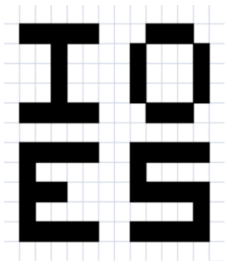
TAP Labo 7/12/2016

Tipografías

La finalidad de la presente práctica es crear un conjunto de clases que permitan la generación de patrones gráficos que representen los caracteres (letras, dígitos y símbolos) de una lengua. Para ello, partiremos de la interfaz `Grapheme`:

```
public interface Grapheme {
      public void setSize(int size);
      public int getSize();
      public String[] createPattern();
}
```



Un grafema es la abstracción de una letra o tipo. Puede representar a cualquier letra de una lengua dada (en nuestro caso el español) y tiene un tamaño asociado. Este tamaño hace referencia a la matriz cuadrada de puntos que se utilizará para su representación visual. En la figura se muestra un ejemplo de grafemas de tamaño 5 para las letras `I`, `O`, `E` y `S` respectivamente.

El método `createPattern()` devolverá un array de Strings cuyo tamaño corresponderá al tamaño del grafema, y el contenido de cada String será una de las líneas del grafema (utilizaremos caracteres para representar cuadrados *negros* y *blancos*). Por ejemplo, en el caso de la I del la imagen, la primera y última de las cadenas de caracteres podrían contener "`XXXXX`" mientras que las cadenas intermedias podrían contener " `X` " (en este caso concreto, un cuadrado negro es representado por una '`X`' y un cuadrado blanco es representado por un espacio, ' ') . Si mostrásemos cada una de las cadenas de caracteres por pantalla, obtendríamos:

```
XXXXX
  X
  X
  X
XXXXX
```

Además de la interfaz `Grapheme`, se define la siguiente interfaz:

```
public interface Typography {
    public Grapheme valueOf(char c);
}
```

Toda clase que implemente la interfaz `Typography` deberá ser capaz de, dado un carácter `c`, generar su correspondiente grafema. Para poder mostrar los grafemas por pantalla, se creará la siguiente clase Grafemes que está compuesta únicamente por métodos estáticos:

```
public class Graphemes {

  public static String toString(Typography t, char c){
       Grapheme g = t.valueOf(c);
       StringBuilder sb = new StringBuilder();
       for(String s : g.createPattern()) sb.append(s).append('\n');
       return sb.toString();
  }

  public static String toString(Typography t, int size, char c){
       Grapheme g = t.valueOf(c);
       g.setSize(size);
       StringBuilder sb = new StringBuilder();
       for(String s : g.createPattern()) sb.append(s).append('\n');
       return sb.toString();
  }

  public static String toString(Typography t, String s){
       throw new UnsupportedOperationException("Not supported yet!");
  }

  public static String toString(Typography t, int size, String s){
       throw new UnsupportedOperationException("Not supported yet!");
  }

}
```

Estos métodos ofrecen la posibilidad de generar una única cadena de caracteres que represente un carácter o una cadena de caracteres. Estos métodos podrían ser utilizados por ejemplo de la siguiente manera:

```
Typography t ;
// Inicializar la tipografía
System.out.println(Graphemes.toString(t,'H’));
System.out.println(Graphemes.toString(t,4,'E’));
System.out.println(Graphemes.toString(t,6,'L’));
System.out.println(Graphemes.toString(t,8,'L’));
System.out.println(Graphemes.toString(t,10,'O’));
System.out.println(Graphemes.toString(t,”HELLO”));
System.out.println(Graphemes.toString(t,10,”HELLO”));
```

La finalidad del laboratorio es doble. Por un lado, completar la clase Graphemes (dos de sus métodos no están implementados). Por otro lado, crear una clase que implemente la interfaz `Typography` para poder ejecutar el código anterior.

Extra: podemos usar lo planteado en la práctica para hacer una clase con este prototipo:

```
public class BannerWriter extends FilterWriter{
    BannerWriter(Typography t, Writer out){ /* código */ }
    /* código */
}
```