

Diseño Orientado a Objetos

Operación prestarLibro

Paso 1: Captura de Requisitos

¿Qué quiere el cliente?

Una aplicación para gestionar una biblioteca. Dicha aplicación debe tratar la siguiente información:

- Relación de todos los socios. Pada socio se recoge su nombre, el número de socio, dirección, y la referencia de los ejemplares que tiene en préstamo. Se pueden tener 3 libros en préstamo como máximo.
- Fondos de la biblioteca, donde se recopila la información de todos los ejemplares de la biblioteca. Por cada libro se guarda su código de identificación, título, autor, estado (libre, prestado o desaparecido). Hay algunos libros que no se pueden prestar, como las enciclopedias, diccionarios, o libros especialmente raros.
- Relación de ejemplares prestados. Por cada ejemplar prestado se guarda su código, el número de socio que lo tiene en préstamo, la fecha del préstamo y la fecha de devolución (el tiempo máximo de préstamo son 15 días).

La primera funcionalidad que se quiere diseñar para esta aplicación es la correspondiente al préstamo de un libro.

Paso 2: Análisis y Diseño (I)

Clases involucradas (guía de Booch)

Identificar a partir de los requisitos, las entidades que se deben manejar en el proceso. La **guía de Booch** nos ofrece pequeños trucos, para intuir qué clases hay que definir en la aplicación, cuáles son TAD o Singleton y qué atributos contiene cada una.

En la siguiente transparencia se marcan en negrita las clases identificadas inicialmente y en cursiva sus atributos. También aparecen subrayados los requisitos establecidos por el cliente.

Paso 2: Análisis y Diseño (II)

Clases involucradas (guía de Booch)

Una aplicación para gestionar una **biblioteca**. Dicha aplicación debe tratar la siguiente información:

- **Relación de todos los socios.** Poda **socio** se recoge su *nombre*, el *número de socio*, *dirección*, y la referencia de los *ejemplares que tiene en préstamo*. Se pueden tener 3 libros en préstamo como máximo.
- **Fondos de la biblioteca**, donde se recopila la información de todos los ejemplares de la biblioteca. Por cada **libro** se guarda su *código de identificación*, *título*, *autor*, *estado* (libre, prestado o desaparecido). Hay algunos libros que no se pueden prestar, como las enciclopedias, diccionarios, o libros especialmente raros.
- **Relación de ejemplares prestados.** Por cada **ejemplar prestado** se guarda su *código*, el *número de socio* que lo tiene en préstamo, la *fecha del préstamo* y la *fecha de devolución* (el tiempo máximo de préstamo son 15 días).

La primera funcionalidad que se quiere diseñar para esta aplicación es la correspondiente al préstamo de un libro.

Paso 2: Análisis y Diseño (III)

Proceso que se desea diseñar (guía de Ellis)

Una aplicación para gestionar una **biblioteca**. Dicha aplicación debe tratar la siguiente información:

- **Relación de todos los socios.** Poda **socio** se recoge su *nombre*, el *número de socio*, *dirección*, y la referencia de los *ejemplares que tiene en préstamo*. Se pueden tener 3 libros en préstamo como máximo.
- **Fondos de la biblioteca**, donde se recopila la información de todos los ejemplares de la biblioteca. Por cada **libro** se guarda su *código de identificación*, *título*, *autor*, *estado* (libre, prestado o desaparecido). Hay algunos libros que no se pueden prestar, como las enciclopedias, diccionarios, o libros especialmente raros.
- **Relación de ejemplares prestados.** Por cada **ejemplar prestado** se guarda su *código*, el *número de socio* que lo tiene en préstamo, la *fecha del préstamo* y la *fecha de devolución* (el tiempo máximo de préstamo son 15 días).

La primera funcionalidad que se quiere diseñar para esta aplicación es la correspondiente al **préstamo de un libro**.

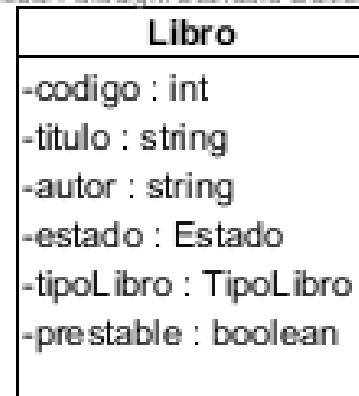
Paso 2: Análisis y Diseño (IV)

Definición de las Clases identificadas (TAD y Singleton)

