

Procedimentos e escopo global

- Agora vamos adicionar *procedimentos* a TINY, usando a sintaxe abaixo:

```
TINY  -> PROCS ; CMDS
      | CMDS
PROCS -> PROCS ; PROC
      | PROC
PROC  -> procedure id ( ) CMDS end
CMD   -> id ( )
      | ...
```

- Nomes de procedimentos vivem em um *espaço de nomes* separado do nome de variáveis, e são visíveis em todo o programa
- Variáveis visíveis em todo o bloco principal do programa também são visíveis dentro de procedimentos (variáveis globais)

Exemplo – escopo de procedimentos

- Procedimentos podem ser mutuamente recursivos

```
procedure par()  
  if 0 < n then  
    n := n - 1;  
    impar()  
  else  
    res := 1  
  end  
end;
```

```
procedure impar()  
  if 0 < n then  
    n := n - 1;  
    par()  
  else  
    res := 0  
  end  
end;
```

```
var x, n, res;  
read x;  
n := x;  
par();  
write res;  
n := x;  
impar();  
write res
```

variáveis globais

*visibilidade
retrativa*

Analizando escopo global

- Para termos escopo global, precisamos fazer a análise semântica em duas *passadas*
 - A primeira coleta todos os nomes que fazem parte do escopo global, e detecta declarações duplicadas
 - A segunda verifica se todos os nomes usados foram declarados
- A primeira passada constrói uma tabela de símbolos que é usada como entrada para a segunda
- No caso de TINY, essa tabela de símbolos é diferente da que usamos para variáveis

Escopos em MiniJava

- MiniJava tem vários tipos de nomes:

- Variáveis

- Campos

- Métodos

- Classes

) mesmo espaço de nomes

- espaço de nomes próprio

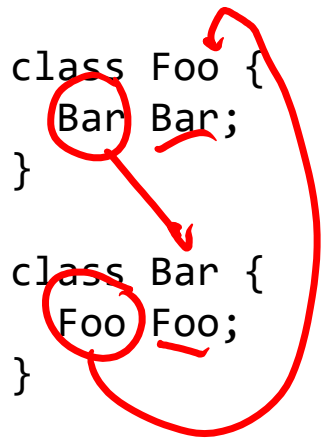
- espaço de nomes próprio

- Cada um desses tem suas regras de escopo; alguns compartilham espaços de nomes, outros têm espaços de nomes separados

Classes

- O escopo das classes é *global*
- Uma classe é visível no corpo de qualquer outra classe
- Classes estão em seu próprio espaço de nomes → *recursive*

```
class Foo {  
  Bar Bar;  
}  
  
class Bar {  
  Foo Foo;  
}
```



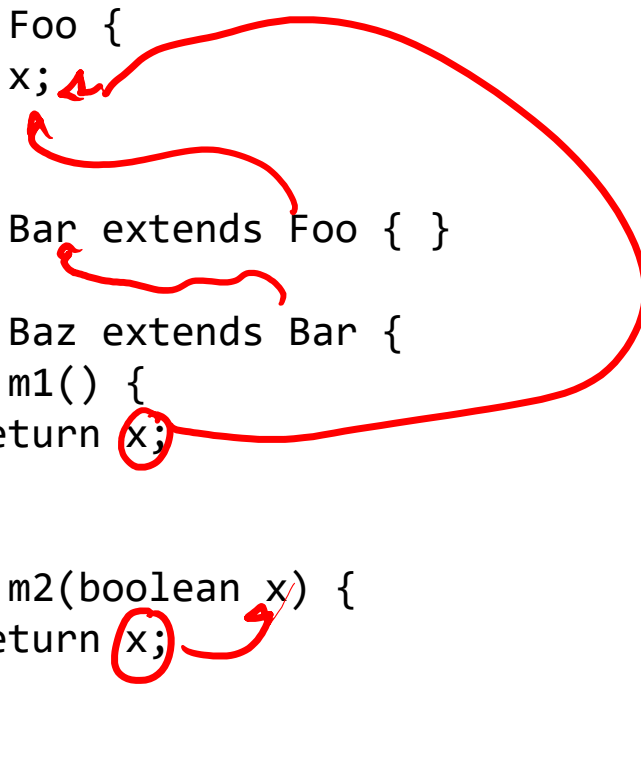
Variáveis e campos

- Variáveis e campos compartilham o mesmo espaço de nomes, mas as regras de escopo são diferentes
- O escopo de variáveis locais é o escopo de bloco tradicional
- O escopo de campos respeita a *hierarquia de classes* de MiniJava, uma relação dada pelas cláusulas *extends* usadas na definição das classes
- Um campo de uma classe é visível em todos os métodos daquela classe e de *todas as suas subclasses, diretas ou indiretas*
- Variáveis locais ocultam campos, mas campos não podem ser redefinidos nas subclasses

Exemplo – escopo de variáveis e campos

- O escopo do campo `x` inclui todas as subclasses de *Foo*

```
class Foo {  
    int x;  
}  
  
class Bar extends Foo { }  
  
class Baz extends Bar {  
    int m1() {  
        return x;  
    }  
  
    int m2(boolean x) {  
        return x;  
    }  
}
```



Métodos

- Como classes, métodos estão em seu próprio espaço de nomes
- Mas, como campos, o escopo de um método é a classe em que está definido e suas subclasses
- Um método não pode ser definido duas vezes em uma classe, mas pode ser redefinido em uma subclasse contanto que a assinatura seja a mesma
- A *assinatura* do método é o seu tipo de retorno, seu nome e os tipos dos seus parâmetros, na ordem na qual eles aparecem
- Como classes, métodos e campos podem ser referenciados antes de sua declaração, a verificação de escopo de MiniJava também ocorre em duas passadas

Exemplo - métodos

- O método *m2* é visível em Baz, que redefine *m1*

```
class Foo {  
    int m1() {  
        return 0;  
    }  
    int m2() {  
        return 1;  
    }  
}  
class Bar extends Foo { }  
class Baz extends Bar {  
    int m1() {  
        return this.m2();  
    }  
}
```