

Sea una empresa que vende computadoras. La misma vende sus productos tanto a clientes finales como a revendedores. Los clientes finales pueden comprar productos individuales, que tienen un precio o paquetes cerrados de productos que tienen un precio y no necesariamente es la suma de los precios de los productos que incluye. Por ejemplo: un gabinete, una fuente, un microprocesador. Otro paquete puede ser: un gabinete, uno mother, un microp., memoria, teclado y Mouse.

Además el negocio brinda promociones. Una promoción incluye productos y paquetes y tiene un descuento. Solo si la compra incluye todos los productos de la promoción, se le aplica el descuento. Las promociones varían periódicamente. El cliente compra y el negocio factura, teniendo en cuenta las promociones.

Por último, los clientes tienen distintas formas de pago, cada forma con su % de descuento: efectivo 10%, cheque 3%, tarjeta de debito 7%, tarjeta de credito 5%. El descuento de las tarjetas de credito y debito solo se aplica si el cliente gastó más de 1000 pesos.

Los revendedores solo pueden comprar productos en cantidad mayorista (esto es, para cada producto se determina la cantidad mínima que un mayorista puede comprar) y poseen un precio mayorista.

1. Realice el diagrama de clase del diseño.

2 Realice el /los diagramas de secuencia para calcular el valor de una factura por un paquete que se paga con tarjeta de débito.

3 Implemente el mensaje Empresa>>facturar: **productos** a Cliente: unCliente y todos los mensajes necesarios. Considere que **productos** puede incluir productos simples o paquetes. Recuerde tener en cuenta las promociones para hacer el descuento correspondiente en la facturación.

4 Muestre en el workspace como se crea el producto Motherboard Asus con valor U\$200. Muestre como se crea un paquete formado por esta Motherboard, y un micro Athlon 64 de valor U\$250. El valor de todo el paquete es U\$400.

Orientación a Objetos I Final Nov. 2017

A) Un spooler de impresión es un objeto que administra una colección de impresoras y los documentos que se imprimirán en ellas. Recibe documentos y los envía a una impresora que esté libre (se asume que siempre hay al menos una impresora libre). Hay dos formatos de documentos: PDF y PostScript, cada documento se imprime de forma distinta. El cuerpo de los documentos es un String. Los documentos PDF se imprimen en mayúsculas y los PostScript en minúscula. El tamaño de un documento está dado por el largo de su cuerpo (el String) + 30 bytes en el PDF y 50 en el PostScript.

A.1.- Realice el diagrama de clases que modele la aplicación anterior.

A.2. realice un diagrama de secuencia para el mensaje Spooler>>imprimir: unDocumento

A.3.- Implemente los mensajes enunciados en A.1

A.4.- Implemente los mensajes de inicialización del spooler y de asignación de impresoras al spooler.

B) Al spooler anterior se le agrega una cola de impresión. Cuando se envía un documento se busca una impresora libre. Si no existe ninguna, se agrega el documento en la cola. Luego, cuando una impresora se libera (Asuma que al liberarse una impresora, al spooler le llega en forma automática el mensaje #impresoraLiberada: unImpresora), se le da un documento de la cola.

B.1.- Modifique/Extienda el diagrama de clases con una cola de impresión.

B.2.- Implemente los mensajes necesarios para encolar documentos si no existe una impresora libre y para que extraiga un documento, cuando se libera una impresora.

B.3- Al spooler se le puede enviar los siguientes mensajes, implementelos:

a) #documentosOrdenadosPorTamaño "Retorna una colección de los documentos en la cola, ordenados por tamaño, de mayor a menor"

b) #tamañoTotalDeEncolados "Retorna el tamaño total de la cola de impresión, es decir, la sumatoria de los tamaños de los documentos encolados"

C) Existe **un nuevo tipo de spooler** que funciona exactamente como el anterior (B) salvo que encola los documentos según la prioridad asociada al documento, eso significa que cuando se libere una impresora se imprimirá el documento de mayor prioridad. A prioridades iguales se devuelve según el orden de llegada.

C.1- Modifique el diagrama de clase de manera que contemple esta nueva situación.

C.2.- Implemente los mensajes necesarios para encolar documentos si no existe una impresora libre y para que extraiga un documento, cuando se libera una impresora.

C.3- Implemente los mensajes de inicialización del spooler.

