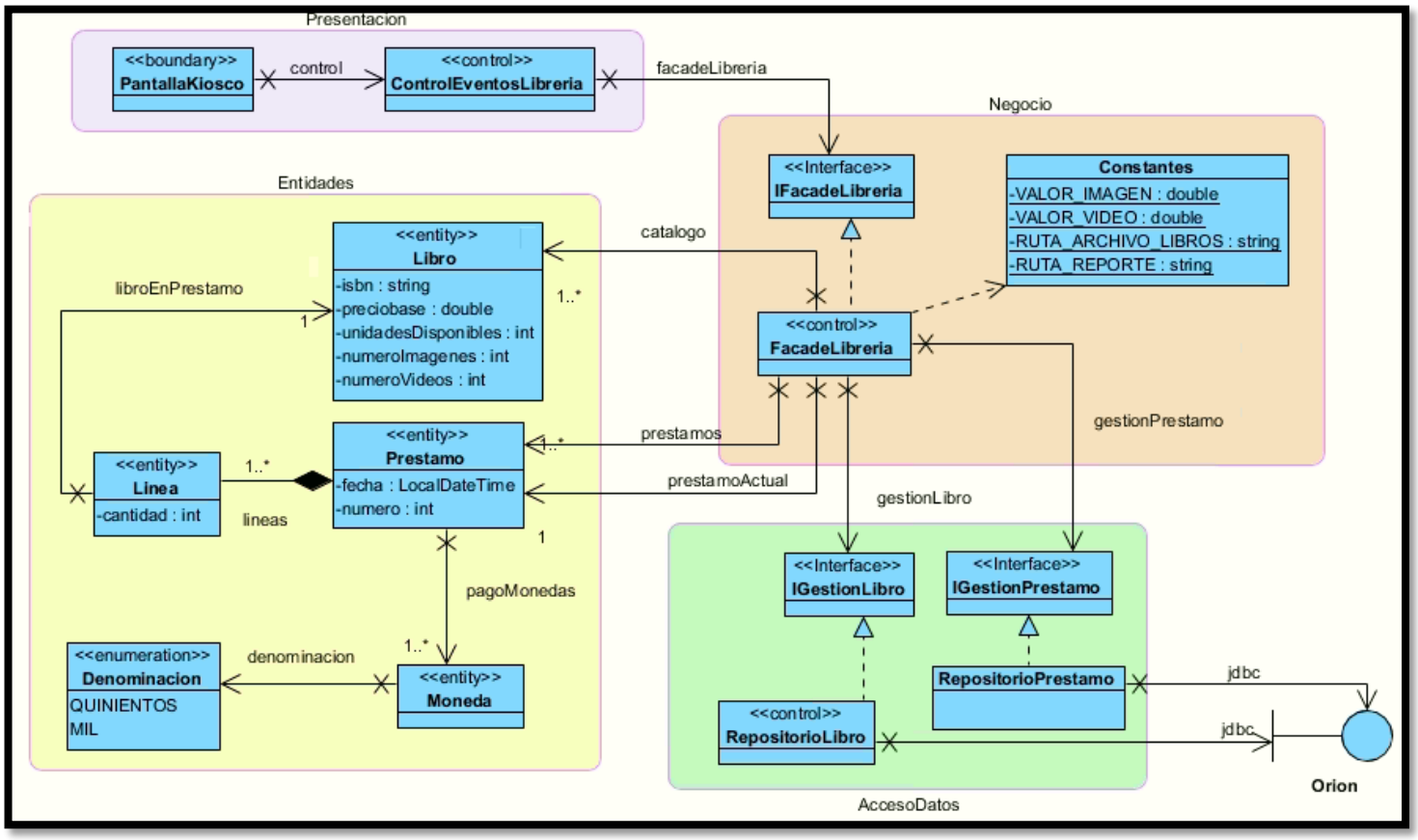
Kiosco de Libros.

- Se requiere hacer un programa orientado a objetos que funcionará en kioscos.
- Al finalizar el día los prestamos se envían a un servidor central y se limpian los préstamos en el kiosco.
- El kiosco tiene ahora una pantalla más amigable al usuario.

