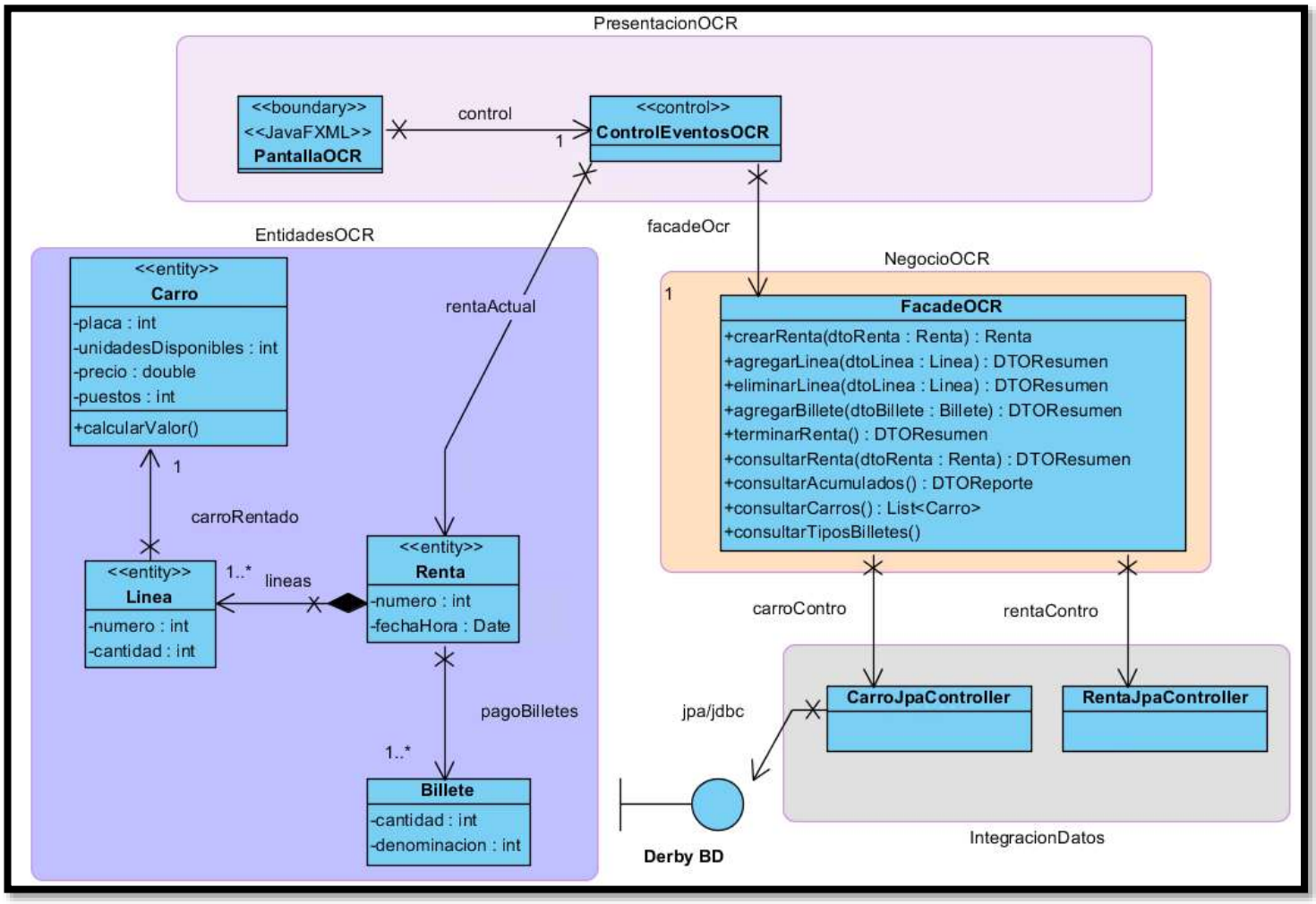
- Se requiere hacer un programa orientado a objetos basado en tablas relacionales y con arquitectura stand alone.
- En un kiosco hemos puesto una pantalla más amigable al usuario para que se puedan rentar los carros para grandes empresas que rentan a la vez varios carros para sus empleados. En el kiosco se pueden rentar y pagar las rentas.
- El siguiente es el diagrama de clases, observe que cada capa tiene un color diferente, cada capa es un paquete dentro de un único un jar.



