A continuación encontrará cuatro casos que deben resolverse bajo la misma modalidad de ejercicios que se han practicado en clase.

Cada equipo de trabajo deberá:

* Anotar sus nombres y números de carné en la tabla que se presenta abajo.
* Para cada caso deben resolver los puntos (1..5) y completar la tabla asociada a cada caso, incorporándo redacción formal donde se solicita la justificación y proporcionando imágenes de modelos y código **ABSOLUTAMENTE LEGIBLES**.
* Este documento debe renombrarse bajo el nombre IIExamenGrupoNNMM y entregarlo en formato **PDF (sin excepción)**.
* El equipo creará un proyecto de programación Java (preferiblemente usando IDE Netbeans) denominado **IIExamenGrupoNNMM (**donde N es 40 si es de la sede CTLSJ o 02 si es de la Sede Cartago, y MM es el número de grupo asignado de trabajo). Construyan en el proyecto cuatro carpetas, cada una con el nombre **CasoX\_Patron** (donde X es el número de caso a resolver y Patron es el nombre del patrón con el que proponen la solución). Ejemplo **Caso7\_Patron.**

Al tec Digital deberá subir un archivo comprimido llamado **IIExamen\_GrupoNNMM** siguiendo la misma nomenclatura del proyecto programado que contiene el proyecto Java y el enunciado del examen en formato **PDF**.

## Caso 1

Los objetos FileInputStream típicamente representan archivos de texto accedidos en orden **secuencial**, byte a byte. Con FileInputStream, se puede acceder a un byte, varios bytes o el archivo completo.

Si se revisa el API IO de Java, la cantidad de objetos especializados en el manejo de archivos de texto es enorme. Hay objetos BufferedInputStream que incluye en su funcionamiento un buffer de datos para un mejor rendimiento y añade una funcionalidad de leer una línea, *readLine()*, para leer un renglón a la vez.

Adicionalmente, existe otro tipo de objeto LineNumberInputStream que añade la funcionalidad de contar la cantidad de líneas contenidas en el archivo y hasta puede indicar cual es el número de línea actualmente accedido en un momento del tiempo.

**Lo que debe hacer en este caso:**

1. ¿Cuál patrón considera usted que fue utilizado para poder proveer todos estos tipos distintos de manejadores de archivos de texto? **(2 puntos)**

El patrón utilizado fue el decorador.

1. Justifique formalmente la selección del patrón para este caso. Aporte del diagrama original de UML correspondiente al patrón. **(3 puntos)**

Además de la misma documentación del API del paquete Java.io, es notorio debido a que en su funcionalidad lo que se hace es “pasarle” el mismo objeto a otro que lo “contiene”. Este comportamiento de contener otro objeto que proviene de la misma interfaz es el mismo comportamiento que propone el patrón Decorator.

1. Siguiendo el modelo del patrón, construya detalladamente el modelo de UML que da solución a este planteamiento. Añada al modelo la posibilidad de ofrecer además un tipo de Stream asociado al manejador de archivos de texto que transforme el contenido del archivo en minúsculas ! Para el manejador de archivos que transforma a minúsculas, llame al objeto LCIStream, esto para no interferir con las propias del API si lo programa en Java.  **(5 puntos)**

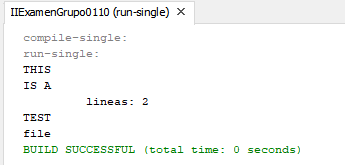
**Ver la tabla mas adelante**

1. Programe el funcionamiento del modelo del patrón seleccionado según su propuesta. Para efectos de la programación del caso, construya para representar las clases FileInputStream, BufferedInputStream y LineNumberInputStream bajo los nombres FIStream, BIStream y LNIStream respectivamente con las funcionalidades necesarias para realizar las tareas de leer el contenido de un archivo de texto según lo realiza el API de Java de acuerdo a lo expuesto. **(10 puntos)**

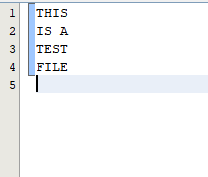
**Revisar el código adjunto**

1. Evidencias del código producido de los elementos significativos del patrón y su programa de prueba, así como screenshots de funcionamiento. **(5 puntos)**

