Conditions

Effective Programming in Scala

Conditions

Let us say that we want to show the price of the paint to buy.

As long as that price is lower than 100 Euros we show it, otherwise we show a message “This is too expensive”.

`if` Expressions

```
def showPrice(paintingArea: Double, paintPrice: Double): String =
  val price = paintingArea * paintPrice
  if price > 100 then
    "This is too expensive"
  else
    price.toString
```

An `if` expression takes a condition followed by the keyword `then`, and has two continuation branches separated by `else`.

It is worth noting that the `if`/`else` construct is an expression: it evaluates to a value.

Conditions

A condition must be an expression of type `Boolean`, otherwise it's an error:

```
if 1 then "foo" else "bar"
   ^
       error: type mismatch;
        found   : Int(1)
        required: Boolean
```

else if

You can define more than two branches by using `else if`:

```
if price > 100 then
  "This is too expensive"
else if price < 10 then
  "This is very cheap"
else
  price.toString
```

`if` Expressions (Scala 2 Compatibility)

In Scala 2, the syntax was a bit different. The condition was written between parenthesis, and there was no `then` keyword:

```
if (price > 100) {
  "This is too expensive"
} else {
  price.toString
}
```

Summary

Alternative branches of computations can be implemented by using `if` expressions.