Cada capa UML es un .jar

- El siguiente es el diagrama de clases, observe que cada capa tiene un color diferente, cada capa es un proyecto diferente de java, esto es, un jar para cada capa.



- El diagrama de clases anterior no muestra todos los detalles ni métodos sólo se muestran los elementos relevantes.
- Las colecciones pueden manejarse MAP o LIST
- Las clases mostradas en el diagrama se corresponden con tablas en la base de datos.

Para este proyecto se solicita implementar las siguientes funcionalidades en la clase 'Kiosco'

Interfaz Gráfica de Usuario

El siguiente es el aspecto básico de la pantalla, pero usted puede enriquecer los elementos visuales sin sacrificar la funcionalidad solicitada.

El proceso de préstamo se resume de la siguiente manera:

1. **[20]** Al iniciar el día se debe:
    a. crear la colección de libros llamada 'catalogo' (método en el controlador 'IGestionLibro' que crea la lista de libros), cree el método 'CargarLibros()' en iGestionLibro, cuya funcionalidad es leer los libros desde la tabla de Libros y devolver una Lista de Libros para que sea asignada al 'catalogo'
    b. La clase Libreria debe invocar en su constructor el método anterior

2. **[10]** Crear Préstamo: Inicialmente la máquina crea un nuevo préstamo y se queda esperando por la introducción de monedas. (método 'crearNuevoPrestamo' de la clase 'Libreria' que no recibe parámetros, retorna booleano indicando si se pudo crear el préstamo)
    a. Este préstamo se maneja en la relación 'prestamos'
    b. El último préstamo pasa a ser manejado con la relación 'prestamoActual'
    c. Se toma la fecha y hora del sistema (use LocalDate)
    d. El número del préstamo no se puede repetir
    e. No se puede crear un nuevo préstamo sino existen unidades disponibles de ningún libro
        i Retorna falso
    f. Se debe desplegar un mensaje en pantalla indicando si se pudo o no crear el préstamo.

Notas:

- Este método se invoca al presionar el botón 'Nuevo Prestamo', al oprimir el botón se deben limpiar todas las etiquetas dejarlas en cero, limpiar la tabla de líneas y la caja de cantidad ponerla en cero.

Existe una clase 'DtoResumen' (usted debe crear esta nueva clase) que se utilizara para devolver datos desde todos los métodos que vienen a continuación, cada método debe diligenciar los atributos correspondientes.

El Dto tiene:

- Un atributo 'mensaje' de tipo cadena con mensajes de error
- La colección de objetos de Líneas conteniendo:
    - Objeto Libro
    - cantidad
    - El valor total del libro (precio)
    - subtotal de la línea
- Un atributo de tipo booleano que indica si se pudo agregar la línea al préstamo
- El total de todo el préstamo
- El saldo de las monedas ingresadas
- La cantidad de vueltos del préstamo actual
- Agregue todos los demás atributos que requiera para devolver y poder refrescar la GUI.

NOTA: se debe persistir el préstamo en las tablas usando el 'RepositorioPrestamo'

3. **[20]** Agregar Línea: El usuario va agregando líneas al préstamo

Método 'agregarLinea' que recibe un objeto libro del catálogo y una cantidad de libros para crear una nueva línea; retorna un 'DtoResumen' que contiene:

- Un atributo 'mensaje' de tipo cadena con mensajes de error
- La colección de objetos de Líneas conteniendo:
    - Objeto Libro
    - cantidad
    - El valor total del libro (precio)
    - subtotal de la línea
- La anterior colección tiene la nueva línea creada
- Un atributo de tipo booleano que indica si se pudo agregar la línea al préstamo
- El total de todo el préstamo

El código del método 'agregarLinea' tiene que:

a. Verificar Libro en Catalogo (método privado)
    i. El sistema verifica que el libro que llega como parámetro se encuentra en el catalogo
    ii. Si el libro existe se vincula en la relación 'libroEnPrestamo'

iii. Si el libro no existe debe diligenciar el atributo 'mensaje' del 'DtoResumen' de retorno