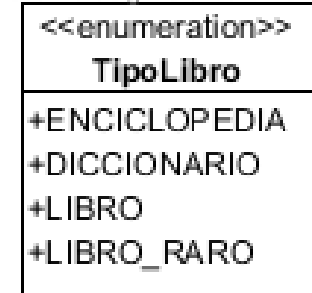
Visual Paradigm Standard Edition(Universidad del Pais Vasco)



Visual Paradigm Standard Edition(Uni



Visual Paradigm Standard Edition(Uni



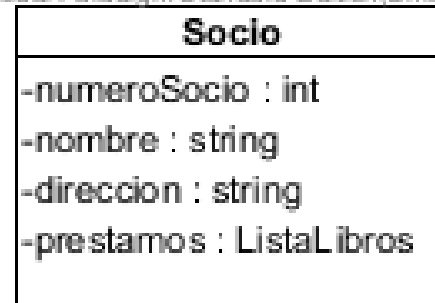
Visual Paradigm Standard Edition(Universidad



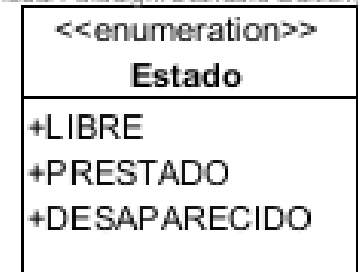
Visual Paradigm Standard Edition(Universidad del Pais Vasc



Visual Paradigm Standard Edition(Universic



Visual Paradigm Standard Edition(Uni



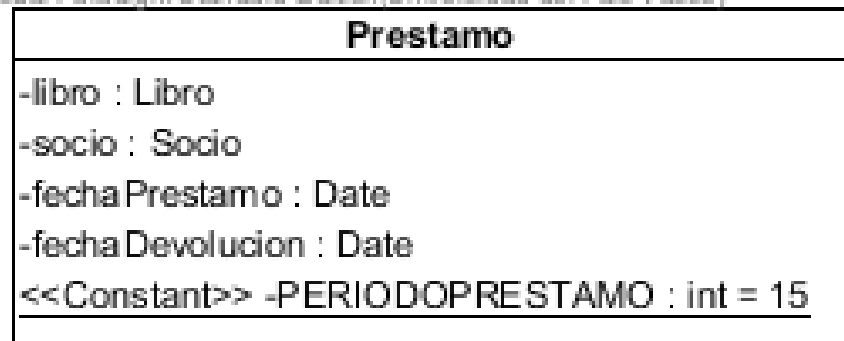
Paso 2: Análisis y Diseño (V)

Definición de las Clases identificadas (TAD y Singleton)

Visual Paradigm Standard Edition (Universidad del País Vasco)



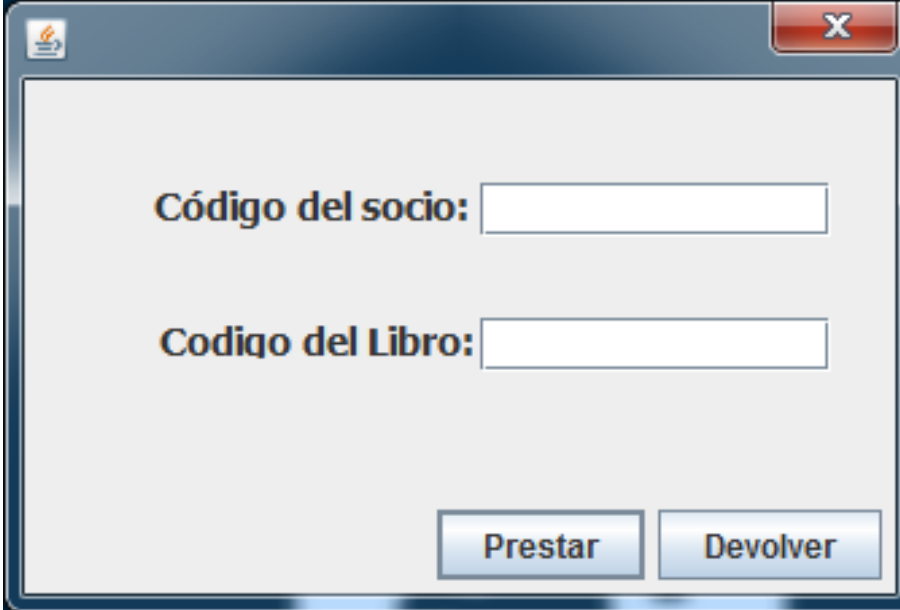
Visual Paradigm Standard Edition (Universidad del País Vasco)



Paso 2: Análisis y Diseño (VI)

Aspecto de la ventana desde la que arranca el préstamo del Libro

VentanaPrestamos



The image shows a screenshot of a software window titled "VentanaPrestamos". The window has a standard Windows-style title bar with a small icon on the left and a close button (X) on the right. The main content area is light gray and contains two text input fields. The first field is labeled "Código del socio:" and the second is labeled "Codigo del Libro:". Below these fields are two buttons: "Prestar" and "Devolver". The window has a blue border and a small icon in the top-left corner.

