

# Programación Avanzada

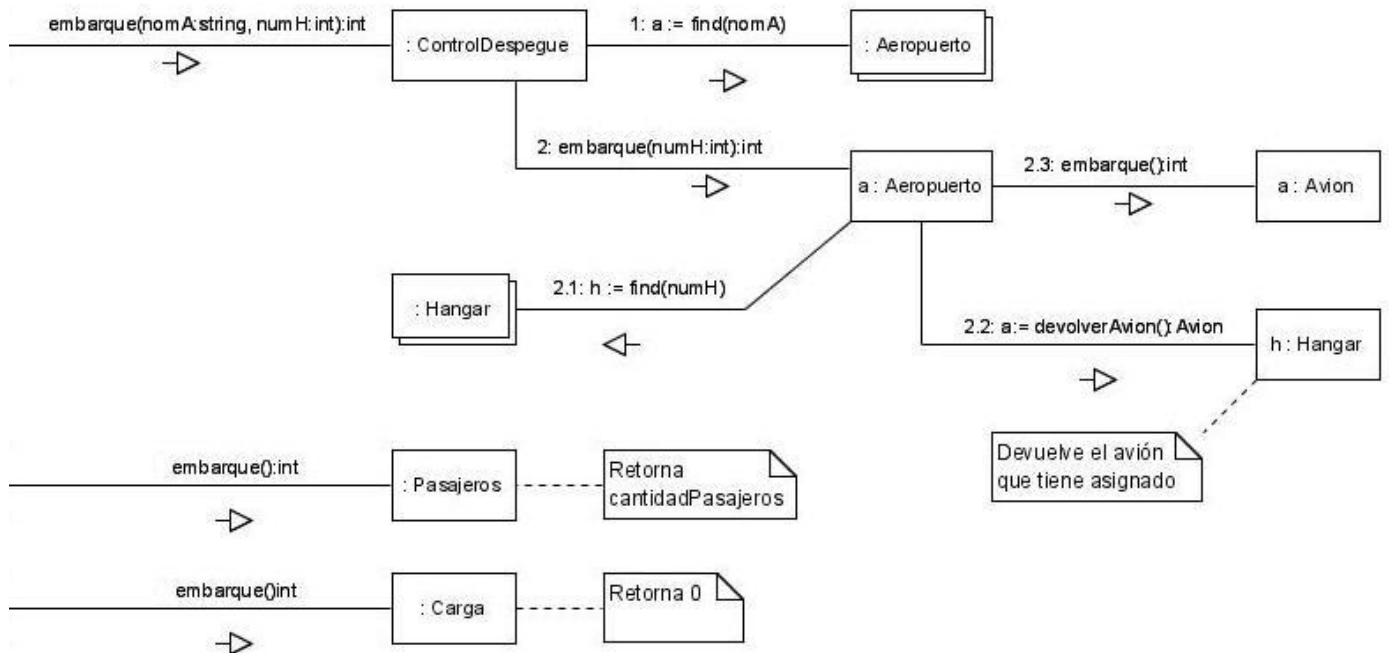
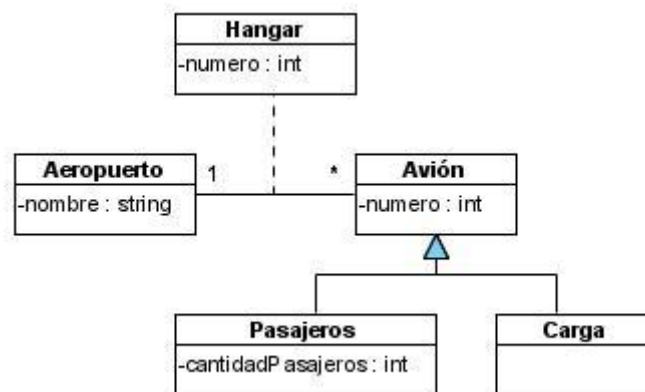
## PRÁCTICO 5

### Ejercicio 1 (básico, imprescindible)

Diseñar la estructura correspondiente al diseño de interacciones de los Ejercicios 1, 2, 3, 4 y 5 del Práctico 4.

### Ejercicio 2 (medio, imprescindible)

Diseñar la estructura correspondiente al modelo de dominio y diseño de interacciones presentado en los siguientes diagramas.



