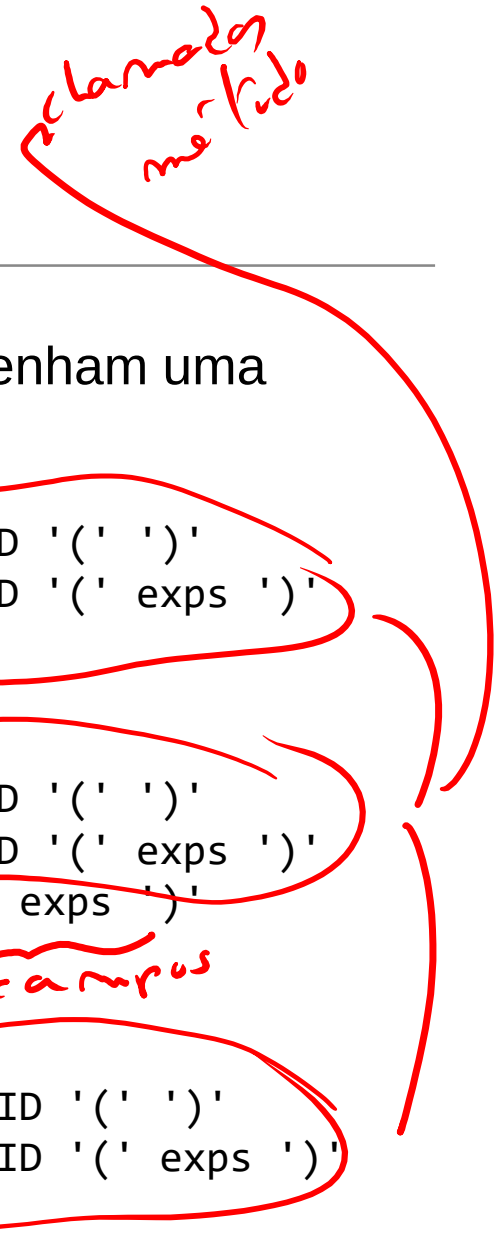
Classes

- Para nossos registros virarem classes precisamos que eles tenham uma *superclasse* e que tenham *métodos:*

```
s : classes ';' procs ';' cmds
  | classes ';' cmds
  | procs ';' cmds
  | cmds
  ;

classes : classes ';' classe
        | classe
        ;

classe : CLASS ID BEGIN decls END
       | CLASS ID BEGIN decls ';' procs END
       | CLASS ID BEGIN procs END
       | CLASS ID ':' ID BEGIN decls END
       | CLASS ID ':' ID BEGIN decls ';' procs END
       | CLASS ID ':' ID BEGIN procs END
       ;
```

```
cmd : <outros>
    | rexp '.' ID '(' ')'
    | rexp '.' ID '(' exps ')'
    ;

exp : <outras>
    | rexp '.' ID '(' ')'
    | rexp '.' ID '(' exps ')'
    | NEW ID '(' exps ')'
    ;

rexp : <outras>
     | rexp '.' ID '(' ')'
     | rexp '.' ID '(' exps ')'
     ;
```



Classes

- Vamos usar as mesmas regras para subtipagem e redefinição de métodos de MiniJava
- Quando instanciamos um objeto precisamos alocar espaço para todos os seus campos (próprios ou herdados)
- Ler e escrever campos em um objeto são operações análogas à ler e escrever campos de um registro (de fato, podemos usar as mesmas operações **ldfld** e **stfld**)
- Mas métodos são diferentes, por causa da redefinição: precisamos de *vtables*

Vtables

- Uma vtable é um vetor que contém o endereço de todos os métodos de uma classe
- A vtable de uma subclasse começa igual à de sua superclasse direta
  - Novos métodos aumentam a vtable
  - Métodos redefinidos substituem entradas que já estão na vtable
- Todo objeto contém um ponteiro para a vtable de sua classe
- Toda chamada a método usa a vtable do objeto