b. Verificar Existencias Libro. (método privado)
   i. El sistema valida que la existencia del libro sea suficiente (atributo 'unidadesDisponibles' de la clase libro.
      1. Si no hay existencia debe diligenciar el atributo 'mensaje' del 'DtoResumen' de retorno
   ii. Si un libro ya existe en el préstamo o sea ya está en una 'línea' se acumula la cantidad existente con la solicitada.
c. Crear Linea (método privado)
   i. Crea la línea y la introduce en la lista de 'líneas' del préstamo actual
d. Calcula el valor del libro (método privado)
   i. Precio base + (número imágenes * valor imagen) + (numero de videos * valor video)
e. Calcula el subtotal de una línea (método privado)
   i. Multiplica el valor del libro (calculado en el método anterior) por la cantidad de libros de la línea
f. Calcula el total del préstamo (método privado)
   i. sumatoria de los subtotales de cada línea (calculados en el método anterior)
g. Crear el 'DtoResumen' que va a retornar
   i. Use los métodos ya implementados anteriormente.

Notas:

- Este método 'agregarLinea' se invoca cuando se presiona el botón 'Agregar Linea', para los parámetros de entrada del método: el libro se debe tomar del combo de 'Seleccionar Libro', la cantidad se toma de la caja de texto 'Cantidad'
- Se debe refrescar la GUI, esto es, refrescar la grilla y los totales del préstamo, use el 'DtoResumen' retornado por el método; se deben mostrar 'mensaje' de error si lo hay.
  - El objeto línea que devuelve el método debe ser vinculado a la grilla de libros del préstamo.

NOTA: se debe persistir la línea en las tablas y consultar las líneas desde la tabla para devolverlas a la lógica (usando el 'RepositorioPrestamo')

4. **[10]** Eliminar una línea del Préstamo

Método publico eliminarLinea recibe un objeto de tipo 'Linea' y retorna un 'DtoResumen' que contiene:

- Un atributo 'mensaje' de tipo cadena con mensajes de error
- La colección de objetos de Líneas conteniendo:
  - Objeto Libro
  - cantidad
  - El valor total del libro (precio)
  - subtotal de la línea
- La anterior colección sin la línea borrada
- Un atributo de tipo booleano que indica si se pudo eliminar la línea del préstamo
- El total de todo el préstamo

El código de 'eliminarLinea' tiene que:

a. Verificar Línea (método privado)

i. Si el objeto de tipo Linea que llega esta nulo se diligencia el 'mensaje' del 'DtoResumen'

b. Buscar la línea y quitarla de la colección de líneas del préstamo actual
    i. Si no se encuentra la línea se diligencia el 'mensaje' del 'DtoResumen'
c. Crear el 'DtoResumen' que va a retornar.
    i. Reutilice los métodos ya implementados en el controlador.

Notas:

- Para Eliminar se debe seleccionar en la grilla de la GUI la línea a Eliminar y presionar el botón 'Eliminar Linea'
- Se debe refrescar la GUI, esto es, refrescar la grilla y los totales del préstamo, use el 'DtoResumen' retornado por el método; se deben mostrar 'mensaje' de error si lo hay.

NOTA: se debe persistir la eliminación de la línea en las tablas y consultar las líneas desde la tabla para devolverlas a la lógica (usando el 'RepositorioPrestamo').

5. **[10]** Introducir Monedas

Método publico introducirMoneda recibe un enumerado de tipo 'Denominacion' y una cantidad de moneda de la denominación; y retorna un 'DtoResumen' con el atributo de 'saldo de monedas ingresadas' ya diligenciado con el total de monedas de 'pagoMonedas' del préstamo.

El código de 'introducirMoneda' tiene que:

a. Validar que exista el enumerado que llega como parámetro
    i. Si no se encuentra se diligencia el 'mensaje' del 'DtoResumen'
b. Crear una nueva 'Moneda', vinculando el enumerado que llega como parámetro
    i. Se asume la cantidad como 1 una moneda
c. Agregar la moneda creada a la colección 'pagoMonedas' del préstamo
d. Crear el 'DtoResumen' que va a retornar.

Notas:

- Para Agregar una moneda se debe digitar el número de monedas, la denominacion y presionar el botón 'Agregar Moneda'
- Se debe refrescar la GUI, esto es, refrescar la etiqueta de la pantalla cuyo nombre es 'saldo disponible de monedas ingresadas', use el 'DtoResumen' retornado por el método; se deben mostrar 'mensaje' de error si lo hay.

