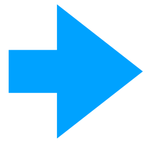
Concurrency with actors, attempt 2

- So … everyone has akka uder control
- The goal of the exercise today is to write efficient translator from “any” language to any other language.
- As input we have dictionaries (with translations to english) of 1000 most frequently used words in few European languages.
- The suggested solution:
  - Have one super-star actor getting words (as messages) form the user.
  - The super-star actor communicates this word to “worker” actors.
  - There would be one worker actor per language. Once the word is found by worker it is reported back to super-star together with the translation to english.
  - The superstar then asks every worker to report back translation from english. Once this is available the super-star just prints:
    the word looked for, the english translation and the translations to other languages



There are files (1000 most frequent) at the bottom of the webpage and below is the code to read them in a list of pairs.

```
import scala.io.Source

case class Word( val native:String, val eng: String )

object DictFile {
    def read(cc:String) : List[Word] = {
        val fileName = cc+"1000.txt"
        val fileBuffer = Source.fromFile(fileName)
        val flatContent = fileBuffer.getLines().toList
        val words =  flatContent.grouped(3).map( d => { Word(d(1), d(2))}).toList
        fileBuffer.close()
        return words
    }
}
```

if you want interactive input …

```
val user_input = scala.io.StdIn.readLine
```