**Notas de la versión 1.0 de nuestra aplicacion de migración**

Nuestra aplicación está compuesta de las siguientes carpetas, clases y demás documentos organizada de la siguiente manera:

* Carpeta **src**: Contiene todo el código fuente de nuestro proyecto (Ver siguiente apartado).
* Carpeta **test**: Contiene varios tests usados a lo largo del desarrollo.
* Carpeta **doc**: Contiene todo el javadoc generado automáticamente para el proyecto.
* Carpeta **anotaciones:** Contiene mi documentación personal sobre el proyecto, un fichero de análisis y este documento de notas de la versión.
* Carpeta **ficheros**:
  + Carpeta **input**: Contiene el fichero que vamos a procesar.
    - personas.txt
  + Carpeta **output**: Inicialmente no contiene ningún fichero pero después de ejecutar nuestra aplicación contendrá los ficheros que nuestra aplicación va a generar:
    - **estadisticas.txt**: Contiene estadísticas relativas a la ejecución de nuestra aplicación.
    - **personas-correctas.txt**: Contiene las personas correctas según nuestros criterios (Ver análisis).
    - **personas-error.txt**: Contiene las personas incorrectas según nuestros criterios (ver análisis).
    - **personas-repetidas.txt**: Contiene las personas que tienen el DNI repetido.

**Descripción de las clases y ficheros incluidos en nuestro codigo fuente ( carpeta src )**

1. **Constantes.java**

Contiene ciertas constantes que van a ser utilizadas en distintas clases a lo largo del proyecto, en este fichero se han intentado incluir las constantes que van a ser usadas en mas de una de las otras clases del proyecto, como por ejemplo los nombres de los ficheros sobre los que vamos a trabajar

1. **Migracion.java**

Es una clase con una función main que sirve para arrancar nuestro proceso/aplicacion. Esa es la unica funcion de esta clase.

1. **Persona.java**

Gracias a la programación orientada a objetos podemos representar objetos del mundo real dentro de nuestros proyectos y aplicaciones. En esta aplicación vamos a trabajar con un fichero lleno de entradas de personas por lo que parece lógico que nos creemos una clase Persona para representar esas entradas. La clase persona dispone de dos constructores, uno que utiliza setters que comprueban los valores usados en la construcción del objeto para poder validar cada persona de acuerdo con los criterios establecidos por el cliente (ver análisis) y de esa forma lanzar excepciones que nos permitan identificar la causa de ese fallo y otro constructor que nos permite crear el objeto persona sin validarlo que usaremos para comprobar que personas tienen el dni repetido.

1. **PersonaException.java**

Es nuestra excepción personalizada, la que lanzaremos cuando alguna persona de las que estemos procesando no cumpla con los criterios que nuestro cliente nos ha detallado (ver análisis). La excepción dispone de varias constantes que indican el código y el mensaje de error según la excepción que lancemos en cada momento.

1. **Fichero.java**

Esta clase contiene todas las funciones que nos permitirán manipular y trabajar con los ficheros (leer y escribir) y en definitiva conseguir unos ficheros de salida acordes a nuestras necesidades y requerimientos.

1. **PersonaComparator.java**

Esta es una clase sencilla que nos permite comparar objetos de tipo persona usando su DNI. Se usa para crear el TreeSet que usaremos para poder encontrar a las personas con DNI duplicado.

1. **Utilidades.java**

Esta es una clase que contiene funciones de validación de los atributos de las personas de acuerdo a los criterios establecidos por nuestro cliente. Contiene funciones tanto para validar DNI (usando una [librería de java creada por la propia agencia tributaria](http://www.agenciatributaria.es/AEAT.desarrolladores/Desarrolladores/Descarga_de_modulos/Descarga_de_modulos.html) para este fin) y email como para validar la codificación de las cadenas en UTF-8.

**Otros ficheros**

* **Build.xml**

Contiene tareas ant relacionadas con la generación y borrado automático de javadoc y con el borrado de los ficheros generados por nuestra aplicación con fines de testeo.

* **Valnif.jar**

Librería obtenida en la [página de desarrolladores de la agencia tributaria](http://www.agenciatributaria.es/AEAT.desarrolladores/Desarrolladores/Descarga_de_modulos/Descarga_de_modulos.html). Contiene las clases y funciones para la validación del DNI.