# 1. Vocabulario y conceptos

1. ¿Por qué debe aplicar un patrón de diseño siempre que sea posible?

   **Solución** COMPLETAR.

2. Enuncie una posible desventaja al usar un patrón de diseño.

   **Solución** COMPLETAR.

3. Explique en qué fase del ciclo de producción de un sistema de software usaría los patrones de diseño.

   **Solución** COMPLETAR.

4. Aclare la o las razones por las cuales seleccionaría el patrón Strategy en lugar de Visitor en el caso en que la función asignada a ambos deba recorrer, al menos, cierta porción de una estructura de objetos.

   **Solución** COMPLETAR.

5. Describa la forma en que es conveniente seleccionar un patrón de diseño.

   **Solución** COMPLETAR.

6. Explique la diferencia entre herencia de clases y herencia de tipos o interfaces. ¿Cuál es la que más conviene usar desde el punto de vista del diseño? ¿Por qué?

   **Solución** COMPLETAR.

# 2. Aplicación de patrones de diseño

En todos los problemas que siguen se debe aplicar el o los patrones de diseño indicados para generar un DOO. Cada diseño debe documentarse como se explicó en clase y como se pide en la práctica de diseño.

1. Aplicar el patrón Command al problema «Estación climatométrica» descrito en la práctica de diseño.

**Solución**

| **Pattern based on** | **Recopilación de Datos**<br>Command | | |
|---|---|---|---|
| **because** | ▪ Facilita añadir nuevas características al sistema. Por ejemplo se puede crear una nueva orden que emita alarmas según los valores.<br>▪ Desacopla el temporizador del muestrador y el calculador. | | |
| **where** | Invocador | **is** | Temporizador |
| | *Orden* | ***is*** | *Orden* |
| | *Ejecutar()* | ***is*** | *Ejecutar()* |
| | OrdenConcreta | **is** | OrdenMuestrar |
| | Ejecutar() | **is** | Ejecutar() |
| | Receptor | **is** | Muestrador |
| | Accion() | **is** | Muestrar() |
| | OrdenConcreta | **is** | OrdenCalcular |
| | Ejecutar() | **is** | Ejecutar() |
| | Receptor | **is** | Calculador |
| | Accion() | **is** | Calcular() |
| **comments** | ▪ El muestrador debe conocer a los sensores y el repositorio.<br>▪ El calculador debe conocer al repositorio. | | |

2. Aplicar el patrón Command al problema «Acceso a un canal de navegación» descrito en la práctica de diseño.

**Solución**

| **Pattern** | **Llegada de un barco** | | |
|---|---|---|---|
| **based on** | Command | | |
| **because** | ▪ Las mismas ordenes funcionan si se agregan esclusas.<br>▪ Permite encolar las ordenes para cuando lleguen varios barcos. | | |
| **where** | Invocador | **is** | ColaIzquierda |
| | *Orden* | ***is*** | *Orden* |
| | *Ejecutar()* | ***is*** | *Ejecutar()* |
| | OrdenConcreta | **is** | Nivelar |
| | Ejecutar() | **is** | Ejecutar() |
| | Receptor | **is** | Nivelador |
| | Accion() | **is** | Nivelar() |
| | OrdenConcreta | **is** | Ingresar |
| | Ejecutar() | **is** | Ejecutar() |
| | Receptor | **is** | Ingresador |
| | Accion() | **is** | Ingresar() |
| | OrdenConcreta | **is** | Egresar |
| | Receptor | **is** | Egresador |
| | Accion() | **is** | Egresar() |
| **comments** | ▪ El nivelador debe conocer los sensores de profunidad y la bomba.<br>▪ El ingresador debe conocer la puerta de entrada y el sensor ocupación.<br>▪ El egresador debe conocer la puerta de entrada y el sensor ocupación.<br>▪ El invocador mantiene la cola de llegada y se encarga de ejecutar las ordenes cuando sea posible. | | |

3. Aplicar el patrón Composite al problema «Mantenimiento de catálogos» descrito en la práctica de diseño. Hacerlo para definir con detalle la estructura de los catálogos. Cada producto ofertado en un catálogo tiene las siguientes características: nombre, código, descripción, varios niveles de secciones (por ejemplo, Ferretería, Tornillos y Bulones, Tornillos, Tornillos de acero, Tornillos de 1,5mm, etc.), precio, marca, color, etc.

