

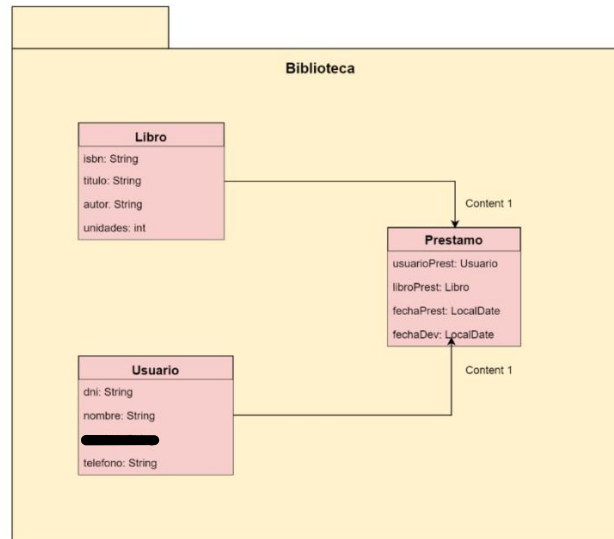
PROGRAMACIÓN 1º DAW – TRABAJO CALIFICADO CUATRIMESTRE 1 (2023-2024)

Propuesta: Tutoría colectiva 22/12/2023

Entrega (Fecha límite): Tutoría colectiva 25/01/2024

ENUNCIADO:

Desarrollar un programa Java para gestionar préstamos en una biblioteca. Las clases VO implicadas son:



Siguiendo como modelo el ejemplo de **Cuentas Bancarias** trabajado en clase, el programa dará lugar a una 4ª clase que llamaremos **Biblioteca**. En esa clase **Biblioteca** se organizarán los **usuarios**, **libros** y **préstamos** de la biblioteca en sendas colecciones de tipo **ArrayList**. En la clase **Biblioteca** se arrancará la aplicación (método **main**) y se mostrará el siguiente menú principal:

```
MENU DE OPCIONES

1 - GESTION USUARIOS/AS
2 - GESTION LIBROS
3 - GESTIÓN PRÉSTAMOS/DEVOLUCIONES
9 - SALIR
```

Las opciones 1-2-3 “GESTIÓN ... ” nos llevarán cada una de ellas al correspondiente submenú. La opción 9 provocará la salida del programa. A continuación, vemos las opciones de cada submenú de **GESTIÓN**.


```
GESTIÓN DE USUARIOS/AS

1 - ALTA NUEVO USUARIO/A
2 - BAJA USUARIO/A
3 - LISTADO USUARIOS/AS
9 - VOLVER
```

GESTIÓN DE LIBROS

- 1 - ALTA NUEVO LIBRO
- 2 - BAJA LIBRO
- 3 - MODIFICACION DATOS LIBRO
- 4 - LISTADO DE LIBROS DISPONIBLES
- 9 - VOLVER

GESTIÓN DE PRÉSTAMOS

- 1 - PRÉSTAMOS
- 2 - DEVOLUCIONES
- 3 - PRÓRROGAS
- 4 - LISTADO PRESTAMOS (TODOS)
- 
- 9 - VOLVER

GESTIÓN DE USUARIOS.

1. **ALTA** de un nuevo usuario/a aportando todos sus datos por teclado (Dni, Nombre, Telefono).
2. **BAJA** (eliminación) de un usuario/a aportando su Dni.
3. **LISTADO** de todos los usuarios/as en un momento determinado mostrando todos sus datos.

GESTIÓN DE LIBROS

1. **ALTA** de un nuevo libro aportando todos sus datos por teclado (Isbn, Título, Autor, Unidades).
2. **BAJA** (eliminación) de un libro aportando su **ISBN**.
3. **MODIFICACIÓN** de un libro. Consiste en modificar el **número de Unidades** disponibles de un libro en concreto en la biblioteca. **Se solicita el ISBN por teclado y el número de unidades (+/- unidades).**
 - + para añadir nuevos ejemplares de ese libro.
 - para restar unidades por extravío o deterioro.
4. **LISTADO** de todos los libros disponibles en un momento determinado mostrando todos sus datos.

GESTIÓN DE PRÉSTAMOS

1. **ALTA** de un nuevo préstamo en el sistema. Se solicita el **DNI** del usuario y el **ISBN** del libro a prestar. Se comprueba que existen ambos, y si todo está OK se continúa con el préstamo. Los préstamos se hacen partiendo de la fecha actual, y poniendo como fecha de devolución 15 días a partir de la actual. Para trabajar con fechas usaremos la clase Java **LocalDate (Import java.LocalDate)**
2. **DEVOLUCIONES**. Se solicita el **DNI** del usuario y el **ISBN** del libro devuelto. Se buscan ambos en el ArrayList **préstamos** y se elimina el préstamo correspondiente del ArrayList.
3. **PRÓRROGAS**. Se solicita el DNI del usuario y el ISBN del libro para ampliar el plazo de préstamo. Se buscan ambos en el ArrayList **préstamos** y se amplía la Fecha de devolución 15 días más.
4. **LISTADO** de todos los Préstamos en un momento determinado mostrando los datos esenciales.

