Programación Orientada a Objetos

Práctica 4: Herencia y polimorfismo
Implementación del caso de uso 4
«Gestión del catálogo de artículos» del S. G. L.

Curso 2016–17

Índice

1. Introducción

En esta práctica se va a llevar a cabo la implementación y prueba del caso de uso 4 «Gestión del catálogo de artículos» del Sistema de Gestión de Librería (SGL) descrito de manera general en la sección de prácticas del campus virtual de la asignatura.

Se partirá de la solución del caso de uso 3, desarrollada en la práctica anterior, junto con los requisitos correspondientes al caso de uso 4, y a partir de su diagrama de clases se realizará su implementación en el lenguaje de programación C++. Después se deberá realizar una serie de pruebas para asegurarse de que la aplicación, para el caso de uso 4, funciona correctamente y satisface los requisitos especificados.

2. Objetivos

Implementar en C++ el modelo de clases para el caso de uso 4 del SGL cumpliendo los principios de la programación orientada a objetos y todas las pruebas necesarias para que estemos seguros en un alto grado de que la implementación tiene el comportamiento esperado.

3. Descripción de los requisitos del caso de uso 4

Continuaremos el desarrollo de nuestro SGL completando la gestión del catálogo de artículos, partiendo de la gestión de pedidos. Para esto es necesario incorporar al sistema los distintos tipos de artículos que van a formar parte del catálogo.

Se podrán añadir al catálogo tres tipos de artículos: libros, cederrones y libros digitales (*ebooks*). Todos ellos tienen un código de referencia interno (una cadena), un precio, un título y una fecha de publicación, pero solo de los libros y cederrones hay que almacenar el *stock* (número de artículos disponibles), ya que los libros digitales se distribuyen mediante descarga por Internet. De todos los artículos también se guardan los autores, que pueden ser uno o varios. Los libros digitales tienen además una fecha de expiración en la que dejan de venderse, ya que se consideran obsoletos. De los libros se almacena el número de páginas, y de los cederrones el tamaño en MB.

4. Diagrama de clases del caso de uso 4

En la figura 1 aparece el diagrama de clases asociado con el caso de uso 4. Se trata de una especialización de la clase *Articulo*.

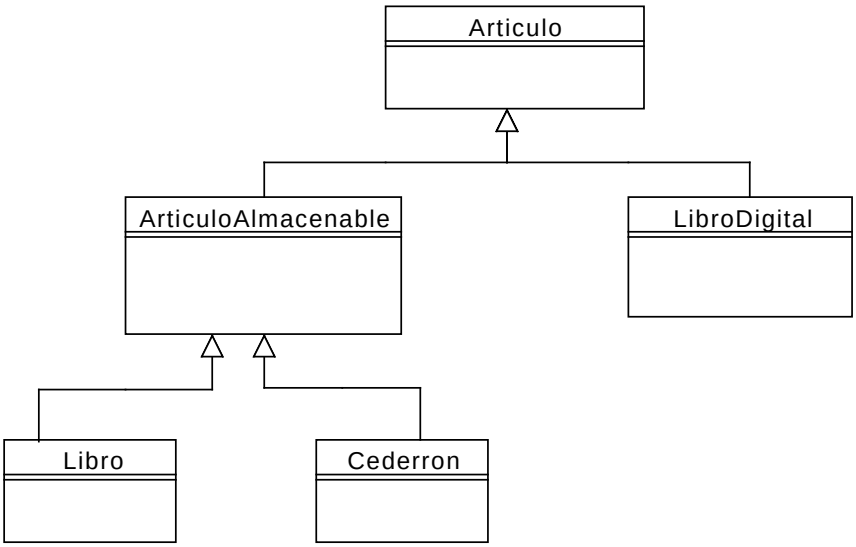

Figura 1: Diagrama de clases del caso de uso 4

5. Implementación de las clases

5.1. La clase Articulo

La clase *Articulo* se verá ahora afectada por la introducción de las clases derivadas. Algunos de sus métodos y atributos deberán ser eliminados de la clase y pasados a las derivadas.

Defina esta clase como abstracta, de forma que no se podrán crear objetos de ella. Ahora el primer parámetro del constructor será *Autores* (V. §5.6).

Reescriba el operador de inserción (<<) para esta clase de forma que se elija durante la ejecución la definición adecuada dentro de la jerarquía de clases. Defina un método llamado *void Articulo::impresion_especifica(ostream&)* para imprimir los atributos específicos de cada tipo de artículo. El formato de salida para cada tipo de artículo será el que se muestra en los ejemplos a continuación. Use un tabulador para sangrar la segunda línea:

- Libros:

```
[110] "Cómo montar en bici sin sillín", de Poyáquez, Picuito. 1.998. 35,95 €
        650 págs., 100 unidades.
```

- Cederrones:

```
[220] "Jarabe de Falo", de Poyáquez, Picuito, Leches. 2.007. 12,90 €
        547 MB, 30 unidades.
```

- Libros digitales:

```
[035] "Horarios", de Poyáquez. 2.009. 9,00 €
       A la venta hasta el miércoles 19 de junio de 2019.
```

Observe que esta modificación afectará al formato de salida de la función *mostrar_carro*, que imprime el contenido del carrito de un usuario en un flujo de salida. Reescriba esta función, de forma que no dependa del operador de inserción de *Articulo*, para conservar el formato de salida original.