**Solución**

| **Pattern** | **Estructura del catálogo** | | |
|---|---|---|---|
| **based on** | Composite | | |
| **because** | ▪ Permite tratar de manera uniforme a las secciones y a los productos. | | |
| **where** | *Componente* | ***is*** | *Componente* |
| | *Operacion()* | ***is*** | *Listar()* |
| | *Añadir(Componente)* | ***is*** | *Añadir(Componente)* |
| | *Eliminar(Componente)* | ***is*** | *Eliminar(Componente)* |
| | *ObtenerHijo(Int)* | ***is*** | *ObtenerHijo(Int)* |
| | Compuesto | **is** | Seccion |
| | Operacion() | **is** | Listar() |
| | Añadir(Componente) | **is** | Añadir(Componente) |
| | Eliminar(Componente) | **is** | Eliminar(Componente) |
| | ObtenerHijo(Int) | **is** | ObtenerHijo(Int) |
| | Hoja | **is** | Producto |
| | Operacion() | **is** | Listar() |
| **comments** | Cada sección puede dividirse en subsecciones tantos niveles como sea necesario, hasta llegar a un producto. | | |

4. Respecto del problema 3, aplicar el patrón Iterator para recorrer la estructura de un catálogo.

**Solución**

| **Pattern** | **Recorrido del catálogo** | | |
|---|---|---|---|
| **based on** | Iterator | | |
| **because** | ▪ Permite varios recorridos sobre el catálogo.<br>▪ Oculta la estructura del catálogo.<br>▪ Define una interfaz uniforme. | | |
| **where** | Cliente | **is** | Informes |
| | *Agregado* | ***is*** | *Agregado* |
| | *crearIterador()* | ***is*** | *crearIterador()* |
| | AgregadoConcreto | **is** | Catalogo |
| | crearIterador() | **is** | crearIterador() |
| | *Iterador* | ***is*** | *Iterador* |
| | *primero()* | ***is*** | *primero()* |
| | *siguente()* | ***is*** | *siguente()* |
| | *haTerminado()* | ***is*** | *haTerminado()* |
| | *elementoActual()* | ***is*** | *elementoActual()* |
| | IteradorConcreto | **is** | BFS |
| | primero() | **is** | primero() |
| | siguente() | **is** | siguente() |
| | haTerminado() | **is** | haTerminado() |
| | elementoActual() | **is** | elementoActual() |
| **comments** | Explicación coloquial de la relación entre los elementos del patrón y los elementos del diseño concreto; otros comentarios adicionales que ayuden a entender cómo se aplica el patrón de diseño | | |

5. Defina operaciones de búsqueda por precio, código, sección y palabra en la descripción aplicando el patrón Visitor sobre cada catálogo del problema 3. Muestre código de ejemplo de cómo se implementaría una de las búsquedas.

**Solución**

| **Pattern** | **Operaciones de busqueda** | | |
|---|---|---|---|
| **based on** | Visitor | | |
| **because** | ▪ Permite modificar y añadir nuevas búsquedas, sin alterar el código del catálogo.<br>▪ Agrupa toda las funcionalidades de búsqueda en un mismo lugar, en vez de distribuirlas por el catálogo. | | |
| **where** | Cliente | **is** | COMPLETAR |
| | EstructuraDeObjetos | **is** | Catalogo |
| | *Elemento* | ***is*** | *Elemento* |
| | *Aceptar(Visitante)* | ***is*** | *Buscar(Busqueda)* |
| | ElementoConcretoA | **is** | Producto |
| | Aceptar(Visitante) | **is** | Buscar(Busqueda) |
| | OperacionA() | **is** | ObtenerPrecio() |
| | ElementoConcretoB | **is** | Seccion |
| | Aceptar(Visitante) | **is** | Buscar(Busqueda) |
| | OperacionB() | **is** | ObtenerNombre() |
| | *Visitante* | ***is*** | *Busqueda* |
| | *VisitanteConcretoElementoA(ElementoConcretoA)* | ***is*** | *EnProducto(Producto)* |
| | *VisitanteConcretoElementoB(ElementoConcretoB)* | ***is*** | *EnSeccion(Seccion)* |
| | VisitanteConcreto1 | **is** | PorPrecio |
| | VisitarElementoConcretoA(ElementoConcretoA) | **is** | EnProducto(Producto) |
| | VisitarElementoConcretoB(ElementoConcretoB) | **is** | EnSeccion(Seccion) |
| | VisitanteConcreto2 | **is** | PorSeccion |
| | VisitarElementoConcretoA(ElementoConcretoA) | **is** | EnProducto(Producto) |
| | VisitarElementoConcretoB(ElementoConcretoB) | **is** | EnSeccion(Seccion) |
| **comments** | COMPLETAR. | | |