**Ejercicio 3 (medio, de práctica)**

Considerar el documento de Visión del Problema del Ejercicio 10 del Práctico 2 referente al videoclub. Basándose en el Modelo de Dominio construido, realizar el diseño de la operación del sistema `alquilarPelícula` identificada a partir del siguiente caso de uso.

El videoclub tiene como política que para alquilar una película debe haber una reserva previa. Cuando un socio quiere alquilar una película, el sistema chequea que exista la reserva adecuada, y en caso de que la haya, se borra la reserva y se registra el alquiler. En caso de que la reserva no exista, la operación no debe tener efecto sobre el sistema.

**Ejercicio 4 (medio, imprescindible)**

Diseñar la estructura correspondiente al diseño de interacciones de los Ejercicios 7, 8 y 9 del Práctico 4.

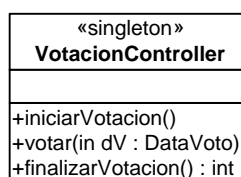
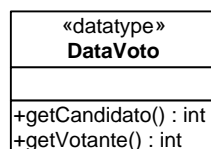
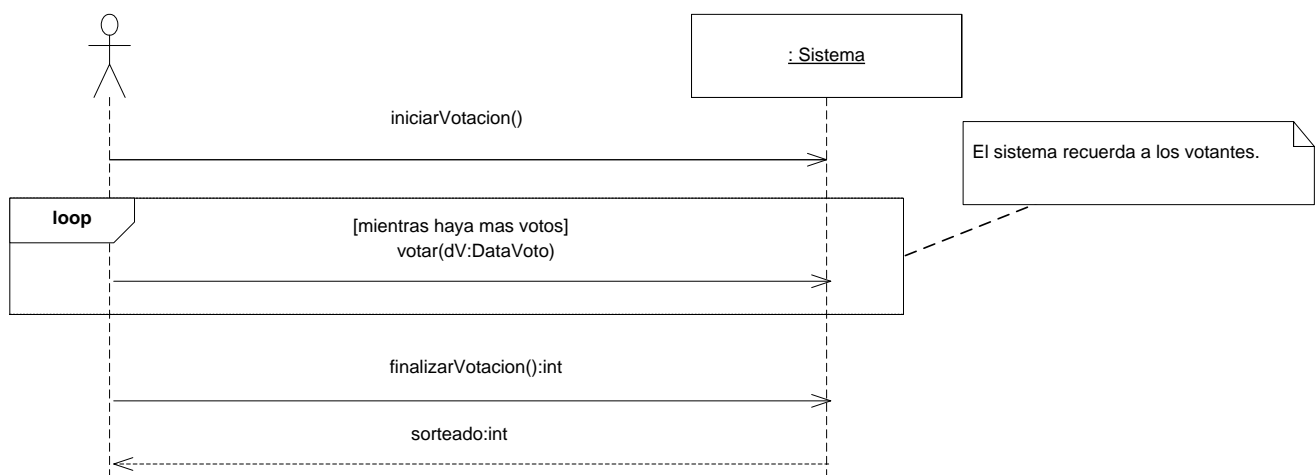
**Ejercicio 5 (medio, imprescindible)**

Se desea desarrollar un sistema embebido en un sencillo reloj digital de dos botones. El reloj permite visualizar la hora actual y modificarla. Presionando el botón A, se cambia a uno de los siguientes modos (en forma circular): *configurar hora*, *configurar minuto* y *normal*. El botón B incrementa (también en forma circular) el valor de la hora cuando se está en modo *configurar hora*, el valor de los minutos cuando se está en modo *configurar minuto* y no realiza ninguna acción en el modo *normal*.

Realizar las actividades de análisis y diseño para desarrollar este sistema.

**Ejercicio 6 (basico, imprescindible)**

Se desea diseñar un sistema de votación genérico, el cual debe permitir comenzar una votación, registrar los votos, y finalizar la votación que realiza un sorteo entre los votantes determinando un ganador. Durante la fase de análisis y diseño se obtuvieron los siguientes artefactos:



**Se pide:**

- i. Modifique el diseño de la clase `VotacionController` teniendo en cuenta que deberá permitir a otros (potencialmente múltiples) sistemas, realizar acciones una vez ejecutada la operación `finalizarVotacion`. Justifique brevemente y realice los diagramas de comunicaciones que muestren como resuelve dicho requerimiento.
- ii. Realice el DCD correspondiente.
- iii. Explique qué patrón(es) de diseño utilizó indicando (para cada patrón) las clases participantes y sus roles.

