How to provide Multi-language support

Managing the different text translations of a game can easily be done by supplying a simple text file for each language in the resource folder. For instance, resource strings for English go in **en.txt**:

```
Add|Choose|Remove|View|etc...
```

resource strings for Spanish **es.txt**:

```
Añadir| Elegir| Eliminar| Ver | etc...
```

In this case we choose "|" as a delimiter. It's important to use a character that doesn't commonly appear in text.

Then you can parse the text into an array using the following code:

```
import java.io.*;
import java.util.Vector;

public class Translator
{
 private static Vector text;
 public static final short add = 0;
 public static final short choose = 1;
 public static final short remove = 2;
 public static final short view = 3;
 private static final int maxBuffer = 0x800;
 public Translator()
 {}

 public static void readText()
 {
  text = new Vector();
  try
  {
   // get the text resource as a stream
   InputStream is = Runtime.getRuntime().getClass()
    .getResourceAsStream("/en.txt");
   InputStreamReader isr = new InputStreamReader(is);
   char[] buffer = new char[maxBuffer];
   char[] currItem = new char[400];
   int n = 0;

   int read = 0;
   int currPos = 0;
   char c;
   String strText;
   do
   {
    read = isr.read(buffer, 0, maxBuffer);
    if (read == -1)
     break;
    n = 0;
    do
    {
```


(C) Copyright Nokia 2014. All rights reserved.

```
      // now we have a part of the file stored in buffer, so then
      // we load it into curr string
      while(n < read && buffer[n] != '|')
      {
       currItem[currPos] = buffer[n];
       n++;
       currPos++;
       if (currPos >= currItem.length-1)
       {
        DebugStuff.print("buffer overflow at "+text.size()+" item");
        return;
       }
      } // end while
      if (n == read) break;
      // we have recieved a token before finishing the buffer
      if (n < read)
      {
       strText = new String(currItem, 0, currPos);
       text.addElement(strText);
       currPos = 0;
       do
       {
        n++;
        if (n == read) break;
        c = buffer[n];
       } while (c == '\n' || c == '\r' || c == ' ');
       // Note: there might be a problem if white space after token occurs exactly after
reading a block
      }
     } while(true);
    } while (true);
    // note: any trailing characters without a delimiter will be ignored
    isr.close();
   }
   catch (IOException e)
   {
    DebugStuff.print(e.toString());
    return;
   }
  }

  static String get(int id)
  {
   if (id >= text.size()) return "missing string";
   return (String)text.elementAt(id);
  }
 }
```

(In fact this is quite handy code to use for parsing any text delimited file)

- Note: this code does not support Unicode.

Finally you access the localised strings as follows:

```
String str = Translator.get(Translator.add)
```


(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.