D) Si para resolver el inciso C evaluó varias alternativas de diseño, describa cuál fue la alternativa descartada y explique cuáles fueron las razones que lo llevaron a tomar esa decisión.

Ejercicio 1: VentasOnLine

VentasOnLine es empresa de ventas online de productos que quiere desarrollar un servicio de seguimiento de pedidos. Los pedidos representan la compra realizada por un comprador on-line, que tiene identificado al usuario, la lista de productos comprados y dirección de envío. El pedido es el resultado de finalizar la compra.

El servicio de seguimiento de pedido se encarga de mostrar en que estado esta el pedido. Los estados posibles son:

- En preparación
- Enviado
- En destino
- Retirado

El pasaje de un estado a otro es secuencial, como se enunciaron arriba. Cuando se crea el pedido automáticamente queda en estado "en preparación".

De un estado a otro se pasa enviando el mensaje #avanzar. Sin embargo, dependiendo del estado actual es lo que sucede:

- Si está en estado "en preparación" para pasar a estado "enviado" hay que elegir la compañía de correo y asociarle el número de seguimiento al pedido. La elección de la compañía de correo simularla.
- • Si esta en estado "enviado" para pasar a estado "en destino" hay que enviar un email al comprador para avisar que ya llegó el pedido a destino (simular).
- Si está en estado "en destino" para pasar a estado "retirado", se debe asentar en el sistema datos de la persona que retiró el pedido.

Platee una solución para la situación descripta arriba explotando las cualidades del modelo de objetos.

1. Diseñe un diagrama de clases
2. Implemente el método #crearPedido.
3. Describa cual/cuales son las clases responsables de implementar el método #avanzar. Justificar
4. Diseñe un diagrama de secuencia para #avanzar un pedido de estado "enviado" a "en destino"
5. Implemente el método #avanzar en la clase/las clases que haya discutido en el punto 2.
6. Implemente el método #mostrarEstado

Ejercicio 2: Peaje

En una ruta existen estaciones de peajes. La estación está formada por varias cabinas. Los vehículos (motos, autos y camiones) llegan al peaje y se suman a la cola de una cabina.

Las cabinas pueden ser de pago exacto, con vuelto y de camiones. Las cabinas de pago exacto sólo admiten a quienes disponen del pago justo. Las cabinas de camiones sólo admiten camiones. Estas restricciones no impiden que un camión ingrese por la cabina de con vuelto o pago exacto (si lo tiene), o que un vehículo con pago exacto pase por la cabina de con vuelto. Los valores que deben abonar, dependen del vehículo y de la cabina.

	Pago exacto	Con vuelto	Camion
Moto	1	1.5	
Auto	2	2.5	
Camion	16	8.5	8

El vehículo (suponga que no hay conductor) tiene el dinero para pagar. El dinero puede ser exacto o no. Las cabinas además llevan el registro de los vehículos que pasaron como así también del importe abonado.

Parte 1:

1. Realice el/los diagramas de secuencia para asignar un auto a una cabina y cobrarle peaje.
2. Realice un diagrama de clases (indicando herencia, relaciones de conocimiento, cardinalidad y responsabilidades) a partir del problema planteado para resolver el problema de asignar a un auto a una cabina de peajes con menos vehículos en su cola de espera.
3. Defina los mensajes necesarios para crear e inicializar una estación de peajes, con cabinas de peajes.
4. Implemente los siguientes mensajes

```
EstacionDePeaje>> asignarCabinaAUnVehiculo: unVehiculo
```

```
"encola al vehículo en alguna cabina"
```

```
~Cabina>>admiteVehiculo: unVehiculo
```

```
"retorna true si la cabina admite a ese vehiculo"
```

```
Cabina>>precioDePeajeParaUnVehiculo: unVehiculo
```

```
"retorna la cantidad a pagar por unVehiculo en el receptor"
```

```
Cabina>>cobrarAVehiculo: unVehiculo
```

```
"cobra el monto que unVehiculo debe pagar"
```

```
EstacionDePeaje>>cabinaQueRecaudoLaMayorCantidadDeDinero
```