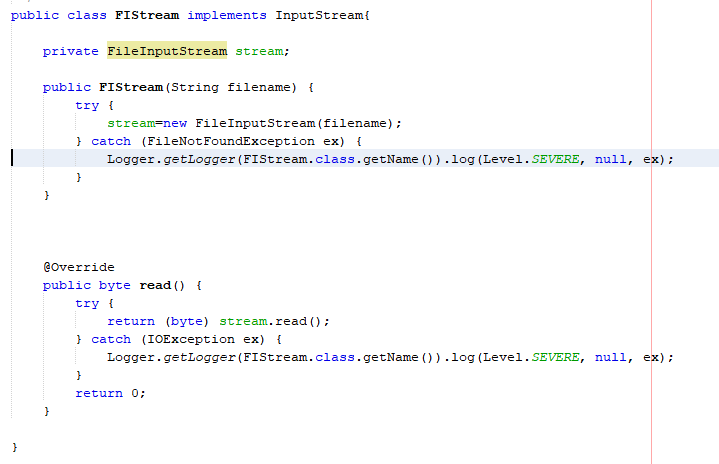
Corrida de prueba:

****

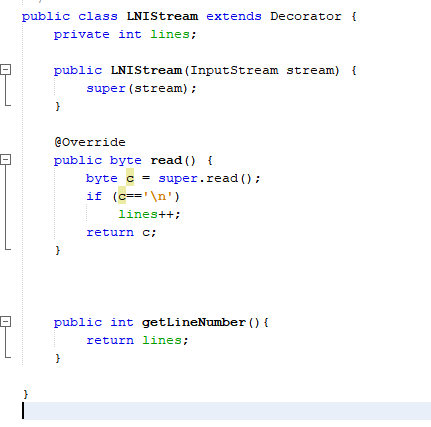
Archivo de prueba:

”

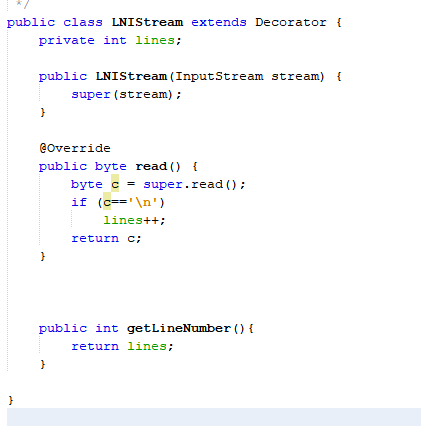
FIStream:



LNIStream:



BIStream:



|  |  |  |
| --- | --- | --- |
| **Respuesta Caso 1** | **Patrón a utilizar** | Decorator |
| **Tipo de Patrón** | Estructural |
| Justificación de uso del patrón | El patron Decorator se utiliza para agregarle responsabilidades dinámicamente a un objeto, que es lo que se desea hacer con el FileInputStream. Además, utilizar este patrón nos permite poder crear diferentes decoradores para un tipo de input stream (en este caso para el file input stream). | |
| Diagrama de UML asociado |  | |
| Modelo en UML que soluciona el problema | En el código los nombres son remplazados utilizando el formato solicitado por la profesora. | |
| Evidencias de código significativo en la implementación del patrón | Ver arriba | |
| Screeshoots de funcionamiento | Ver arriba | |

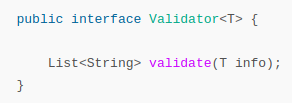
## Caso 2

En el desarrollo de un proyecto de curso, los estudiantes deberán manejar el acceso de los usuarios al mismo a través de un mecanismo que permita la autenticación del usuario a través de sus credenciales (login y contraseña). Además, se debe y permitir el registro de nuevos usuarios que forman parte de un equipo de trabajo y que requerirán acceder al sistema, por lo que el perfil del nuevo usuario debe completarse con información relevante, como su nombre con dos apellidos, fecha de nacimiento, datos de contacto como dirección física, correo electrónico, número del móvil como mínimo, eventualmente podrían agregarse otros datos a la sección de contactos como código postal, cuenta de alguna red social, entre otros. Los datos de contacto deben tener el formato adecuado y por supuesto no se debe permitir el registro de usuarios menores de edad. Los dos apellidos del usuario son requeridos.

Como es evidente, se requerirá validar algunos de los campos que forman parte del perfil del usuario al momento de registro y además, posteriormente se deberá realizar la validación de las credenciales, sin mencionar que en el proyecto como tal se requiere realizar validaciones que van de acuerdo a la lógica de negocio de la aplicación en desarrollo.

