

Polymorphie & Aufzählungstypen



Polymorphe Funktionen

- **bisher:** meist auf `Integer` operiert oder auf `Any`
- **jetzt:** Typsystem verallgemeinern
- **Bsp.: Funktion** `map`

```
def map(xs:List[Int], f:Int => Int) : List[Int] =  
  xs match {  
    case Nil      => Nil  
    case y :: ys => f(y) :: map(ys, f)  
  }
```

- **besser:** für beliebiges `f:A => B`
- **hierzu:** Polymorphes `map`

```
def map[A,B] (xs:List[A], f: A => B) : List[B] = s.o.
```

- **Aufruf:** Instanziierung von A und B:

```
map[Int,String] (1::2::Nil, _=> "Hi")  
-> "Hi"::"Hi"::Nil
```

Polymorphe Case Klassen / Objects

Definition

- **bisher:** Polymorphes `map` auf interner Listenstruktur
- **jetzt:** Polymorphe Erweiterung von selbstdefinierten Strukturen
- **Bsp.:** selbstdefinierte Listenstruktur aus Kap. 5

```
abstract class MyList
case object Nil extends MyList
case class List( head:Int,
                tail:MyList) extends MyList
```

- **bisher:** Eintragstyp `Int`
- **jetzt:** Polymorpher Eintrag **A**

```
abstract class MyList[+A]
case object Nil extends MyList[Nothing]
case class List[A]
(head:A,
 tail:MyList[A]) extends MyList[A]
```

Polymorphe Case Klassen / Objects

Funktionen und Aufrufe

- **Beispiel:** `append` auf nicht polymorpher `MyList`

```
def append(xs:MyList,ys:MyList) : MyList =  
  xs match {  
    case Nil => ys  
    case List(y,zs) => List(y,append(zs,ys))    }
```

- **jetzt:** `append` auf polymorpher `MyList[+A]`

```
def append[A](xs:MyList[A],ys:MyList[A]) : MyList[A] =  
  xs match {  
    case Nil => ys  
    case List(y,zs) => List(y,append(zs,ys))    }
```

- **Aufruf:** `append` auf `MyList[Int]`

```
append[Int](List(1,Nil),List(2,Nil))
```

Case Klassen / Objects

Aufzählungstypen

- **oft:** endliche Typen
- **Bsp.:** Alphabete, Zustandsmenge, Farben einer Ampel
- als Case Klassen Objects definieren:

```
abstract class Ampel
```

```
case object Rot extends Ampel
```

```
case object RotGelb extends Ampel
```

```
case object Grün extends Ampel
```

```
case object Gelb extends Ampel
```

- Übungsaufgabe Funktion `schalte` für `Ampel` mit Pattern Matching