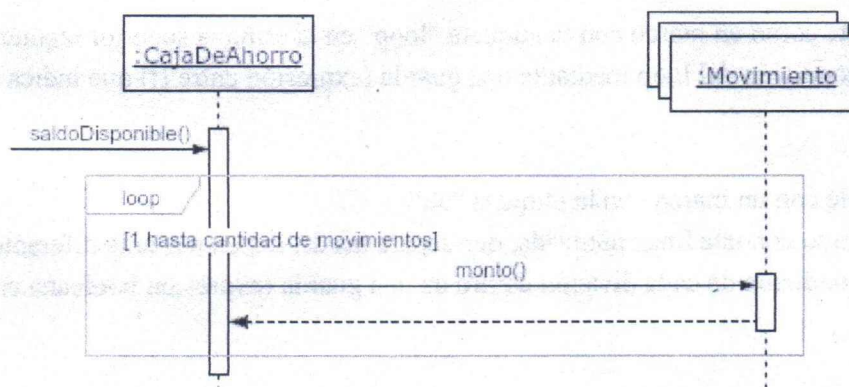
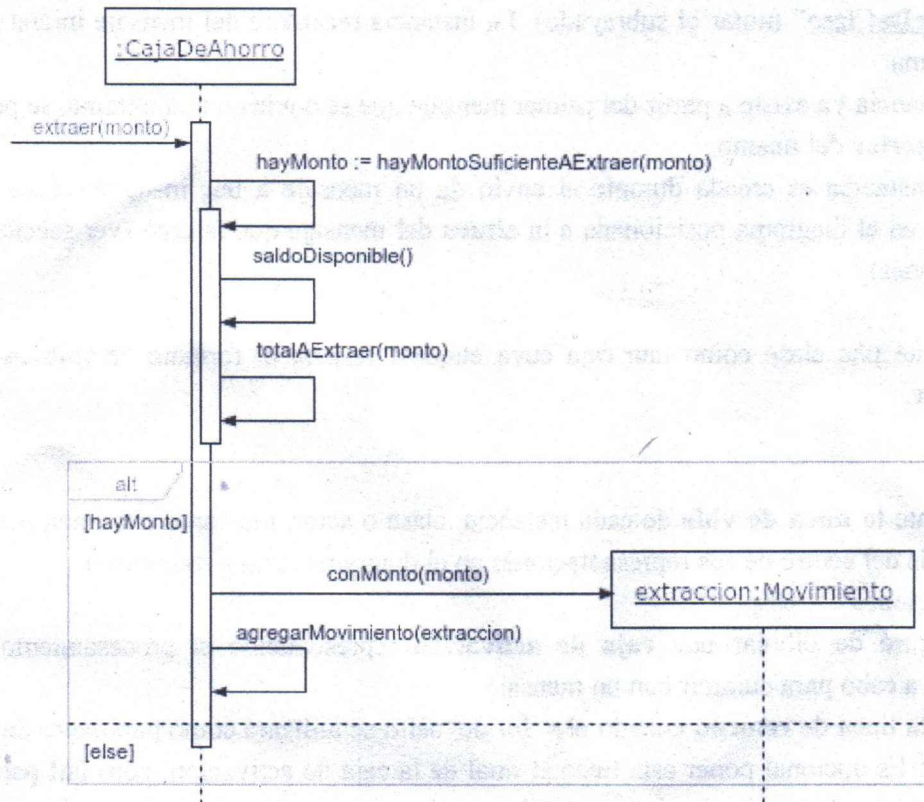
```
"retorna aquella cabina con mayor recaudación acumulada"
```

Parte 2: Indique para cada una de las siguientes afirmaciones si son verdaderas o falsas. Considere el polimorfismo para todos los mensajes públicos de un objeto.

1. Dos objetos polimórficos siempre son instancias de la misma clase. **F**
2. Dos clases polimórficas deben siempre pertenecer a la misma jerarquía. **F**
3. Si dos objetos son polimórficos es posible intercambiarlos sin que se produzcan errores del tipo Mensaje no entendido. **V**

Recomendaciones de sintaxis UML: Diagrama de secuencia

Ejemplos:



El diagrama

Describe siempre un **escenario de uso**, no es necesario que represente todas las posibilidades

Actores

- ☐ Representé **actores** (participante externo al sistema), solo si es necesario y de una de estas dos maneras:

1. mediante un monigote con una etiqueta.

2. mediante una caja con una etiqueta en la zona superior, y con un estereotipo «actor» en un segundo renglón.

- ☐ Expresé el nombre de un actor respetado el formato “**Nombre De Actor**”.

