TOUR OF SCALA

POLYMORPHIC METHODS

Methods in Scala can be parameterized by type as well as value. The syntax is similar to that of generic classes. Type parameters are enclosed in square brackets, while value parameters are enclosed in parentheses.

Here is an example:

```
def listOfDuplicates[A](x: A, length: Int): List[A] = {
  if (length < 1)
    Nil
  else
    x :: listOfDuplicates(x, length - 1)
}
println(listOfDuplicates[Int](3, 4))  // List(3, 3, 3, 3)
println(listOfDuplicates("La", 8))  // List(La, La, La, La, La, La, La, La)
```

The method `listOfDuplicates` takes a type parameter `A` and value parameters `x` and `length`. Value `x` is of type `A`. If `length < 1` we return an empty list. Otherwise we prepend `x` to the list of duplicates returned by the recursive call. (Note that `::` means prepend an element on the left to a list on the right.)

In first example call, we explicitly provide the type parameter by writing `[Int]`. Therefore the first argument must be an `Int` and the return type will be `List[Int]`.

The second example call shows that you don't always need to explicitly provide the type parameter. The compiler can often infer it based on context or on the types of the value arguments. In this example, `"La"` is a `String` so the compiler knows `A` must be `String`.

← previous next →

Contributors to this page:

kimhongji ckipp01 mlachkar ashawley SethTisue OlivierBlanvillain jkatsnelson heathermiller