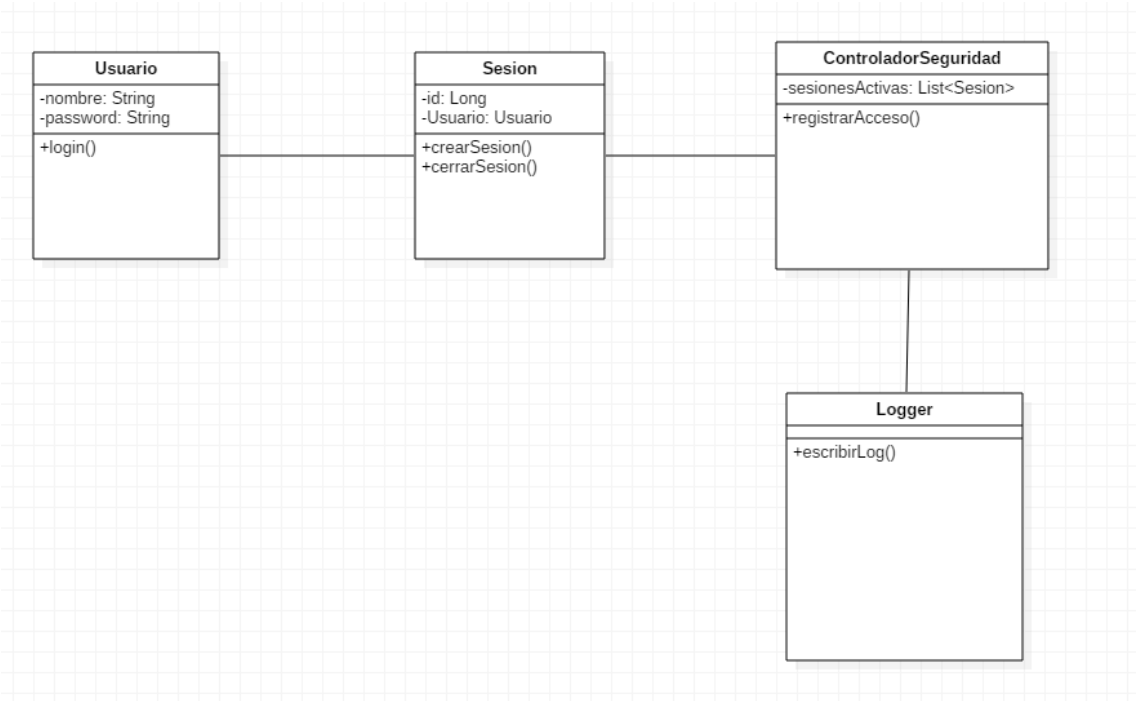
Ingeniería de Software II
Patrones de Diseño
Practica Integradora

En esta guía práctica estudiaremos algunos patrones de diseño y aplicaremos los conceptos vistos utilizando diagramas de clases en UML.

Ejercicio 1:

Considere el siguiente diagrama de UML, el cual modela un módulo de seguridad de una aplicación:



Dada ésta implementación se necesita:

- Se debe poder crear sólo una instancia de la clase ***ControladorSeguridad*** ya que existe un único objeto encargado de realizar dicha tarea.
- Cada vez que se registre el acceso de un usuario, la clase ***Logger*** debe escribir en un archivo el suceso, con el fin de registrar dicha información.

En base a esto se pide:

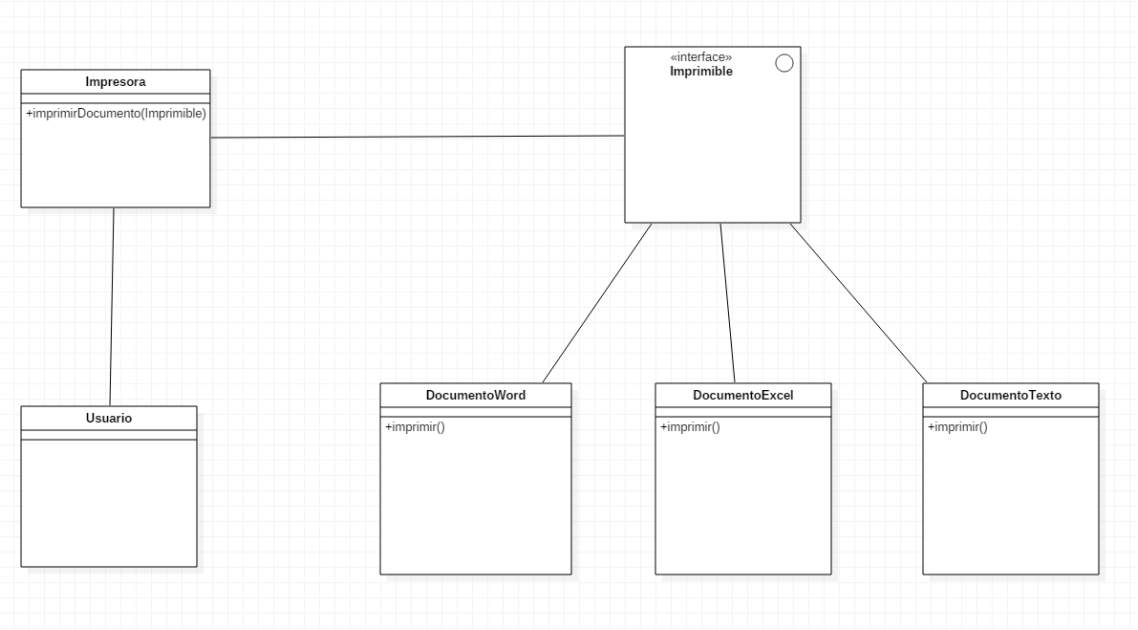
- Indicar que patrón / patrones se pueden utilizar para cada caso y justificar su uso.
- Reescribir el diagrama de UML con la solución implementada.