Instancias

- ☐ Representé una instancia como una caja cuya etiqueta respeta el formato “**nombreDeInstancia :NombreDeClase**” (notar el subrayado). La instancia receptora del mensaje inicial generalmente es anónima.
- ☐ Si la instancia ya existe a partir del primer mensaje que se envía en el diagrama, se posiciona en la **zona superior** del mismo.
- ☐ Si una instancia es creada durante el envío de un mensaje a una instancia/clase, su caja debe aparecer en el diagrama posicionada a la **altura** del mensaje que la creó (ver sección Mensajes y activaciones).

Clases

- ☐ Representé una clase como una caja cuya etiqueta respeta el formato “**NombreDeClase**”, sin subrayar.

Líneas de vida

- ☐ Representé la **línea de vida** de cada instancia, clase o actor, mediante una línea punteada que se desprende del centro de sus representaciones en el diagrama (caja o monigote).

Mensajes y activaciones

- ☐ Me aseguré de dibujar una **caja de activación** representando el procesamiento que se está llevando a cabo para cumplir con un mensaje.
- ☐ Indiqué la línea de **retorno** cuando el valor devuelto se utilizará como parámetro de otro mensaje posterior. Es opcional poner esta línea al final de la caja de activación, pero útil para marcar más claramente donde termina la ejecución del mensaje.

Loops

- ☐ Lo representé como un marco con la etiqueta “loop” en la esquina superior izquierda
- ☐ Indiqué la condición del loop mediante una guarda (expresión entre []) que indica el intervalo.

Alt

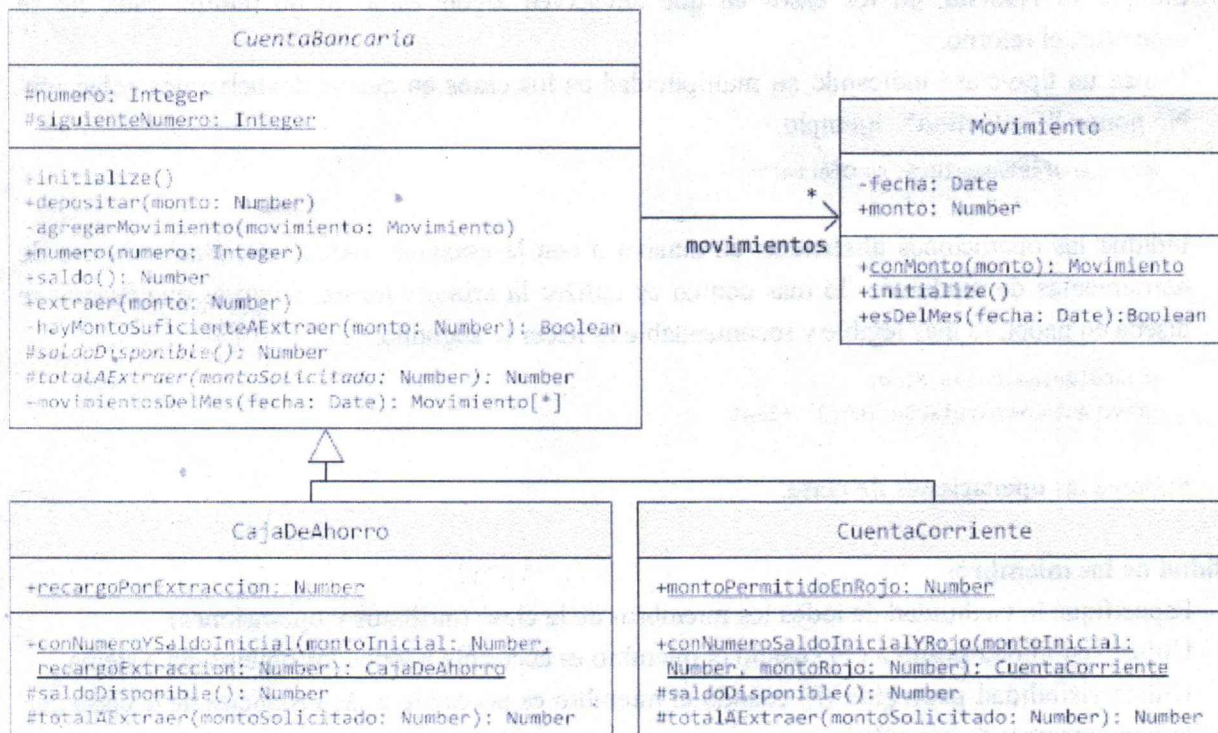
- ☐ Lo representé con un marco con la etiqueta “alt”
- ☐ Dividí el marco con una línea punteada, que separa las descripciones ante diferentes condiciones
- ☐ Indiqué la condición de cada división dentro de una guarda (expresión booleana entre []).