- 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 'Entidades' mostradas en el diagrama se corresponden con tablas en la base de datos.**

*** : Todas las operaciones de base de datos que esten marcadas en el enunciado con el * deben quedar en las clases XXXXXJpaController y deben ser invocadas desde FacadeOCR.**

Notas del diagrama:

- En el diagrama se muestran solo dos XXXXXJpaController pero pueden ser todos los que usted considere conveniente.

- Existe una sola FacadeOCR pero si lo considera conveniente puede crear otras facades.

- En el diagrama no se dan de forma explícita los métodos de los XXXXXJpaController ya que usted debe proveerlos. Debe pasar parámetros DTO y devolver objetos DTO.

    - Si utiliza NetBeans el wizard le ayuda a crear la parte CRUD de los métodos

Qué se pide:

I. (25%) Modelo de tablas a partir del modelo de clases

a. **Se debe entregar el modelo de tablas (se envía un VPP) el 12 de mayo 10 p.m.**

II. (75%) Código java para acceder las tablas del modelo (calificado sobre 140 puntos)

a. **Se debe entregar máximo el 4 de junio**

Notas para representar el modelo de datos a partir del modelo de clases (parte I de la entrega)

A. Se debe revisar la capa de Entidades OCR
B. Las relaciones de composición son relaciones identificantes.
C. En la memoria no se montan todos los datos; en las tablas están todos los datos.
D. El modelo de clases no siempre es bidireccional, en la base de datos siempre se debe establecer la bidireccionalidad
E. El modelo de clases solo establece cardinalidad respecto a los datos que se cargan en memoria; en las tablas se debe establecer la cardinalidad pensando en todos los datos.
F. En el modelo de clases no siempre se reflejan las relaciones muchos a muchos; debido a la navegabilidad y cardinalidad representada en memoria a los datos cargados.

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.

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

1. **[15]** Construir Respuesta Renta *(ESTE METODO DEBE REUTILIZARSE DESDE TODOS LOS METODOS QUE REQUIERAN RETORNAR EL DTORESUMEN)*

Método: Construir Respuesta Renta
Entrada: recibe un objeto 'renta'
Retorno: un 'DtoResumen' descrito a continuación

Para la mayoría de los métodos de FacadeOCR se debe retornar un objeto de la una clase 'DtoResumen' (usted debe crear esta nueva clase).

Para diligenciar los datos del DTOResumen debe ir a consultar en la base de datos.

El DtoResumen contiene:

- Un atributo 'mensaje' de tipo cadena con mensajes de error
  - Cuando este atributo no sea nulo se asume que se presentó un error y se debe mostrar en la pantalla.
- La colección de objetos de Líneas conteniendo:
  - Objeto Carro

    - cantidad de carros rentada
    - El valor total de los carros (precio del carro * cantidad)
    - subtotal de la línea
- El total de toda la renta
- El saldo de los billetes ingresados
- La cantidad de vueltos de la renta

Agregue todos los demás atributos que requiera para devolver y poder refrescar la GUI.

Cada método referenciado en la entrega debe diligenciar los atributos correspondientes acorde a la disponibilidad de datos en cada punto del proceso.

2. **[5]** Consultar lista de Carros.

Método: consultarCarros
Entrada:
Retorno: una lista de carros habilitados para ser rentados.

Con la lista se debe llenar el dropdown de Carros para la opción 'Seleccione Carro Por número de Puestos'. Se debe mostrar la placa y la cantidad de puestos del carro. Al crear una nueva renta se debe desplegar la lista de carros.

3. **[5]** Consultar lista de denominaciones de billetes.

Método: consultarTiposBillete
Entrada:
Retorno: una lista de las denominaciones de billetes que se manejan en el sistema.

Con la lista se debe llenar el dropdown de 'denominaciones'. Al crear una nueva renta se debe desplegar la lista de denominaciones.

El proceso de renta se resume de la siguiente manera:

4. **[10]** Crear Renta. Inicialmente se crea una nueva renta y se queda esperando por la selección de carros.