Por esta razón, al momento de la conceptualización del proyecto, se decidió definir una interface que permita estandarizar el llamado a mecanismos de validación independientemente del elemento a revisar, de modo que cuando se requiera una clase en particular, pueda implementar este mecanismo de la forma que le corresponda, según su contexto por medio de clases especializadas que lleve a cabo esta tarea.

Esta interface establece que el método devolverá una lista de hileras que contiene los errores que pudieron ser detectados en el proceso de validación. La interfaz se ha definido de manera totalmente genérica dentro del proyecto, a fin de que pueda ser utilizada con cualquier elemento que requiera implementar mecanismos de validación. Se muestra a continuación la sintaxis de la interface definida:



Se desea que la página o pantalla de registro de usuarios sea sometida al proceso de revisión de las distintas validaciones al momento de intentar registrar un nuevo usuario de modo que si se obtienen errores de distinta índole en el proceso sean desplegados en la pantalla y no se permita el registro para que el usuario corrija.

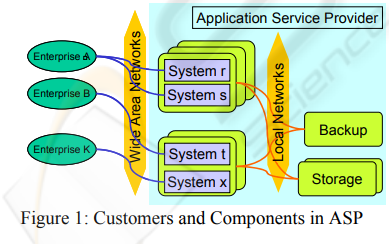
**Lo que debe hacer en este caso:**

1. ¿Cuál patrón cree usted que el equipo de trabajo utilizó para aportar flexibilidad y uniformidad al llevar a cabo las validaciones de información? **(2 puntos)**
2. Justifique formalmente la selección del patrón para este caso. Aporte del diagrama original de UML correspondiente al patrón. **(3 puntos)**
3. Proponga el diagrama de UML que pudo haber implementado el equipo de trabajo e incopore los elementos necesarios para ajustarse al requerimiento técnico que debe cumplir el formulario de registro de nuevos usuarios al sistema? **Justifique su respuesta**.  **(5 puntos)**
4. Programe el funcionamiento del modelo del patrón seleccionado según su propuesta. Desarrolle una pequeña interfaz de usuario que permita el registro de un nuevo perfil de usuario de modo que al momento de intentar registrar un usuario, se solicite al patrón llevar a cabo las validaciones correspondientes mostrando la lista de errores que podrían generarse o bien permitiendo el registro exitoso del nuevo usuario. Recuerde que una validación final debe ser que no se permite la duplicidad de registros. **(10 puntos)**

|  |  |  |
| --- | --- | --- |
| **Respuesta Caso 2** | **Patrón a utilizar** | ***Coloque aquí el nombre del patrón*** |
| **Tipo de Patrón** | ***Coloque aquí el tipo de patrón seleccionado*** |
| Justificación de uso del patrón |  | |
| Diagrama de UML asociado |  | |
| Modelo en UML que soluciona el problema |  | |
| Evidencias de código significativo en la implementación del patrón |  | |
| Screeshoots de funcionamiento |  | |

Caso 3

Application Service Providing o ASP es un modelo de negocio que da soporte a medianas empresas con software empresarial complejo y altamente integrado. Este modelo implementa una técnica y una infraestructura organizativa que garantiza un alto grado de disponibilidad del sistema y la seguridad de los datos. Varios clientes comparten el uso de la infraestructura central y cuentan con un mecanismo de configuración para acceder a sus datos y sus diversas funcionalidades que han sido contratadas.



La administración técnica de ASP debería ser soportado por un software de monitoreo que abarque una gran gama de componentes de software y hardware, esto porque un ASP normalmente es un sistema distribuido que tiene múltiples servidores, las aplicaciones son basadas en una arquitectura multicapa, tiene una gran exigencia de disponibilidad dado que múltiples usuarios pueden solicitar múltiples recursos en simultáneo, la apariencia de la aplicación puede variar dependiendo del cliente que ingrese, esto es, la misma aplicación puede tener distintas apariencias para ser manipulada por usuarios distintos, entre otros.

A menudo los monitores para componentes específicos son basados en técnicas no orientada a objetos, por ejemplo, parsing de archivos de log o accesos a información interna de ejecución que es llevada a cabo por los system-call de los sistemas operativos.

