

1) **Ejercicio conceptual** (se recomienda leer todo el ejercicio antes de comenzar). Si se desea implementar un juego de ajedrez¹, algunas entidades de dominio podrían ser:

Tablero: el conjunto de las 64 celdas sobre las que se mueven las piezas.

Celda: cada una de las 64 celdas del tablero.

Pieza: cada una de las piezas móviles, con distintos movimientos permitidos (comportamientos) según el tipo de pieza que se trate.

Juego: cada partida de ajedrez, con sus jugadores, su tablero y su comportamiento.

Jugador: cada uno de los jugadores representados por su color (blanco o negro). Cuando juega la computadora, el jugador tiene un comportamiento inteligente; cuando juega el usuario, éste puede elegir cualquier celda disponible.

Basados en el siguiente repositorio: https://github.com/fiuba/algo3_chess/

Se requiere implementar el siguiente cambio de requerimiento:

Un jugador solo puede mover una ficha/pieza a un casillero cuya columna sea impar y cuando el número de fichas/piezas suyas en el tablero sea impar.

Se pide:

a. Confeccionar el diagrama de clases en UML que modela el requerimiento anterior. Utilice como base los diagramas provistos. Use nombres adecuados para todas las clases, métodos y asociaciones que defina. Incluya todos los métodos que le parezca necesarios en las clases, pero ninguno más. Los métodos que utilice en los puntos b, c y d, deben figurar en el diagrama.

a. Modelar en UML (diagrama de secuencia, con objetos y mensajes) el algoritmo completo para la determinación de si, en un determinado momento, una pieza come a una pieza contraria. No tiene que modelar nada que dependa del tipo de pieza: use correctamente la abstracción para evitarlo.

a. Escriba, por lo menos, una prueba automatizada positiva y una negativa (no triviales) necesarias para probar el comportamiento modelado en el diagrama de secuencia, usando SUnit. **Adaptar el código suministrado a los efectos de cumplir la nueva consigna.**

b. Escriba el código Smalltalk que haga funcionar las pruebas del punto c. **Adaptar el código suministrado a los efectos de cumplir la nueva consigna.**

2) Explique qué ventajas ofrece el tipeo estático sobre el dinámico. Hay autores que dicen que la falta de genericidad era un déficit en los lenguajes de tipeo estático. ¿Por qué?

¹ El ajedrez se juega de a dos jugadores sobre un tablero de 8x8. Cada jugador cuenta, al principio del juego, con 8 peones, 1 rey, 1 reina, 2 alfiles, 2 torres y 2 caballos. Los turnos son alternados, uno para cada jugador. Cada jugador está identificado con un color: blanco o negro. Cada pieza, según su tipo, tiene un conjunto de movimientos permitidos. Cuando una pieza se coloca en el lugar de una pieza contraria, se dice que la “come”, y la pieza comida se retira del tablero. Hay muchas más reglas que no hace falta definir aquí.

2do cuatrimestre 2020 - 1 de diciembre de 2020

Ver el código Smalltalk en las categorías:

Algo3Parcial

Algo3ParcialTests

1. Programar la clases y métodos necesarios para mejorar el código entregado, aplicando los conceptos de POO vistos en clase. No es necesario programar la clase Camioneta
2. Programar las pruebas unitarias que considere necesarias y colocarlas en la categoría ParcialAlgo3Tests, deberá haber una prueba positiva y una negativa.
3. Realizar un diagrama de clases de la solución planteada.
4. Realizar los diagramas de clases de una prueba positiva y una prueba negativa.

Contexto

El enunciado trata de modelar las restricciones que tienen los servicios de auxilio de automotores en cuanto a llevar pasajeros junto con el automotor que auxilian.

diagramas: si desean realizarlo a mano, pueden hacerlo en una hoja en blanco (no cuadriculada ni rayada) y sacar foto y entregar archivo estándar (.png , .pdf o .jpg). Si no pueden realizarlo con algún editor de UML o similar, entregando archivo estándar (.png , .pdf o .jpg).

Entregar los elementos del punto 1, 2 y 3 en el campus.

Para todos los casos las pruebas que forman parte del enunciado pueden ser modificadas según consideren necesario siempre y cuando se siga respetando la funcionalidad descrita en sus nombres.

