Compiler Practical 2013

Inheritance (Part 1)

Berthold Hoffmann (B. Gersdorf, T. Röfer)

hof@informatik.uni-bremen.de
Cartesium 2.48
1. Inheritance and Derived Classes
2. Storage Organisation
3. Lexical and Syntax Analysis
4. Context Analysis
5. Bonus Task: Extension by Access Protection
Inheritance, Derivation

• Allows to model the *is-a* relation between Classes
  – If class A inherits from class B, A is the *subclass / derived class* of the *superclass / base class* B

• Goals
  – Reuse of code
  – Support of *polymorphism*
Virtual Methods (Polymorphism)

• Points of view
  – A derived class *extends* ist base class
  – A base class is more general, its derived class is more special

• Real /virtual inheritance
  – A method is *really* inherited if the derived class reuses the method
  – A method is *virtually* inherited if the derived class overrides the method.
Speicheroorganisation

<table>
<thead>
<tr>
<th>Address</th>
<th>Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3-2</td>
<td>SELF</td>
</tr>
<tr>
<td>R3-1</td>
<td>return address</td>
</tr>
<tr>
<td>R3</td>
<td>predecesor frame</td>
</tr>
<tr>
<td>R3+1</td>
<td>a</td>
</tr>
<tr>
<td>R3+2</td>
<td>b</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Heap</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>A.c</td>
</tr>
<tr>
<td>n+1</td>
<td>A.d</td>
</tr>
<tr>
<td>n+2</td>
<td>B.d</td>
</tr>
<tr>
<td>n+3</td>
<td>B.e</td>
</tr>
</tbody>
</table>

CLASS A IS
    c, d : Integer;
END CLASS

CLASS B EXTENDS A IS
    d, e : Boolean;
METHOD f IS
    a : A;
    b : B;
    BEGIN
        b := NEW B;
        a := b;
    END METHOD
END CLASS

Storage extract with control flow at this place
Lexical and Syntax Analysis

• Lexical analysis
  – EXTENDS

• Syntax analysis
  – Extend grammar
  – ClassDeclaration needs a baseType attribute
  – Without EXTENDS, Object is the base class

```plaintext
classdecl ::= CLASS identifier [ EXTENDS identifier ] IS
            { memberdecl }
END CLASS
```
• Predefined Classes
  – New predefined class *Object*
  – The only class without a base class
  – Has neither attributes, nor methods (although it might have ...)
  – *Integer* and *Boolean* inherit from *Object*

• Resolving base classes
  – Cycles are forbidden
  – Afterwards, *Program.classes* is an acyclic graph
Context Analysis, Subclasses

CLASS A IS
END CLASS

CLASS B
EXTENDS C IS
END CLASS

CLASS C
EXTENDS A IS
END CLASS

Program

ClassDeclaration
identifier: „A“

ClassDeclaration
identifier: „B“

ClassDeclaration
identifier: „C“
• Check base class before actual class
• Actual class inherits *Declarations* of the base class and extends them
  – *Object* „inherits“ visibility of class names
• Storage offset for attributes starts after the last offset for the base class
  – With *Object*, it starts after *HEADERSIZE*
• Extensions of *ClassDeclaration.isA(...)*

```plaintext
CLASS A IS
  c, d : Integer;
END CLASS

CLASS B EXTENDS A IS
  d, e : Boolean;
METHOD f IS
  a : A;
  b : B;
BEGIN
  b := NEW B;
  a := b;
END METHOD
END CLASS
```
Management of Declarations

CLASS A IS
    c, d : Integer;
END CLASS

CLASS B EXTENDS A IS
    d, e : Boolean;
    METHOD f IS
        a : A; b: B;
        BEGIN
            b := NEW B;
            a := b;
        END METHOD
END CLASS
• $a \text{ isA } b$, if
  – $a = b$, or else
  – $a = \text{nullType AND } b \text{ isA } objectClass$, or else
  – $a \# \text{objectClass AND } a.\text{baseType} \text{ isA } b$
Bonus Task: Access Protection

- Access protection
  - *PRIVATE*: access only within the class
  - *PROTECTED*: *PRIVATE* + access from derived classes
  - *PUBLIC*: access from everywhere (default)

- Class *Declarations*
  - Storing access rights with identifiers
  - *resolve(...) needs class of access*
  - Overriding must not restrict the access to a method

```
CLASS Example IS
  PRIVATE internal : Integer;
  PUBLIC METHOD readonly: Integer IS
    BEGIN
      RETURN internal;
    END METHOD
END CLASS
```
memberdecl ::= [ PRIVATE | PROTECTED | PUBLIC ]
( vardecl ';' |
  METHOD identifier [ '(' vardecl { ';' vardecl } ')' ] |
  ':' identifier ] IS methodbody )