Paso 2: Análisis y Diseño (VII)

Algoritmo del proceso a diseñar

1. El usuario teclea el código del libro, *pLibro*, el número del socio, *pSocio* y pulsa el botón “Prestar”
2. Comprobar si el libro es apto para el préstamo

Si es prestable y está disponible

- 3.1. Comprobar que el socio no tiene ya el máximo de préstamos

Si no tiene el máximo de préstamos,

- 4.1. Crear el préstamo → unPrestamo
- 4.2. Añadir el libro prestado a la lista de préstamos del Socio
- 4.3. Incluir unPestamo en la relación de préstamos
- 4.4. Cambiar el estado del libro

Si no es apto,

5. Lanzar la excepción MaxPrestamosException

Si no es prestable,

6. Lanzar la excepción LibroNoPrestableException

Paso 2: Análisis y Diseño (VIII)

Incluir en cada clase, los métodos necesarios para prestarLibro

NOTA!!! Cada clase debe contener los métodos para gestionar sus atributos

1.a. Desde VentanaPréstamos se accede al Singleton **FondosBiblioteca**:

Definir en **FondosBiblioteca**:

prestarLibro (pCodLibro: int, pCodSocio: int): void

Esta clase sólo contiene un atributo que es de la clase ListaLibros, así que tiene que delegar en la clase ListaLibros la implementación de prestarLibro.

1.b. En la clase **ListaLibros** se debe continuar la implementación del método

Definir en **ListaLibros** :

prestarLibro (pCodLibro: int, pCodSocio: int): void

1.c. Buscar el Libro a partir de su código:

Definir en **ListaLibros** :

buscar(pCodLibro: int): Libro

Una vez obtenida la instancia del libro a prestar, se debe comprobar si es apto para el préstamo.

Paso 2: Análisis y Diseño (IX)

2. Comprobar si el **Libro** es apto para ser prestado:

Definir en **Libro**:

esAptoPrestamo(): boolean

esPrestable(): boolean

estaDisponible(): boolean

Si el libro es apto, hay que comprobar si el socio tiene el máx. de préstamos

3.a. Buscar el Socio a partir de su código:

Definir en **RelacionSocios**:

buscar(pCodSocio: int): Socio

3.b. Comprobar si tiene en préstamo el máximo permitido:

Definir en **Socio**:

tieneMaximoLibrosEnPrestamo(): boolean

Paso 2: Análisis y Diseño (X)

4.a. Crear el préstamo → invocar a la constructora de **Prestamo**

4.b. Añadir el libro prestado a la lista de préstamos del socio:

Definir en **Socio**:

anadirLibroPrestado(pLibro: Libro): void

4.c. Incluir el nuevo préstamo en la relación de préstamos

Definir en **RelacionPrestamos**:

anadirPrestamo(pPrestamo: Prestamo): void

4.d. Cambiar el estado del libro

Definir en **Libro**:

asgEstado(pEstado: Estado): void

DEFINIR LAS CLASES CORRESPONDIENTES A LAS EXCEPCIONES NECESARIAS

- **NoDisponibleException**
- **NoPrestableException**
- **TieneMaxPrestamosException**

Paso 2: Análisis y Diseño (XI)

Incluir en cada clase, los métodos necesarios

Visual Paradigm Standard Edition(Universidad del País Vasco)

| |
|---|
| <<Singleton>> FondosBiblioteca |
| -listaLibros : ListaLibros <u>-mFondosBiblioteca : FondosBiblioteca</u> |
| <<constr>> -FondosBiblioteca() <u>+getFondosBiblioteca() : FondosBiblioteca</u> <u>+prestarLibro(pCodLibro : int, pCodSocio : int) : void</u> |

Visual Paradigm Standard Edition(Universidad del País Vasco)

| |
|--|
| ListaLibros |
| -lLibros : Coleccion<libro> |
| <<constr>> +ListaLibros() +buscarLibro(pCodLibro : int) : Libro -getIterador() : Iterator<Libro> +anadirLibro(pLibro : Libro) : void +tieneMaxLibros() : boolean |

Paso 2: Análisis y Diseño (XI)