Entonces en ocasiones un monitor específico para un ASP requiere por ejemplo tener acceso a logs de estado que le sean provistos por un “FileMonitor”, y en otras necesita accesar a la tabla de procesos de un sistema operativo que pueden ser obtenidas por un “ProcessMonitor”.

Por otro lado podría requerirse distintos tipos de monitores, uno que esté monitoreando las bases de datos en la zona de almacenamiento (ORCLMonitor, DBaseMonitor), y en ocasiones se deberá conectar otro que rastree el ASP directamente (ASPMonitor).

**Lo que debe hacer en este caso:**

1. ¿Cuál sería el patrón que se puede utilizar para implementar esta estrategia de monitoreo asociada a este tipo de aplicación como es el ASP? **(2 puntos)**
2. Justifique formalmente la selección del patrón para este caso. Aporte del diagrama original de UML correspondiente al patrón. **(3 puntos)**
3. Proponga el diagrama de UML que implementa la flexibilidad de poder brindar al ASP el uso de monitores distintos que puedan dedicarse a recuperar información de comportamiento de los componentes de hardware y software involucrados en el ASP? **Justifique su respuesta**.  **(5 puntos)**
4. Programe el funcionamiento del modelo del patrón seleccionado según su propuesta. Suponga una instancia de un ASP con un configuración particular que debe llevar a cabo en un momento determinado operaciones de monitoreo de comportamiento de accesos a la base de datos y ver los archivos de log, y en otras ocasiones debe analizar el comportamiento de atención de procesos por parte del sistema operativo que le da soporte a la aplicación. **(10 puntos)**

|  |  |  |
| --- | --- | --- |
| **Respuesta Caso 3** | **Patrón a utilizar** | ***Coloque aquí el nombre del patrón*** |
| **Tipo de Patrón** | ***Coloque aquí el tipo de patrón seleccionado*** |
| Justificación de uso del patrón |  | |
| Diagrama de UML asociado |  | |
| Modelo en UML que soluciona el problema |  | |
| Evidencias de código significativo en la implementación del patrón |  | |
| Screeshoots de funcionamiento |  | |

Caso 4

Suponga una casa de producción de contenido que contrata agentes vendedores para escribir contenido para la organización.

Para cada proyecto asignado a un proveedor, el departamento de RRHH proporciona al vendedor un contrato con una serie de cláusulas que contienen términos y condiciones y además un acuerdo de confidencialidad que debe aceptar antes de comenzar a trabajar.

El contenido de los acuerdos sigue siendo el mismo para todos los proveedores y un empleado de Recursos Humanos sólo debe completar el nombre del proveedor antes de enviar un acuerdo al proveedor.

Tanto el formato del contrato como del acuerdo de confidencialidad se encuentran en una base de datos remota distintas que deben ser accedida para solicitar el envío un nuevo contrato para un nuevo vendedor. El vendedor no será contratado si no ha firmado ambos documentos-

El contrato debe ser firmado tanto por el encargado de RRHH y el vendedor.

**Lo que debe hacer en este caso:**

1. ¿Cuál sería el patrón que se puede utilizar para obtener ambos documentos y contratar al vendedor? **(2 puntos)**
2. Justifique formalmente la selección del patrón para este caso. Aporte del diagrama original de UML correspondiente al patrón. **(3 puntos)**
3. Proponga el diagrama de UML que implementa la solicitud por parte del encargado de RRHH de los documentos que se requieren para la contratación d de un nuevo vendedor. **Justifique su respuesta**.  **(5 puntos)**
4. Programe el funcionamiento del modelo del patrón seleccionado según su propuesta. Recuerde que cada documento debe ser obtenido de repositorios (o bases de datos distintas) para que ambos puedan ser completados por las partes, y ambos documentos deben estar firmados por ambos (encargado de RRHH y vendedor) para que se finiquite la contratación. **(10 puntos)**

|  |  |  |
| --- | --- | --- |
| **Respuesta Caso 4** | **Patrón a utilizar** | ***Coloque aquí el nombre del patrón*** |
| **Tipo de Patrón** | ***Coloque aquí el tipo de patrón seleccionado*** |
| Justificación de uso del patrón |  | |
| Diagrama de UML asociado |  | |
| Modelo en UML que soluciona el problema |  | |
| Evidencias de cóS13 digo significativo en la implementación del patrón |  | |
| Screeshoots de funcionamiento |  | |

**FIN DE LA PRUEBA**