6. Use el patrón Iterator para muestrear los valores sensados en el problema 1.

**Solución**

| **Pattern** | **Recorrido de sensores** | | |
|---|---|---|---|
| **based on** | Iterator | | |
| **because** | ▪ Permite varios recorridos sobre los sensores.<br>▪ Oculta la estructura donde se almacenan los sensores.<br>▪ Define una interfaz uniforme. | | |
| **where** | Cliente | **is** | Muestrador |
| | *Agregado* | **is** | *Agregado* |
| | *crearIterador()* | **is** | *crearIterador()* |
| | AgregadoConcreto | **is** | Sensores |
| | crearIterador() | **is** | crearIterador() |
| | *Iterador* | **is** | *Iterador* |
| | *primero()* | **is** | *primero()* |
| | *siguente()* | **is** | *siguente()* |
| | *haTerminado()* | **is** | *haTerminado()* |
| | *elementoActual()* | **is** | *elementoActual()* |
| | IteradorConcreto | **is** | AccesoAlfabetico |
| | primero() | **is** | primero() |
| | siguente() | **is** | siguiente() |
| | haTerminado() | **is** | haTerminado() |
| | elementoActual() | **is** | elementoActual() |
| **comments** | El muestrador utilizará un iterador para tomar cada muestra y poder actualizar el repositorio. | | |

7. La empresa de venta por correo desea poner sus catálogos en su sitio Web. Además desea que los catálogos puedan mostrarse ordenados de diferente forma. Por ejemplo, orden alfabético de productos, de secciones de nivel 1, por precio, etc. Aplique el patrón Strategy para implementar esta funcionalidad.

**Solución**

| **Pattern** | **Orden del catálogo** | | |
|---|---|---|---|
| **based on** | Strategy | | |
| **because** | ▪ Permite cambiar el orden del catálogo en tiempo de ejecución. | | |
| **where** | Contexto | **is** | Catálogo |
| | InterfazContexto() | **is** | (ver comments) |
| | estrategia | **is** | criterio |
| | *Estrategia* | ***is*** | *Orden* |
| | *InterfazAlgoritmo()* | ***is*** | *Ordenar()* |
| | EstrategiaConcretaA | **is** | Alfabetico |
| | InterfazAlgoritmo() | **is** | Ordenar() |
| | EstrategiaConcretaB | **is** | PorPrecio |
| | InterfazAlgoritmo() | **is** | Ordenar() |
| **comments** | La interfaz del contexto es la que define el composite. Deben agregarse métodos para intercambiar hermanos. | | |

8. Considere el problema del banco de la práctica de diseño. Ahora suponga que el banco ha comprado otros bancos. Cada banco comprado tiene su propia forma de definir una cuenta (caja de ahorro, cuenta corriente, en pesos y dólares), plazo fijo, cliente, etc. Estudie las posibilidades de aplicar (y eventualmente aplique) el patrón Abstract Factory para que el sistema compuesto cree los diferentes productos.

**Solución**

| **Pattern** | **Multiples bancos** | | |
|---|---|---|---|
| **based on** | Abstract Factory | | |
| **because** | ▪ Permite usar el mismo código para cualquier banco.<br>▪ Facilita agregar nuevos bancos. | | |
| **where** | Cliente | **is** | Sistema |
| | *FabricaAbstracta* | ***is*** | *FabricaAbstracta* |
| | *CrearProductoA()* | ***is*** | *CrearCuenta()* |
| | *CrearProductoB()* | ***is*** | *CrearCliente()* |
| | FabricaConcreta1 | **is** | BancoNacion |
| | CrearProductoA() | **is** | CrearCuenta() |
| | CrearProductoB() | **is** | CrearCliente() |
| | FabricaConcreta2 | **is** | BancoProvincia |
| | CrearProductoA() | **is** | CrearCuenta() |
| | CrearProductoB() | **is** | CrearCliente() |
| | *ProductoAbstractoA* | ***is*** | *Cuenta* |
| | ProductoA1 | **is** | CajaDeAhorros |
| | ProductoA2 | **is** | CuentaCorriente |
| | *ProductoAbstractoB* | ***is*** | *Cliente* |
| | ProductoB1 | **is** | PersonaFisica |
| | ProductoB2 | **is** | PersonaJuridica |
| **comments** | COMPLETAR. | | |

9. Continuando con el problema de la estación climatométrica, suponga que la empresa que la desarrolla tiene varios modelos diferentes con sensores de distinta precisión, robustez y precio. Utilice el patrón Bridge para diseñar un sistema que se auto configure en tiempo de ejecución de acuerdo a la plataforma donde está ejecutando. Muestre código de ejemplo de cómo se implementaría este aspecto de la aplicación.