NOTAS IMPORTANTES: Se recomienda utilizar **DNI**s para usuarios e **ISBN**s para libros imaginarios y cortos, para evitar una entrada de datos de prueba tediosa. **Ejemplo** DNIs "11" "22" – ISBNs "1-11" "2-22".

También se recomienda en el main() cargar un **conjunto de datos de prueba**, tanto en la ArrayList de **Libros** como en la ArrayList de **Usuarios**. Esto es para no partir de cero en cada nueva ejecución del programa, pues cada vez que lo ejecutemos se limpia la memoria **HEAP**, desaparecen todos los objetos creados y nos veremos obligados/as a teclear todos los datos de nuevo. Hacer esto va en contra de las buenas prácticas *de* código "limpio", pero es sólo circunstancial mientras duren las pruebas de funcionamiento del código. Suponiendo que los ArrayList de la aplicación se llaman **libros**, **usuarios** y **prestamos**, sería algo así:

```
public void cargaDatos(){
    libros.add(new Libro("1-11", "El Hobbit", "JRR Tolkien", 3));
    libros.add(new Libro("1-22", "El Silmarillon", "JRR Tolkien", 3));
    libros.add(new Libro("1-33", "El Médico", "N. Gordon", 4));
    libros.add(new Libro("1-44", "Chamán", "N. Gordon", 3));
    libros.add(new Libro("1-55", "Momo", "M. Ende", 2));
    libros.add(new Libro("1-66", "Paraíso inhabitado", "A.M.Matute", 2));
    libros.add(new Libro("1-77", "Olvidado Rey Gudú", "A.M.Matute", 2));
    libros.add(new Libro("1-88", "El último barco", "D.Villar", 3));
    libros.add(new Libro("1-99", "Ojos de agua", "D.Villar", 2));

    usuarios.add(new Usuario("11", "Ana", "621111111"));
    usuarios.add(new Usuario("22", "David", "622222222"));
    usuarios.add(new Usuario("33", "Bea", "623333333"));
    usuarios.add(new Usuario("44", "Lucas", "624444444"));
    usuarios.add(new Usuario("55", "Carlota", "625555555"));
    usuarios.add(new Usuario("66", "Juan", "626666666"));
}

public static void main(String[] args) {
    Biblioteca b=new Biblioteca();
    b.cargaDatos();
    b.menu();
}
```

- Todo el código debe de cumplir con los estándares de Java en cuanto al uso de Mayúsculas/minúsculas en los identificadores utilizados para variables, clases, atributos, métodos, etc.
- Todo el código deberá estar desarrollado de forma modular. No puede haber ningún método de más de 30 líneas de código.
- **Siguiendo todas las especificaciones enunciadas, el ejercicio se valorará sobre 6 puntos. Los 4 puntos restantes de la tarea se enunciarán** en la tutoría del 11/01/2024 y se referirán a añadir funcionalidad a la aplicación "base" propuesta en este documento.
- Hay que entregar los 4 archivos .java del ejercicio. Se pueden entregar sueltos, pero si se hace el ejercicio con BlueJ, os agradecería que me entreguéis la carpeta completa del proyecto (comprimida), igual que las versiones de la aplicación de cuentas bancarias que os dejé en la tutoría del 21/12.
- Como sugiere el enunciado, entregad por favor el ejercicio con un método **cargaDatos()** que sirva para cargar datos (al menos de libros y usuarios) al arrancar la aplicación.

Tutoría 11/01/2024 – Funcionalidad restante para la aplicación BIBLIOTECA (4 puntos)

(+1 pto) Añadir las siguientes opciones al menú de “Gestión de préstamos” y hacer que funcionen.

1. **LISTADO** de todos los libros en préstamos que tiene un usuario/a en un momento determinado. Se solicita el DNI del usuario/a en cuestión. (Se refiere sólo los que ese usuario tiene actualmente en préstamos, no a los libros que tuvo en préstamo en algún momento pero ya ha devuelto)
2. **LISTADO** de todos los préstamos **ACTIVOS** para un libro en concreto, mostrando los datos de los usuarios/as que lo tienen en préstamo. Se solicita el ISBN del libro en cuestión. (nos referimos a los préstamos activos de un libro).

(+1 pto)

- Al prestar un libro se decrementa en una unidad el número de ejemplares disponibles del libro y se informa con un mensaje de cuantas unidades permanecen disponibles a partir de ese momento.
- Al devolver un libro se incrementa en una unidad el número de ejemplares disponibles del libro y se informa con un mensaje de cuantas unidades permanecen disponibles a partir de ese momento.

(+1 pto)

- No se presta un libro si no hay ejemplares disponibles, hay que hacer esta comprobación antes de prestar un libro, e informar al usuario si no hay unidades disponibles.
- No se presta un libro a un usuario si ya tiene en préstamo otro ejemplar del mismo libro.

(+1 pto)

Se debería controlar que se cumplen los plazos de devolución, pero como estamos trabajando en memoria y sin persistencia, no podemos estar esperando a que venzan los plazos de devolución. En su lugar lo que se exige en este punto es que **cuando se produzca una devolución** el sistema muestre un mensaje indicando **cuantos días ha estado el libro prestado**.