



The following slides contain advanced material and are optional.

- CAT calls
- Generic conformance

- For more information, see http://dev.eiffel.com/Comparison_of_catcall_solutions
(Some of the information may be outdated...)

CAT: Changed availability or type



```
class PARENT
feature
  f do end
  g (a: ANY) do end
  h: ANY
end
```

```
class CHILD
inherit
  PARENT
  redefine g
  export {NONE} f end
```

Restricting export status

```
feature
  g (a: STRING) do a.to_upper end
  h: STRING
end
```

Covariant redefinition of
argument and result type

Problems with CAT



```
class APPLICATION
```

```
feature
```

```
  make
```

```
    local
```

```
      p: PARENT
```

```
      c: CHILD
```

```
    do
```

```
      create {CHILD} p
```

```
      p.f
```

```
      p.g (1)
```

```
    end
```

```
end
```

Exported to NONE

Wrong argument type will lead to runtime exception



- Changed availability (restricting export status)
 - Not allowed anymore in ECMA Eiffel
- Covariant result type
 - This is not a problem
- Covariant argument type
 - Currently checked at run-time
 - Different solutions proposed, not yet decided on a particular strategy

Generic conformance



```
class APPLICATION
```

```
feature
```

```
  make
```

```
    local
```

```
      any_list: LIST[ANY]
```

```
      string_list: LIST[STRING]
```

```
      integer_list: LIST[INTEGER]
```

```
    do
```

```
      string_list := any_list           x
```

```
      string_list := integer_list       x
```

```
      integer_list := any_list          x
```

```
      integer_list := string_list       x
```

```
      any_list := string_list           ✓
```

```
      any_list := integer_list          ✓
```

```
    end
```

```
end
```

Generic conformance vs. changed type



```
class LIST [G]
feature
  put (a: G) do end
  first: G
end
```

```
interface class LIST [ANY]
feature
  put (a: ANY)
  first: ANY
end
```

```
interface class LIST [STRING]
feature
  put (a: STRING)
  first: STRING
end
```

LIST [STRING] conforms to LIST [ANY], thus the changed type in the argument and result are like a covariant redefinition.

Problems with generic conformance



```
class APPLICATION
```

```
feature
```

```
  make
```

```
    local
```

```
      any_list: LIST[ANY]
```

```
      string_list: LIST[STRING]
```

```
    do
```

```
      create string_list.make
```

```
      any_list := string_list
```

```
      any_list.put(1)
```

```
      string_list.first.to_upper
```

```
    end
```

```
end
```

Wrong element type



- Novariant conformance
 - No conformance between generics of different type
 - Used by C#
- Usage-site variance
 - Specify conformance for each generic derivation
 - Used by Java
- Definition-site variance
 - Specify conformance for each generic class
 - Implemented by CLR