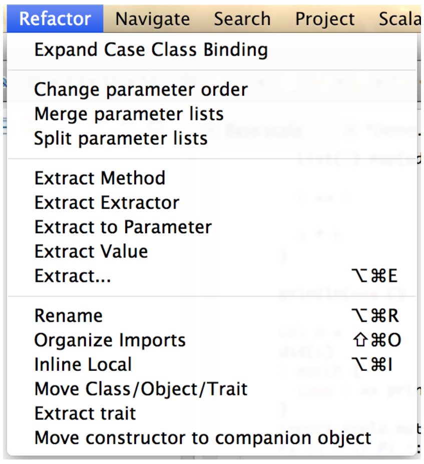
A. User Guide for the Extraction Refactorings

The extraction refactorings offer automated extraction of almost arbitrary code fragments into new methods, values or even extractor objects. All you have to remember is that when you want to extract something you have to select it, hit `Cmd` + `Alt` + `E` and the assistant guides you through the available extractions. Naturally, each of the refactorings is also invokable over the refactoring menu.

Figure A.1.: The extraction tools in the Refactoring menu of Scala IDE



An extraction creates a new abstraction (value, method etc.) with the identical behavior as the selected code and replaces the selection by an according call to this abstraction. Extractions are typically useful to make the code more readable and to allow the reuse of a code fragment in other places.

"Extract..." proposes based on the selected code one or more of the following extractions:

Extract Value

Creates a new `val` definition from the selected code and replaces the selection by a reference to the new value

Extract Method

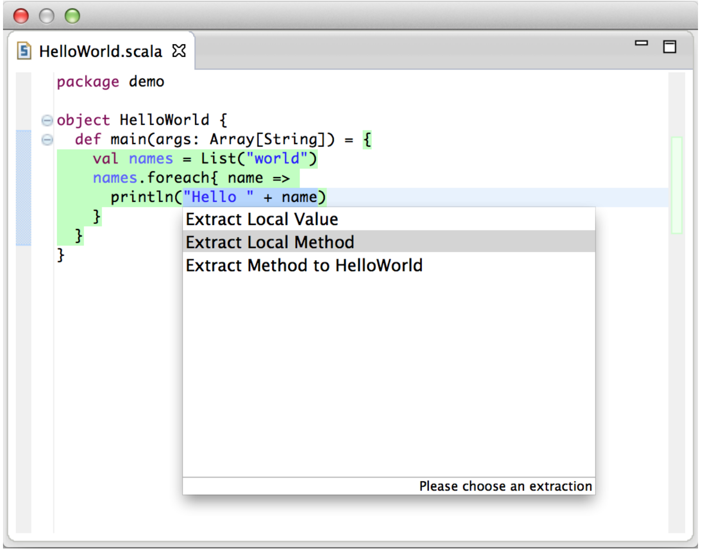
Creates a new method (`def`) from the selected code and replaces the selection by the according call

Extract Parameter
Adds a new parameter to an enclosing method whose default value is the selected code and replaces the selection by a reference to the new parameter

Extract Extractor
Creates a new extractor object based on a selected pattern in a case statement and replaces the pattern by an according call to the new extractor

After invoking "Extract…" a drop-down opens that contains a list of available extractions. When you select one of the extractions with the `Up` and `Down` keys the scope in which the extraction creates the new abstraction is highlighted green. Additionally, if you select only a part of an expression that can be processed by "Extract…" the selection is automatically expanded such that you see what exactly will be extracted.

Figure A.2.: The extraction selection assistant. The code marked green represents the target scope of the selected extraction.



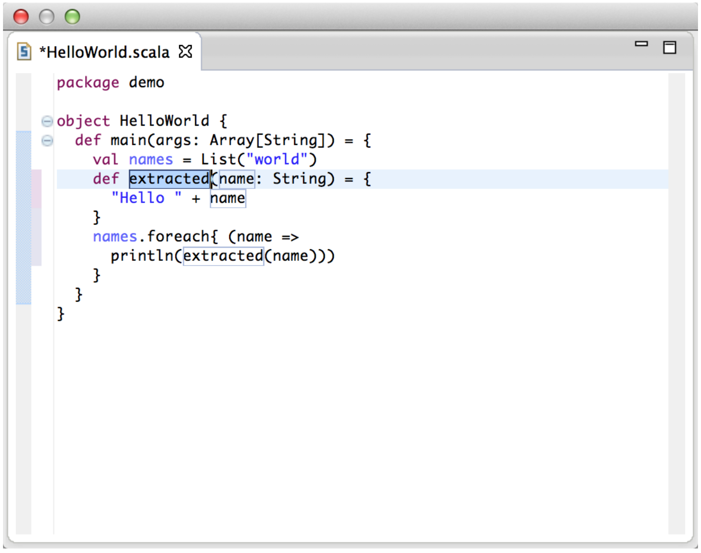
```
package demo

object HelloWorld {
  def main(args: Array[String]) = {
    val names = List("world")
    names.foreach{ name =>
      println("Hello " + name)
    }
  }
}
```

When you found the extraction that you want to become applied you can choose it by hitting `Enter` and the extraction tool performs the according transformations to the source code. Finally, Eclipse enters the linked mode that allows you to rename the new abstraction right in the editor window. If you extracted a method with parameters you can also jump to the parameters by hitting `Tab` and rename them as well.

The extraction refactorings offer many useful features:

- Only one shortcut for all kinds of extraction refactorings
- Lets you precisely choose the scope in which you want to create the new abstraction

Figure A.3.: Inline renaming of an extracted method and its parameter



```
package demo

object HelloWorld {
  def main(args: Array[String]) = {
    val names = List("world")
    def extracted(name: String) = {
      "Hello " + name
    }
    names.foreach{ (name =>
      println(extracted(name)))
    }
  }
}
```

- "Extract…" helps you to choose the right extraction by analyzing the selected code
  - If the code triggers no side effects…
    - it proposes a value extraction for every scope in which all variables referenced in the selection are accessible
    - and a method extraction for every scope in which at least one of these variables is not accessible anymore
    - and a parameter extraction for every method that encloses the selection
  - If the code triggers side effects…
    - it proposes only method extractions
  - If the code is a pattern in a case statement…
    - it proposes extractor extractions
- All tools support idiomatic features of the Scala language
  - Extraction of higher order functions
  - The resulting code uses type inference when possible
  - Extraction into local closure methods
  - Multiple return values with tuples