Incluir en cada clase, los métodos necesarios

Visual Paradigm Standard Edition (Universidad del País Vasco)

| Libro |
|---|
| <ul style="list-style-type: none">-codigo : int-titulo : string-autor : string-estado : Estado-tipoLibro : TipoLibro-prestable : boolean |
| <ul style="list-style-type: none"><<constr>> +Libro(pCod : int, pTit : string, pAut : string, pTipo : TipoLibro)-esPrestable() : boolean-estaDisponible() : boolean+setEstado(pEstado : Estado) : void+esAptoPrestamo() : boolean+igualCodigo(pCodigo : int) : boolean |

Paso 2: Análisis y Diseño (XII)

Incluir en cada clase, los métodos necesarios

Visual Paradigm Standard Edition(Universidad del País Vasco)

| <<Singleton>> RelacionPrestamos |
|---|
| -listaPrestamos : Coleccion<Prestamo> <u>-mRelacionPrestamos : RelacionPrestamos</u> |
| <<constr>> -RelacionPrestamos() <u>+getRelacionPrestamos() : RelacionPrestamos</u> +buscarPrestamo(pCodLibro : int) : Prestamo +anadirPrestamo(pPrestamo : Prestamo) : void +borrarPrestamo(pPrestamo : Prestamo) : void -getIterador() : Iterator<Prestamo> |

Edición(Universidad del País Vasco)

| Prestamo |
|---|
| -libro : Libro -socio : Socio -fechaPrestamo : Date -fechaDevolucion : Date <u><<Constant>> -PERIODOPRESTAMO : int = 15</u> |
| <<constr>> +Prestamo(pLibro : Libro, pSocio : Socio) : Prestamo +getFechaDevolucion() : Fecha |

Paso 2: Análisis y Diseño (XIII)

Incluir en cada clase, los métodos necesarios

Visual Paradigm Standard Edition (Universidad del País Vasco)

| <<Singleton>> RelacionSocios |
|--|
| -listaSocios : Coleccion<Socio> <u>-mRelacionSocios : RelacionSocios</u> |
| <<constr>> -RelacionSocios() <u>+getRelacionSocios() : RelacionSocios</u> -getIterador() : Iterator<Socio> +buscarSocio() : Socio +tieneMaximoLibrosEnPrestamo(pSocio : int) : boolean |

Visual Paradigm Standard Edition (Universidad del País Vasco)

| Socio |
|---|
| -numeroSocio : int -nombre : string -direccion : string -prestamos : ListaLibros |
| <<constr>> +Socio(pNumSocio : int, pNombre : string, pDireccion : string) +tieneMaximoLibrosEnPrestamo() : boolean +anadirLibroPrestado(pLibro : Libro) : void +tieneLibroPrestado(pLibro : int) : boolean |

Paso 2: Análisis y Diseño (XIV)

Diagrama de secuencia

Paso 3: Implementación (I)

Clase FondosBiblioteca

//ATRIBUTOS

```
private ListaLibros listaLibros;
```

```
private static mFondosBiblioteca = new FondosBiblioteca();
```

//CONSTRUCTORA

```
private FondosBiblioteca(){
```

```
    listaLibros = new ListaLibros();
```

```
}
```

```
public static FondosBiblioteca getFondosBiblioteca(){
```

```
    return mFondosBiblioteca;
```

```
}
```

//MÉTODOS

```
public void prestarLibro(int pCodLibro, int pCodSocio){
```

```
    listaLibros.prestarLibro(int pCodLibro, int pCodSocio);
```

```
}
```

Paso 3: Implementación (II)

Clase ListaLibros

//ATRIBUTOS

```
private List<Libros> lLibros;
```

//CONSTRUCTORA

```
private ListaLibros(){
```

```
    lLibros = new ArrayList<Libro>();
```

```
}
```

//MÉTODOS

```
public void prestarLibro(int pCodLibro, int pCodSocio){
```

```
    listaLibros. prestarLibro(int pCodLibro, int pCodSocio);
```

```
}
```

```
public Iterator<Libro> getIterador(){
```

```
    return lLibros.iterator();
```

```
}
```

Paso 3: Implementación (III)

```
public Libro buscarLibro(int pCodLibro){
    boolean encontrado=false;
    Libro unLibro = null;
    it = getIterador();
    while (!encontrado && it.hasNext()){
        unLibro = it.next();
        if (unLibro.tieneCodigo(pCodLibro) == true){
            encontrado = true;
        }
    }
    return unLibro;
}

public void anadirLibro(Libro pLibro){
    lLibros.add(pLibro);
}
```

Paso 3: Implementación (IV)

```
public boolean tieneMaxLibros(){  
    if (lLibros.size()>=3)  
        return true;  
    else  
        return false;  
}  
  
public void anadirLibro(Libro pLibro){  
    lLibros.add(pLibro);  
}
```