Programozási nyelvek és paradigmák

A MyType probléma és megoldásai

Kozsik Tamás (2020)

Kovariancia (+ a statikus típusrendszer gyengesége)

```
class LINKED
feature
    link: like Current assign set
    set( new_link: like link ) do link := new_link end -- set
end -- class LINKED
```

Kovariancia (+ a statikus típusrendszer gyengesége)

```
class LINKED
feature
    link: like Current assign set
    set( new_link: like link ) do link := new_link end -- set
end -- class LINKED

class DOUBLY_LINKED
inherit
    LINKED
        rename
            link as linkF,
            set as setF
        end
feature
    linkB: like Current assign setB
    setB( new_link: like linkB ) do linkB := new_link end
end -- class DOUBLY_LINKED
```

Hasonló Javában dinamikus típusellenőrzéssel

```
class Linked {
    private Linked link;
    public Linked getLink(){ return link; }
    public void setLink( Linked link ){ this.link = link; }
}

class DoublyLinked extends Linked {
    private DoublyLinked link2;
    public DoublyLinked getLink2(){ return link2; }
    public void setLink2( DoublyLinked link ){ this.link2 = link; }

    @Override public DoublyLinked getLink(){
        return (DoublyLinked)super.getLink();
    }
    @Override public void setLink( Linked link ){
        if( link == null || link instanceof DoublyLinked )
            super.setLink(link);
        else throw new IllegalArgumentException(
                        "DoublyLinked expected, Linked received");
    }
}
```

F-bounded polymorphism

```
class Linked<T extends Linked<T>> {
    private T link;
    public T getLink(){ return link; }
    public void setLink( T link ){ this.link = link; }
}

class DoublyLinked<T extends DoublyLinked<T>> extends Linked<T> {
    private T link2;
    public T getLink2(){ return link2; }
    public void setLink2( T link ){ this.link2 = link; }
}

class LinkedClass extends Linked<LinkedClass> {}
class DoublyLinkedClass extends DoublyLinked<DoublyLinkedClass> {}
```

Virtuális típusok

- absztrakt bázistípusban típus, mint tag deklarálva
- leszármazottban specializálva (még mindig absztrakt)
- konkrét leszármazottban definálva
- teljes általánosságban ugyanúgy insecure statikusan, mint a kovariáns paraméter
- Scala: részleges virtuális típusok (statikusan helyes típusrendszer)

Típussal való paraméterezés helyett virtuális típus

```
class Pair[L,R]( left: L, right: R )
val v = new Pair[Int,String](1,"hi")
v.right
```

```
abstract class Pair {
    type L
    type R
    val left: L
    val right: R
}
val v = new Pair{ type L=Int; type R=String;
                  val left=1; val right="hi" }
v.right
```

Bináris műveletek virtuális típussal

```
abstract class Linked {
     type MyType <: Linked
     var next: MyType = null.asInstanceOf[MyType]
}

class List extends Linked { type MyType = List }
```

Bináris műveletek virtuális típussal

```
abstract class Linked {
     type MyType <: Linked
     var next: MyType = null.asInstanceOf[MyType]
}

class List extends Linked { type MyType = List }

abstract class DoublyLinked extends Linked {
     var prev: MyType = null.asInstanceOf[MyType]
}

class List2 extends DoublyLinked { type MyType = List2 }
```

Bináris műveletek

- `like Current` (match-bounded polymorphism helyessé teszi)
- virtuális típus
- F-bounded polymorphism (részben)
- multimethod

Match-bounded polymorphism

- Egy OOP-nyelvbe nem feltétlen kell altípusosság
- Másfajta polimorfizmus is biztosíthat újrafelhasználhatóságot

Altípusossággal: DoublyLinked <: Linked (pseudo-kód)

```
class Linked
    getNext: Linked ...
    setNext( new_next: Linked ) ...
end

class DoublyLinked inherit Linked
    redeclare next: DoublyLinked
    override setNext( new_next: Linked ) ...
    prev: DoublyLinked
    setPrev( new_prev: DoublyLinked ) ...
end

length( list: Linked ): INTEGER
    do
        from Result := 0 until list = Void loop
                list := list.next
                Result := Result + 1
        end
    end
```

Illeszkedéssel: DoublyLinked <# Linked (pseudo-kód)

```
class Linked
    getNext: like Current ...
    setNext( new_next: like Current ) ...
end

class DoublyLinked inherit Linked
    prev: like Current
    setPrev( new_prev: like Current ) ...
end

length[ L <# Linked ]( list: L ): INTEGER
    do
        from Result := 0 until list = Void loop
                list := list.next
                Result := Result + 1
        end
    end
```