**Solución**

| **Pattern** | **HAL** | | |
|---|---|---|---|
| **based on** | Bridge | | |
| **because** | ▪ El programa funciona de la misma manera con cualquier tipo de sensor.<br>▪ Abstrae las operaciones de bajo nivel de cada sensor. | | |
| **where** | Cliente | **is** | Sensores |
| | Abstraccion | **is** | Sensor |
| | Operacion() | **is** | Sensar() |
| | imp | **is** | Marca |
| | *Implementador* | ***is*** | *Marca* |
| | *OperacionImp()* | ***is*** | *SensarImp()* |
| | ImplementadorConcretoA | **is** | SensorACME |
| | OperacionImp() | **is** | SensarImp() |
| | ImplementadorConcretoB | **is** | SensorEMCA |
| | OperacionImp() | **is** | SensarImp() |
| **comments** | COMPLETAR. | | |

10. En el problema 8 se trató el problema de la creación de objetos que representan los diversos productos de los bancos que fue adquiriendo nuestro banco. Ahora toca el turno de aplicar el patrón Wrapper para implementar resoluciones del BCRA referidas a las transacciones que pueden realizar los clientes de cualquier banco sobre sus cuentas corrientes. Algunas de las resoluciones del BCRA son las siguientes:

    **A-2156.** No se podrán realizar más de dos extracciones semanales de las cajas de ahorro en pesos.

    **A-3401.** Si el titular es una persona jurídica cada depósito de más de $10.000 deberá ser informado al BCRA en un archivo mensual descrito en la C-BCRA-B-1190.

    **A-3436.** Los plazos fijos de más de $1.000.000 deberán ser informados al BCRA en un archivo mensual descrito en la C-BCRA-B-1284.

**B-3441.** Los bancos quedan autorizado a no permitir extracciones de más de $10.000 por mes de cualquier cuenta, si así lo consideran apropiado. Esta medida podrá ser aplicada y revocada por cada entidad si lo comunica a sus cliente con una antelación superior a los 30 días.

**A-3211.** Si un cliente posee valores por más de $10.000.000 deberá ser informado mensualmente al BCRA.

**Solución** COMPLETAR.

11. Considere el problema relativo al sistema de archivos de Linux visto en la práctica de diseño. El VFS da una representación jerárquica a los archivos y directorios como usted sabe. Utilice el patrón Composite para representar los directorios y archivos.

**Solución** COMPLETAR.

12. Aplique el patrón Iterator para recorrer el sistema de archivos de diferentes formas.

**Solución** COMPLETAR.

13. Suponga que el VFS provee un módulo que dado un código de error emitido por alguna de sus subrutinas (llamadas al sistema) retorna el mensaje correspondiente. Aplique el patrón Bridge para para poder configurar el VFS para retornar los errores en diferentes idiomas (internacionalización).

**Solución** COMPLETAR.

14. Una utilidad provista por Linux es *du* que retorna el espacio en disco utilizado por un directorio dado. Estas utilidades se proveen a nivel de usuario. Analice la posibilidad de aplicar (y eventualmente aplique) el patrón Visitor para implementar este tipo de servicios (*find*, etc.).

**Solución** COMPLETAR.

15. Otra utilidad provista por Linux es el comando *fscheck* que permite reparar el sistema de archivos ante algunos daños. Analice la posibilidad de aplicar (y eventualmente aplique) el patrón Strategy para implementar diferentes estrategias de reparación. Haga lo mismo con el patrón Visitor. Notar que se requiere recorrer el sistema de archivos completo.

**Solución** COMPLETAR.

16. Suponga que se desean implementar diferentes modelos de control de acceso a los archivos y directorios del sistema de archivos de Linux. Analice la posibilidad de aplicar (y eventualmente aplique) el patrón Wrapper para implementar esos modelos de seguridad.

**Solución** COMPLETAR.