5.2. La clase ArticuloAlmacenable

Esta clase será también abstracta, de forma que tampoco se podránn crear objetos de ella.

Esta clase representa a todos aquellos artículos que se puedan almacenar. Esto es, aquellos para los que hay que guardar un *stock* (existencias, número de artículos disponibles). Las existencias se podrán consultar y modificar a través de los métodos *stock*.

5.3. La clase Libro

Esta es una clase almacenable, o sea, que posee un *stock*, y además hay que guardar el número de páginas, que se podrá consultar a través del observador *n_pag* y no se podrá cambiar tras crear el objeto.

5.4. La clase Cederron

Esta clase también es almacenable (tiene *stock*), y además hay que guardar el tamaño en MB (observador *tam*). El tamaño no se puede cambiar una vez se ha creado el objeto.

5.5. La clase LibroDigital

Para esta clase no hay que almacenar las existencias, pero hay que tener en cuenta la fecha de expiración (observador *f_expir*) en la cual los *ebooks* dejan de venderse. La fecha de expiración no puede cambiar una vez se ha creado el objeto.

Por tanto, no se podrán comprar libros digitales cuya fecha de expiración sea anterior a la actual. Este requisito hace necesario modificar el constructor de la clase *Pedido*: si un usuario ha introducido en su carrito de compra un libro digital expirado, al contrario que ocurre con los artículos almacenables sin existencias, el pedido no se anulará, simplemente el *ebook* no se añadirá al pedido. Si el pedido queda vacío por este motivo, debe lanzar la excepción *Pedido::Vacio* correspondiente.

5.6. La clase Autor

Deberá crear la clase *Autor* para guardar los datos de los autores (nombre, apellidos y dirección). El constructor no lanzará ninguna excepción. Definirá públicamente en *Articulo* un sinónimo *Autores* con el tipo necesario para representar la relación.

Esta clase se ha omitido en el diagrama de la figura 1, ya que el estudiante deberá decidir cómo se relaciona esta con las otras clases. Nótese que en lo único en que estamos interesados es en saber cuáles son los autores de un artículo dado. Todo artículo tendrá al menos uno; de no ser así, se lanzará la excepción *Autores_vacios*. Dicha información (los autores) será proporcionada cuando se cree el artículo, sea del tipo que sea, y no se podrá cambiar posteriormente.

6. El Makefile

Se proporciona el fichero *Makefile*. Examínelo para ver sus reglas con sus objetivos.

La estructura de directorios correcta es:

```
Povedilla_Putierrez_Pancracio/P1/cadena.[ch]pp
                              |  |fecha.[ch]pp
                              |  |Makefile
                              |luhn.cpp
                              |Tests-auto/fct.h
                              |          |test-auto.[ch]pp
                              |          |test-caso0-{cadena,fecha}-auto.cpp
                              |          |test-caso1-2-auto.cpp
                              |          |test-caso[34]-auto.cpp
                              |dsl-comprobaciones/checkCode.{h,cpp}
                              |                  |execute.{h,cpp}
                              |                  |matchers.{h,cpp}
                              |                  |info.h
                              /P4/test-caso4-consola.cpp
                                 |usuario.[ch]pp
                                 |usuario-pedido.hpp
                                 |tarjeta.[ch]pp
                                 |articulo.[ch]pp
                                 |pedido.[ch]pp
                                 |pedido-articulo.[ch]pp
                                 |Makefile
                                 |Make_check.mk
                                 |catalogo_check.cpp
```

Para la entrega, debe usar el objetivo *dist*, que generará un archivo llamado
`Povedilla_Putierrez_Pancracio.tar.gz`
en el directorio del mismo nombre, que, como se ve, **debe** ser el del alumno. Este objetivo solo archiva los ficheros fuente creados por el alumno. Los demás no hacen falta ya que los originales están en poder de los profesores.

No está `usuario-pedido.cpp` porque se pueden hacer todas las funciones correspondientes *inline*, poniéndose entonces en las cabeceras, pero se es libre de implementarlas en ese *cpp* que falta, debiendo en ese caso modificar el *Makefile* para añadirlo a la variable `COMM_SRCS` como se explica en los comentarios del *Makefile*. La clase *Autor* se escribirá en los ficheros `articulo.[ch]pp`.