Recomendaciones de sintaxis UML: Diagrama de clases

Si bien es válido especificar una clase sólo con su nombre (sin atributos ni operaciones), los diseños trabajados en clase deben contar con los siguientes tres compartimentos:

NombreDeClase
atributos
operaciones

Por ejemplo:



Nombre de la clase

- Expresé su nombre en **CamelCase**, comenzando en mayúscula.
- Expresé su nombre en cursiva o con la notación «Abstract» si se trata de una clase abstracta. Haciendo uso de herramientas de modelado, lo más común es utilizar la primera forma, mientras que cuando se diseña en papel, lo más legible y recomendable es hacer lo segundo.

Atributos/Propiedades

- Expresé su nombre en **camelCase**, comenzando en minúscula.
- Subrayé los **atributos de clase**.
- Indiqué aquellos atributos que tiene que ser accedidos públicamente con **visibilidad pública (+)**, en lugar de especificar accessors (**getter** y **setter**).
- No incluí como **atributos** aquellos que ya están expresados como final de asociación con otras clases.
- Incluí el tipo de atributo.

Operaciones

- Expresé el nombre en **camelCase**, comenzando en minúscula .
- Puse () al final del nombre aunque no tenga parámetros.
- Indicé los tipos de parámetros.
- Expresé los **parámetros dentro de paréntesis separados por comas**. Ejemplo:

En Smalltalk:

```
dash: totalDistance spacing: spaceLength
```

En UML:

```
+dashSpacing(totalDistance: Integer, spaceLength: Integer)
```

- Indicé su **retorno**, en los casos en que devuelven algún valor. Si no retorna nada, no se especifica el retorno.
- Utilicé un tipo/clase indicando su multiplicidad en los casos en que se devuelve una colección. **No poner "Collection"**. Ejemplo:

```
+obtenerOfertasDelDia(): Oferta[*]
```

- Indicé las operaciones **abstractas** en cursiva o con la notación «Abstract». Haciendo uso de herramientas de modelado, lo más común es utilizar la primera forma, mientras que cuando se diseña en papel, lo más legible y recomendable es hacer lo segundo.

```
+calcularSueldo(): Float
```

```
«abstract» +calcularSueldo(): Float
```




- Subrayé las **operaciones de clase**.

Visibilidad de los miembros

- Especifiqué la visibilidad de todos los miembros de la clase (atributos y operaciones).
- Utilicé visibilidad **pública** (+) cuando el miembro es accesible a todos los objetos del sistema.
- Utilicé visibilidad **protegida** (#) cuando el miembro es accesible a las instancias de la clase que lo implementa y de sus subclases.
- Utilicé visibilidad **privada** (-) cuando el miembro es sólo accesible a las instancias de la clase que lo implementa.

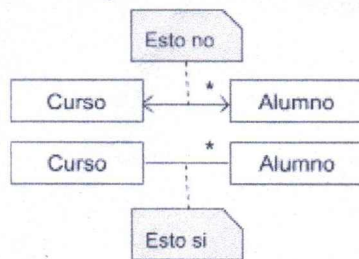
Asociaciones

- Indicé la:

Asociación	con	
Agregación	con	
Composición	con	

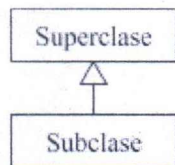
- Indicé el nombre del **rol** en cada extremo del destino navegable que tenga la asociación, opuesto a la clase origen.
- Indicé la multiplicidad en todos los casos, salvo que la misma sea "1".
- Indicé la dirección en las relaciones unidireccionales.

- **No** indiqué la dirección en las asociaciones bidireccionales (conocimiento mutuo).



Generalización

- Indiqué la relación de generalización con una flecha sin relleno, apuntando en dirección a la superclase.



- **No** incluí nombre ni multiplicidad a esta relación.