Ejercicio 2:

Dado el siguiente Diagrama de UML sobre el módulo de impresión en un sistema, se ha observado que la lógica para crear nuevos tipos de documentos es muy compleja y además por otro lado es el usuario el que tiene que especificar cuál es el documento que imprime, por lo que un cambio dentro de la clase DocumentoExcel modifica directamente todo el aplicativo, por lo que se necesita reducir el acoplamiento entre las clases y ocultar su comportamiento.

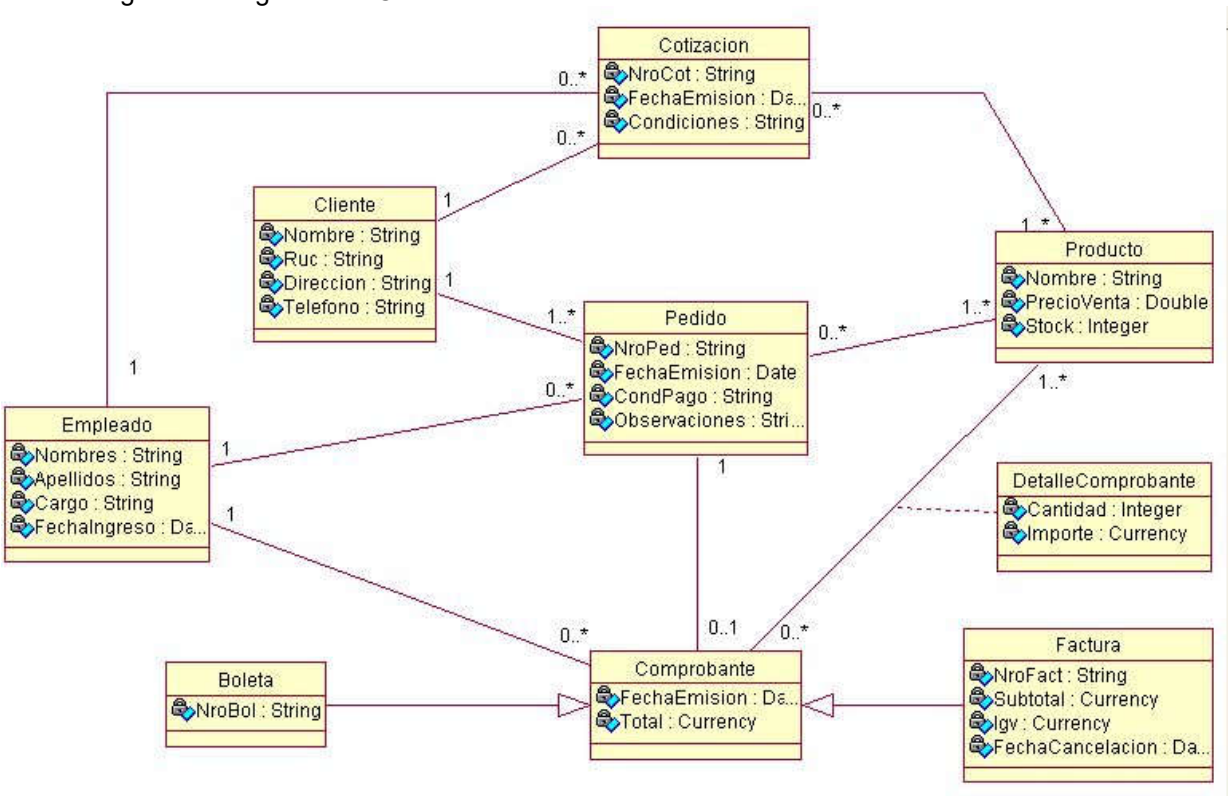
En base a esto se pide:

- Indicar que patrón / patrones se pueden utilizar para cada caso y justificar su uso.
- Reescribir el diagrama de UML con la solución implementada.



Ejercicio 3:

Dado el siguiente Diagrama de Clase



1) Se quiere implementar una lista de productos en oferta que se actualice diariamente. Considere para esto crear una clase llamada GestorProductos, la cual contenga métodos para gestionar productos, pero simplemente pueda crearse una sola instancia de dicha clase.
2) Se solicita implementar dos formas distintas de pedido: una de manera directa, otro mediante licitación, cada uno con particularidades distintas. Se debe tener en cuenta en el diseño la posibilidad de incorporar a futuro otras formas de pedido.
3) Se desea evitar el acomplamiento entre las clases Cliente y Empleado con respecto a la clase Pedido, puesto que un cambio en la segunda clase repercute en las primeras, se desea buscar una simple para brindar acceso a los métodos de la clase, pero sin accederla directamente.

En base a esto se pide:

- Indicar que patrón / patrones se pueden utilizar para cada caso y justificar su uso.
- Reescribir el diagrama de UML con la solución implementada.