**Ejercicio 7 (medio, imprescindible)**

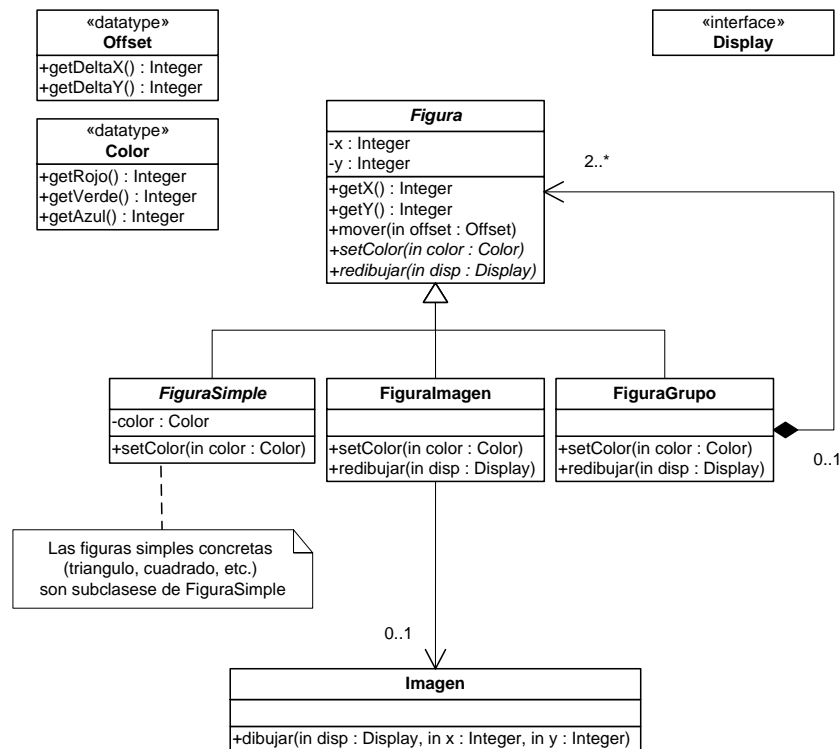
Se desea desarrollar un sistema de dibujo. El mismo deberá permitir manejar figuras simples (por ejemplo cuadrados y círculos), grupos de figuras, y figuras asociadas a una imagen externa. En una primera instancia interesa implementar las siguientes operaciones:

- i. `mover()` que debe desplazar la figura una cierta distancia en horizontal y vertical
- ii. `setColor()` que debe modificar el color de las figuras
- iii. `redibujar()` que debe redibujar las figuras

Por otro lado se debe reutilizar la clase `Imagen`, encargada de dibujar imágenes externas. Esta clase está en uso en otros sistemas en funcionamiento, y por tal motivo no se puede modificar.

Se ha realizado un diseño que soluciona los requerimientos mencionados, el cual se muestra en la figura. Las decisiones de diseño acerca del funcionamiento de las operaciones en las distintas clases se resumen en la siguiente lista:

- i. `Figura::mover()` debe actualizar los atributos `Figura::x` y `Figura::y`, y luego llamar a `redibujar()`
- ii. `FiguraSimple::setColor()` debe actualizar el atributo `FiguraSimple::color`
- iii. `FiguraCompuesta::setColor()` debe invocar a `Figura::setColor()` para todas las figuras agrupadas
- iv. `FiguraCompuesta::redibujar()` debe invocar a `Figura::redibujar()` para todas las figuras agrupadas
- v. `FiguraImagen::setColor()` debe estar implementada para no hacer nada
- vi. `FiguraImagen::redibujar()` debe invocar a `Imagen::dibujar()` con los atributos `Figura::x` y `Figura::y`



Para realizar el diseño de este sistema se aplicaron ciertos patrones de diseño. Indicar cuáles fueron aplicados y justificar por qué lo fueron. Indicar para cada patrón cuáles son las clases participantes.

### Ejercicio 8 (avanzado, imprescindible)

Se le ha encomendado el diseño del motor de un navegador web que proporciona dos modos de navegación: el modo Liviano que usa menos memoria pero no soporta Flash ni otros contenidos multimedia, y el modo Pesado que sí soporta contenidos multimedia y Flash, sin embargo usa más recursos del sistema.