Método: CrearRenta
Entrada: recibe un DTO objeto Renta
Retorno: un 'DtoResumen' descrito anteriormente

Con los datos del retorno se refresca el modelo de objetos en memoria y se despliegan los mensajes de error.

Esta funcionalidad se invoca cuando se presiona el botón 'Nueva Renta'; se deben limpiar todas las etiquetas dejarlas en cero, limpiar la tabla de líneas y la caja de cantidad ponerla en cero.

Método 'crearRenta' debe:

a. El dto que debe enviarse como parámetro debe tener

i la fecha y hora del sistema (use LocalDate)

b. Reglas de negocio:
   i No se puede crear una nueva renta sino existen unidades disponibles de ningún carro
      1 Se debe crear una consulta para determinar esta condición
      2 Se retorna nulo si no hay carros
c. Se inserta una nueva renta en la tabla respectiva.
d. Se retorna un objeto renta con los datos que quedan en la tabla
e. Se debe refrescar la GUI, esto es, refrescar los datos de la renta, use el 'DtoResumen' retornado por el método; se deben mostrar 'mensaje' de error si lo hay.
f. Esta renta queda vinculada en la relación 'rentaActual'
g. Se debe invocar el método 'consultarCarros' para rellenar el dropdown de carros.
h. Se debe invocar el método 'consultarTiposBillete' para rellenar el dropdown de 'denominacion'

5. **[15]** Agregar Línea: El usuario va agregando líneas a la renta actual (relación 'rentaActual')

Método: AgregarLinea
Entrada: recibe un DTO 'línea' que debe agregarse a la 'rentaActual'
Retorno: un 'DtoResumen' descrito anteriormente

Con los datos del retorno se refresca el modelo de objetos en memoria y se despliegan los mensajes de error.

El código del método 'agregarLinea' tiene que:

a. Verificar Carro en Catalogo
   i. El sistema verifica que el carro que llega como parámetro se encuentra en el catálogo; para ello debe realizar una consulta en la tabla de carros. *
   ii. Si el carro no existe debe diligenciar el atributo 'mensaje' del 'DtoResumen' de retorno
b. Verificar Existencias Carro.
   i. El sistema valida que las existencias del carro sean suficientes (atributo 'unidadesDisponibles' de la clase carro. Se debe realizar una consulta a la tabla carros. *
      1. Si no hay existencia debe diligenciar el atributo 'mensaje' del 'DtoResumen' de retorno
   ii. Si un carro ya existe en la renta o sea ya está en una 'línea' se acumula la cantidad existente con la solicitada.
      - Se debe consultar si la renta ya tiene en una línea el carro que se pretende agregar *
      - Si ya existe una línea con ese carro se debe realizar una actualización en la línea de la renta en la tabla respectiva *
   i. Crear Línea
   i. Crea la línea en la tabla *
   ii. Calcula el valor del carro
   i. Precio base – total descuentos
      i Se debe consultar desde la base de datos el precio del carro *
      ii Si se rentaron más de 5 carros se debe descontar un 10%
      iii El 5% y el 10% deben crearse en una **tabla parámetros** y desde allí se deben consultar dichos valores *

iii. Calcula el subtotal de una línea

i. Se debe consultar desde la base de datos las líneas de la renta *

ii. Multiplica el valor de la renta del carro por la cantidad de carros rentados de la línea.

iv. Calcula el total de la renta

i. sumatoria de los subtotales de cada línea

v. Crear el 'DtoResumen' que va a retornar

i. Use los métodos ya implementados anteriormente.

Notas:

b. Este método 'agregarLinea' se invoca cuando se presiona el botón 'Agregar Linea', para los parámetros de entrada del método: el carro se debe tomar del combo de 'Seleccionar Carro', la cantidad se toma de la caja de texto 'Cantidad'

c. Se debe refrescar la GUI, esto es, refrescar la grilla y los totales de la renta, use el 'DtoResumen' retornado por el método; se deben mostrar 'mensaje' de error si lo hay.

- El objeto línea que se creó debe aparecer en la grilla de lineas de la renta.

6. **[10]** Eliminar una línea de la Renta

Método: EliminarLinea
Entrada: recibe un objeto 'linea' que debe quitarse de la 'rentaActual'
Retorno: un 'DtoResumen' descrito anteriormente

Con los datos del retorno se refresca el modelo de objetos en memoria y se despliegan los mensajes de error.

