La prueba consistirá en 3 problemas:
● Cada problema deberá ser resuelto en una clase distinta.
● Se puede usar composer para instalar cualquier componente o librería que se cree necesario.
● Se pueden crear clases auxiliares para estructurar mejor el código
Para este proyecto deberás crear un repositorio git (bitbucket, github, etc). En caso de que quieras hacer el repositorio privado, compartelo a dgarcia@qashops.com
El objetivo de la prueba no es tanto el resultado en sí, sino que decisiones se toma y cómo se estructura el problema.

Prueba 1: Refactoring

Tenemos la clase Product (Anexo 1) con un método llamado stock(). El código de dicho método realiza comprobaciones duplicadas y el código carece de bastante legibilidad. Refactoriza el método (creando métodos auxiliares, reordenando el código, renombrando variables, etc) para que se entienda mejor qué hace y sea más legible. Lo único que no se puede cambiar es la firma del método.

Prueba 2: Aplanamiento XML

Crear un metodo, que dado un path a un fichero XML, lo aplane y guarde en un csv..

Ejemplo de XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <products>
    <product>
      <name_header>name</name_header>
      <description_header>description</description_header>
      <image_1_header>image_1</image_1_header>
      <colour_header>colour</colour_header>
      <size_header>size</size_header>
      <sleeve_length_header>sleeve_length</sleeve_length_header>
    </product>
    <product>
      <name_header>name</name_header>
      <description_header>description</description_header>
      <sku_header>sku</sku_header>
      <image_1_header>image_1</image_1_header>
      <colour_header>colour</colour_header>
```

```
      <heel_height_header>heel_height</heel_height_header>
      <laces_header>laces</laces_header>
    </product>
    <product>
      <name_header>name</name_header>
      <description_header>description</description_header>
      <sku_header>sku</sku_header>
      <image_1_header>image_1</image_1_header>
      <colour_header>colour</colour_header>
      <size_header>size</size_header>
      <sleeve_length_header>sleeve_length</sleeve_length_header>
    </product>
    <product>
      <name_header>name</name_header>
      <description_header>description</description_header>
      <sku_header>sku</sku_header>
      <image_1_header>image_1</image_1_header>
      <colour_header>colour</colour_header>
      <size_header>size</size_header>
      <sleeve_length_header>sleeve_length</sleeve_length_header>
    </product>
  </products>
</xml>
```

Output Esperado:
name_header;description_header;image_1_header;colour_header;size_header;sleeve_length_header;sku_header;heel_height_header;laces_header
name;description;image_1;colour;size;sleeve_length;;;
name;description;image_1;colour;;;sku;heel_height;laces
name;description;image_1;colour;size;sleeve_length;sku;;
name;description;image_1;colour;size;sleeve_length;sku;;

Prueba 3: Merge de dos ficheros CSV

Crear un método, que dado 2 paths de dos ficheros csv, los unifique en un solo fichero. Las cabeceras (asumimos que la cabecera está siempre en la primera fila) que estén en común en ambos ficheros deben estar unificadas y solo estar presentes una vez en el fichero devuelto.

Ejemplo Fichero 1:
Header_1,Header_2,Header_3,Header_4
1,2,3,4
4,5,6,7

Ejemplo Fichero 2:
Header_5,Header_2,Header_6,Header_7
8,9,10,11
12,13,14,15

Output Esperado:
Header_1,Header_2,Header_3,Header_4,Header_5,Header_6,Header_7
1,2,3,4,,,
4,5,6,7,,,
,9,,,8,10,11
,13,,,12,14,15