## Parte 1:

Considere que existe una clase Navegador que contiene una operación con la siguiente firma:

```
abrirPagina (string url)
```

que toma una url, baja el contenido a la computadora y muestra la página bajada en la pantalla del PC. Independientemente del modo de navegación, el mecanismo para abrir una página es el mismo y se puede describir usando el siguiente algoritmo:

abrirPagina(url):

1. llamar al método abrirPestaniaNueva()
2. pagina = bajarDatos(url);
3. Si pagina no es null: llamar al método mostrarPagina(pagina)

Las operaciones `abrirPestaniaNueva()` y `mostrarPagina()` son implementadas según el modo en el que esté el navegador. Las clases `NavegadorPesado` y `NavegadorLiviano` se encargan de implementar el comportamiento específico según el modo de navegación en que se encuentre el navegador. La operación `bajarDatos()` se implementa de igual manera en ambos modos.

Se pide realizar un DCD correspondiente al diseño parcial de la operación abrirPagina.

**Parte 2:**

Considere el método `bajarDatos(url)` de la clase `Navegador`, que se encarga de obtener los datos desde internet a partir de un string de la url. Las urls que maneja el navegador son de la siguiente forma:

`<protocolo>://<ruta>`

Por ejemplo `http://twitter.com;` `ftp://mirror.yongbok.net/`

El navegador es capaz de manejar los protocolos HTTP, FTP y HTTPS. La forma de bajar un archivo depende enteramente del protocolo de la URL por lo que debe haber una implementación diferente de `bajarDatos()` para cada protocolo que maneje el navegador.

Se pide realizar un DCD correspondiente al diseño parcial de la operación `bajarDatos()`. Suponga que la clase `Navegador` tiene un método `obtenerProtocolo(url)` que obtiene el protocolo a partir de la url.

**Ejercicio 9 (avanzado, de práctica)**

Se define el conjunto de expresiones aritméticas sobre enteros EXPI de la siguiente manera:

- los enteros pertenecen a EXPI,
- si  $e_1$  pertenece a EXPI entonces (`opuesto  $e_1$` ) pertenece a EXPI, siendo `opuesto` el operador representando el opuesto de un entero, y
- si  $e_1$  y  $e_2$  pertenecen a EXPI entonces ( `$e_1$  mas  $e_2$` ), ( `$e_1$  menos  $e_2$` ), ( `$e_1$  por  $e_2$` ) y ( `$e_1$  division  $e_2$` ) pertenecen a EXPI, siendo `mas`, `menos`, `por` y `division` los operadores de suma, resta, multiplicación y división entre enteros.

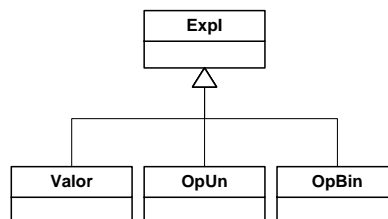
A modo de ejemplo, (`5 division 3`) y (`opuesto ((4 division 0) division 2)`) son dos elementos de EXPI.

Sea `calcularValor` una operación que dada una expresión de EXPI devuelve el valor de la expresión. Esta función devolverá el valor `error` en el caso de una división por 0. A modo de ejemplo, la operación devuelve 1 para el primer ejemplo y `error` para el segundo.

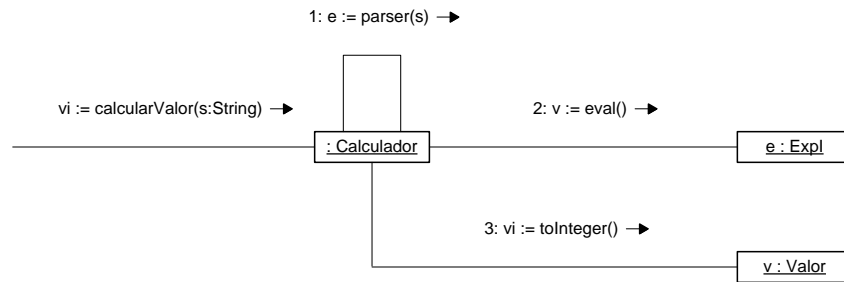
Se desea diseñar un sistema que permita calcular el valor de una expresión de EXPI de manera que facilite la incorporación de nuevos operadores.

Se pide:

- Modificar el siguiente diagrama de clases que presenta los elementos de EXPI teniendo en cuenta que se desea que futuros agregados de nuevos operadores (por ej. el operador `módulo`) no implique cambios de los elementos ya definidos. Indicar qué clases deben ser abstractas.