El código de 'eliminarLinea' tiene que:

a. Verificar Línea

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

b. Buscar (basados en el objeto línea que llega como parámetro) en la tabla la línea de la renta *

c. si se encuentra se debe realizar una eliminación de la fila de la tabla respectiva. *

i. Si no se encuentra la línea se diligencia el 'mensaje' del 'DtoResumen'

i. Crear el 'DtoResumen' que va a retornar.

i. Para crear este resumen reutilice los métodos ya implementados.

Notas:

d. Para Eliminar se debe seleccionar en la grilla de la GUI la línea a Eliminar y presionar el botón 'Eliminar Línea'

e. Se debe refrescar la GUI, esto es, refrescar la grilla y los totales de la renta, use el 'DtoResumen' retornado por el método; se deben mostrar 'mensaje' de error si lo hay.

- El objeto línea que se creó ya no debe aparecer en la grilla de líneas de la renta.

7. **[10]** Introducir Billetes

Método: AgregarBillete

Entrada: recibe un DTO objeto 'billete' que debe agregarse a 'rentaActual'
Retorno: un 'DtoResumen' descrito anteriormente

Con los datos del retorno se refresca el modelo de objetos en memoria y se despliegan los mensajes de error.

El código de 'introducirBillete' tiene que:

a. Validar que exista la denominación del billete que llega como parámetro *
    i. Si no se encuentra se diligencia el 'mensaje' del 'DtoResumen'
b. Insertar un nuevo 'Billete' para la renta *
c. Crear el 'DtoResumen' que va a retornar.

Notas:

f. Para Agregar un billete se debe digitar el número de billetes, la denominacion y presionar el botón 'Agregar Billete'
g. 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.

8. **[10]** Terminar Renta

Método: terminarRenta; este método termina la 'rentaActual'
Entrada:
Retorno: un 'DtoResumen' descrito anteriormente

Con los datos del retorno se refresca el modelo de objetos en memoria y se despliegan los mensajes de error.

El código de 'terminarRenta' tiene que:

a. Verificar Saldo
    i. se debe consultar el valor total de la renta (estos métodos ya debería tenerlos y reutilizarlos en este punto)
    ii.se debe consultar el valor total de billetes de la renta *
    ii.Si el saldo disponible (total de billetes introducidos) no es inferior al valor total de la renta entonces se puede terminar la renta.
        1. En caso contrario diligenciar el mensaje del 'DtoResumen'
b. Actualizar Existencias
    i. Se actualizan las existencias del carro restando en unidades disponibles la cantidad de carros de cada línea de la renta. *
e. Devolver Saldo (método privado)
    i. Si hay saldo restante de debe devolver
    ii. Se debe retornar los vueltos (un double) : La cantidad de vueltos de la renta en el resumenDTO

f. Crear el 'DtoResumen' que va a retornar.

Notas:

h. Para invocar este método se debe presionar el botón 'Terminar Renta'
i. 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.

9. **[10]** Consultar Renta

Método: consultarRenta
Entrada: recibe un DTO objeto 'renta' que contiene el número de la renta a consultar
Retorno: un 'DtoResumen' descrito anteriormente

Con los datos del retorno se refresca el modelo de objetos en memoria y se despliegan los mensajes de error.

El código de 'consultarRenta' tiene que:

i.buscar la renta por número *
ii. retornar null o un DTO con todos los datos que se necesitan para llenar los elementos visuales de la pantalla.

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

10. **[10]** Consultar Acumulados

Método: consultarAcumulados
Entrada:
Retorno: una lista de objetos DTO (cree un nuevo dto)

Se debe retornar un listado que contenga año, mes, cantidad total de carros rentados.

Agregue en la interfaz un botón para invocar este método; crear una tabla en pantalla para mostrar resultados.

11. **[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 'FacadeOCR'

➔ **De forma alternativa si no crea la pantalla puede entregar un Junit o una clase main donde se evidencia que la lógica de negocio y la lógica de integración funciona. Esto no le otorgará puntos, pero permitirá calificarle la parte de las lógicas de negocio e integración.**

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
- Nombre de dos esquemas/usuarios de bases de datos donde se pueda verificar el código

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.
- Cada clase deberá tener 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 []