NOTA: se debe persistir la moneda introducida en las tablas y consultar las monedas desde la tabla para devolverlas a la lógica (usando el 'RepositorioPrestamo').

6. **[20]** Terminar Préstamo

Metodo público 'terminarPrestamo' no recibe parámetros y retorna un 'DtoResumen' con el atributo valor de los vueltos diligenciado.

El código de ‘terminarPrestamo’ tiene que:

a. Verificar Saldo (método privado)
    i. Si el saldo disponible (total de monedas introducidas relación ‘pagoMonedas’) no es inferior al valor total del libro seleccionado entonces: se dispensan los libros.
        1. En caso contrario diligenciar el mensaje del ‘DtoResumen’
b. Actualizar Existencias (método privado)
    i. Se actualizan las existencias del libro restando en unidades disponibles la cantidad de libros de cada línea del préstamo.
e. Devolver Saldo (método privado)
    i. Si hay saldo restante la máquina lo devuelve
    ii. Se debe retornar los vueltos (un double)

f. Crear el ‘DtoResumen’ que va a retornar.

Notas:

- Para invocar este método se debe presionar el botón ‘Terminar Prestamo’
- Se debe refrescar la GUI, esto es, refrescar la etiqueta de la pantalla cuyo nombre es ‘vueltos’, use el ‘DtoResumen’ retornado por el método; se deben mostrar ‘mensaje’ de error si lo hay.

NOTA: se debe persistir la terminación de la línea en las tablas y consultar las líneas desde la tabla para devolverlas a la lógica (usando el ‘RepositorioPrestamo’).

7. **[20]** Consultar Préstamo

Método público en Librería que recibe un número de préstamo; el método busca el número del préstamo y retorna null o un DTO con todos los datos que se necesitan para llenar los elementos visuales de la pantalla referidos al préstamo encontrado: fecha, numero, líneas, total del préstamo.

Agregue en la interfaz gráfica una caja de texto en donde se pueda introducir el número de un préstamo a consultar. Si lo considera necesario redistribuya la pantalla para hacerla más clara.

NOTA: se debe consultar el préstamo desde la tabla para devolverlas a la lógica (usando el ‘RepositorioPrestamo’).

8. **[40]** Cree un aplicativo MVC usando JavaFX que permita probar las funcionalidades
    a. Se debe crea una pantalla similar a la dada en esta entrega
    b. Se debe crear un controlador de eventos que debe usar el controlador ‘ILibreria’

<u>Se deben mostrar en pantalla los mensajes que retornen los diferentes métodos.</u>

Fechas

Entrega

La entrega se hace a través del buzón de transferencia digital del sistema de aprendizaje UVirtual (BlackBoard) a más tardar el día **28 de Mayo de 2020 hasta las 11 a.m.**

Entregas posteriores a esta fecha no serán tenidas en cuenta.

Si realiza varias entregas en el sistema, sólo se tendrá en cuenta la primera entrega.

Grupos

La entrega se realizará en grupos de trabajo. Los grupos no podrán cambiar su conformación y desde el comienzo dichos grupos estarán identificados plenamente.

Entregables

- Archivo .zip con el código fuente de las clases
- Archivo .jar con el código ejecutable del programa

Observaciones

- Se reducirán puntos por malas prácticas de programación: ○ código "quemado". Por ejemplo, usar valores constantes en donde no se deba.
- El diagrama de clases y la implementación deben ser concordantes.
- Si no hay código, la nota corresponderá a 0.0
- SUSTENTACION INDIVIDUAL: en caso de no ser exitosa la sustentación, se reconocerá el 20% del total obtenido.
- Si no se entrega el archivo .jar no se calificará la entrega.
- Cada clase deberá ser debidamente documentada y deberá aparecer el nombre completo de los autores

Restricciones

- La lógica y la presentación deben estar separadas.
- Se deben leer datos en la presentación y procesarlos en la lógica de negocio
    - ○ Toda la creación y procesamiento de objetos debe realizarse en la lógica pasando los parámetros necesarios desde la pantalla
- Para las colecciones no use arreglos []
- Solo se calificarán puntos que tengan la presentación y la lógica ***completas***, esto es, que tengan la GUI (interfaz gráfica) y la lógica de negocio completas.