- El diseño de la interacción para la operación `calcularValor(s:String)` es el siguiente:



Diseñar la interacción de la operación `eval()` de la clase `Expl` y diseñar la estructura correspondiente. Justificar cada uno de los elementos del DCD con relación a los del diagrama de comunicación.

### Ejercicio 10 (medio, imprescindible)

Se desea diseñar un servicio que provea un mecanismo de impresión en impresoras de diferentes fabricantes.

Cada fabricante provee una clase cuyas instancias funcionan de driver de la impresora; por ejemplo, instancias de la clase **DriverHP** proveen servicios de impresión en impresoras HP, instancias de la clase **DriverEpson** provee servicios de impresión en impresoras Epson, etc. Todas estas clases realizan la interfaz **IDriver**, que ofrece dos operaciones: la operación **enLinea** indica si la impresora en el puerto dado puede imprimir o no, y la operación **imprimir** imprime el dato en la impresora en el puerto dado, dando un error cuando la impresora no está en línea.

Se quiere ofrecer un servicio mediante una clase **Impresora** cuyas instancias mantengan el puerto de la impresora en la que imprime y el driver particular que utiliza. Si la impresora no está en línea, la operación **imprimir** también da un error. Por otra parte, se quiere ofrecer además otra clase que brinde una funcionalidad similar a la anterior, pero que simule estar siempre en línea. En esta clase, cuando la impresora asociada no esté en línea, se deben almacenar los datos a imprimir para imprimirlos en el próximo pedido de impresión en que la impresora esté en línea. Ambas clases deben ofrecer sus servicios por medio de la interfaz **IImpresora**.

«interface» <b>IDriver</b>
<i>enLinea(in p : Puerto) : bool</i> <i>imprimir(in p : Puerto, in d : Dato)</i>

<b>DriverHP</b>
<i>enLinea(in p : Puerto) : bool</i> <i>imprimir(in p : Puerto, in d : Dato)</i>

<b>DriverEpson</b>
<i>enLinea(in p : Puerto) : bool</i> <i>imprimir(in p : Puerto, in d : Dato)</i>

«interface» <b>IImpresora</b>
<i>enLinea() : bool</i> <i>imprimir(in d : Dato)</i>

Se pide:

- Realizar el diseño de la estructura aplicando patrones de diseño, presentándolo en un diagrama de clases.
- Explicar qué patrón(es) de diseño utilizó indicando (para cada patrón) las clases participantes y sus roles.

### Ejercicio 11 (avanzado, de práctica) :: Laboratorio 2008

- Considerando la realidad de prácticos anteriores y el diseño de interacciones realizado en el Ejercicio 11 del Práctico 4, realizar el Diagrama de Clases de Diseño correspondiente.
- Considere ahora el siguiente anexo a la descripción de la realidad presentada en el Práctico 2:

Las tareas pueden ser simples o complejas. Las tareas complejas se dividen en una secuencia de tareas que deberán llevarse a cabo para completarlas. El conjunto de tipos de artículo sobre los que aplica una tarea compleja, es la intersección de los tipos de artículo de las tareas que la forman. Una tarea compleja no podrá completarse hasta que todas aquellas que la forman hayan sido completadas.

Además de registrar horas sobre una tarea simple, los técnicos podrán registrar horas de supervisión en las tareas complejas (siempre y cuando les esté asignada). La cantidad total de horas invertidas en una tarea compleja, se calcula como el total de horas sobre la propia tarea más la suma de las horas de las tareas que la forman (que a su vez, podrían ser complejas).

Finalmente, a un reclamo se podrán asignar tanto tareas simples como complejas. Sin embargo, no se podrán asignar a un mismo reclamo dos veces la misma tarea (ya sea directamente o como parte de una tarea compleja).

Además, considere el siguiente caso de uso agregado en esta etapa:

Nombre	<b>Consultar información de tarea</b>
Actores	Administrador
Sinopsis	El usuario ingresa el código de la tarea que desea consultar. El sistema muestra la información de la tarea consultada (código, nombre, descripción, si es simple o compleja, y los tipos de artículos a los que aplica). Si la tarea es compleja, se muestra además la misma información de cada una de las tareas que la componen (y así recursivamente).

Se pide:

- Realizar los diagramas de interacción necesarios para el caso de uso indicado y el diagrama de clases de diseño para que tengan en cuenta los requerimientos adicionales.
- Explique qué patrones de diseño utilizó indicando (para cada patrón) las clases participantes y sus roles.