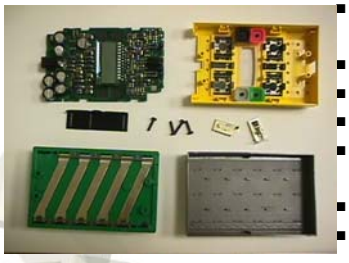
Epílogo da aula I: Trabalhando no espaço da solução

- O programa a seguir mostra uma solução em Java para fazer um pequeno robô móvel evitar obstáculos
- Para saber mais sobre robôs LEGO e Java, visite http://www.lejos.org
- Observe o uso de objetos de classes pertinentes ao problema

O RCX

- Microcontrolador encapsulado num bloco Lego
- Três portas de entrada (analógicas)
- Três portas de saída (analógicas)
- Tem portas de comunicação infravermelha, para comunicação com o desktop ou com outros RCXs

Raio X do RCX



- Microcontrolador Hitachi H8/3297
- 16K de ROM
- 32K de RAM
- 16MHz
- 8 timers de 8 bits, 1 de 16 bits
- 1 serial
- Portas A/D de 8 bits

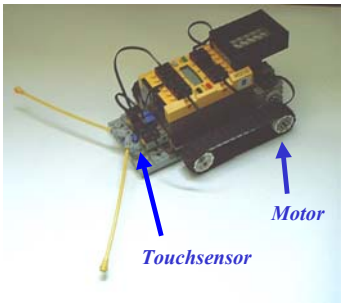
Mas... Java em 32K?

- Na verdade 20K
- Esta restrição não impediu o aparecimento de duas VMs para o RCX: TinyVM e LeJOS.
- Firmware original também é em bytecodes

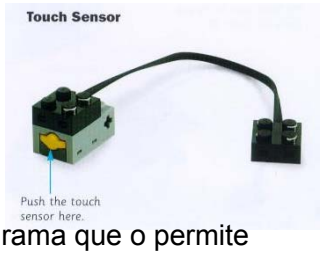
LeJOS: www.lejos.org

Exemplo

- Este robô tem dois sensores de toque (touchsensors) na frente
- Também tem um motor em cada roda traseita



Exemplo



- Veremos um programa que o permite utilizar a informação dos *touchsensors* para desviar de obstáculos (movendo-se com os motores)

Código - fonte

```
import ukcrobots.core.*;
import ukcrobots.util.*;
/**
 * Programa para controle de um robô
  baseado na API da U. Kentucky
 */
public class NewAvoid(){
    public static void main(String[] args){

  //Permite "esperar" algo acontecer
     Sleeper sleeper = new Sleeper();
```

Código – fonte (parte 2)

```
// Cria os sensores
/* Botão do lado esquerdo */
TouchSensor botãoEsquerdo = new
      TouchSensor(1);
/* Botão do lado direito */
TouchSensor botãoDireito = new
      TouchSensor(3);

// Cria os motores
Motor motorEsquerdo = new Motor('A');
Motor motorDireito = new Motor('C');

// Aciona os motores
motorEsquerdo.forward();
motorDireito.forward();
```

Código – fonte (parte 3)

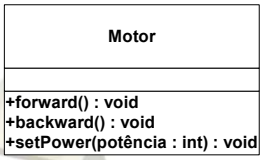
```
while (true) {
      Sleeper.sleep(200); // pausa
   // Quando a esquerda do robô bate
      if (botãoEsquerdo.isPressed()){
        // Recua os dois motores
        motorEsquerdo.backward();
        motorDireito.backward();
        Sleeper.sleep(300);

        motorDireito.backward();
        motorEsquerdo.forward();
        Sleeper.sleep(300);

   //Retoma o movimento anterior
        motorEsquerdo.forward();
        motorDireito.forward();
   }
```

O espaço de soluções

- Como montar os objetos para torná-los úteis?

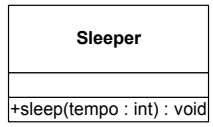


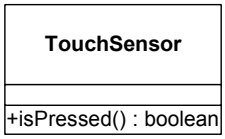
Exemplo de uso da classe Motor.:

```
Motor m = new
Motor('A');
m.forward();
```

Outros objetos do exemplo

- Sleeper
- TouchSensor





Algumas informações das classes foram omitidas

Onde posso achar mais objetos úteis?

- Um bom começo são os javadocs de uma dada API
- Javadocs são uma documentação gerada automaticamente a partir dos códigos-fonte das classes da API (vejamos alguns javadocs)
- Há APIs em Java que não vêm nos Javadocs (muitas)