Algoritmos y Programación III (75.07/95.02) – Curso 1 – Parcial – 09/05/2019

Nombre:	Padrón:
Corrigió:	Nota:

Cada ítem del examen se va a calificar con un valor entre 0 y 1. Para aprobar el examen se deberán cumplir todas las siguientes condiciones:

- En cada uno de los dos primeros ejercicios se debe tener al menos un puntaje de 0,6.
- Se debe sumar como mínimo el 60% del puntaje total obtenido mediante la ponderación de cada ejercicio (ver el coeficiente de ponderación en cada ejercicio).

1 - Ejercicio de modelado (se recomienda leer todo el ejercicio antes de comenzar). Se desea modelar parte de un sistema mediante el paradigma de objetos.

Ponderación: 60%

En la red social FIUBAgram existen los siguientes elementos:

- Usuarios: las personas anotadas en la red.
- Posts: publicaciones de los usuarios.
- Historias: como las publicaciones, pero con un tiempo de expiración (fijo)

Además los usuarios pueden relacionarse con otros usuarios como seguidores o amigos:

- seguidores: relación unilateral, un usuario es seguidor de otro.
- amigos: relación bilateral, dos usuarios son amigos entre sí.

Cada usuario tiene un nivel de visibilidad, que indica quienes pueden ver sus posts e historias:

- Pública: cualquier usuario puede ver sus publicaciones
- Privado: sólo pueden ser vistas por amigos
- semiPrivado: pueden ser vistas por seguidores y amigos.

Nota: la forma en que se establecen los vínculos entre usuarios está fuera del alcance de este ejercicio.

El alumno pensará y modelará:

- **Modelo 1.** Un diagrama de secuencia para obtener una colección de los 20 posts más recientes, que puede ver un usuario A de un usuario B.
- **Modelo 2.** Un diagrama de clases que muestre las clases que intervienen en el Modelo 1, y las relaciones entre ellas.
- **Modelo 3.** Dos pruebas automatizadas que modelen el caso anterior.

Para evaluar conceptos teóricos: (20% cada pregunta)

- 2 - La herencia múltiple permite:
- A. Que haya más de una clase descendiente de cada ancestro
 - B. Que los atributos de una clase sean punteros
 - C. Que un objeto sea instancia de más de una clase
 - D. Que un objeto contenga dentro a otro objeto
 - E. Que una clase tenga más de un ancestro
 - F. Violar el encapsulamiento
 - G. Todas las anteriores
 - H. Ninguna de las anteriores
- 3 - Una clase abstracta sirve para: (marque su respuesta en la hoja, sin justificar)
- A. Evitar el uso de herencia
 - B. Garantizar el polimorfismo
 - C. Generalizar estructura y comportamiento comunes de varias clases descendientes
 - D. Nada, es un mero concepto teórico
 - E. Todas las anteriores
 - F. Ninguna de las anteriores

Algoritmos y Programación III (75.07/95.02) – Curso 2 – Parcial – 09/05/2019

Nombre:	Padrón:
Corrigió:	Nota:

Cada ítem del examen se va a calificar con un valor entre 0 y 1. Para aprobar el examen se deberán cumplir todas las siguientes condiciones:

- En cada uno de los dos primeros ejercicios se debe tener al menos un puntaje de 0,6.
- Se debe sumar como mínimo el 60% del puntaje total obtenido mediante la ponderación de cada ejercicio (ver el coeficiente de ponderación en cada ejercicio).

1 - Ejercicio de modelado (se recomienda leer todo el ejercicio antes de comenzar). Se desea modelar parte de un sistema mediante el paradigma de objetos.

Ponderación: 60%

En la red social FIUBAgram existen los siguientes elementos:

- Usuarios: las personas anotadas en la red.
- Posts: publicaciones de los usuarios.
- Historias: como las publicaciones, pero con un tiempo de expiración (fijo)

Además los usuarios pueden a otros usuarios como seguidores o amigos:

- seguidores: relación unilateral, un usuario es seguidor de otro.
- amigos: relación bilateral, dos usuarios son amigos entre sí.

Cada usuario tiene un nivel de visibilidad, que indica quienes pueden ver sus posts e historias:

- Pública: cualquier usuario puede ver sus publicaciones
- Privado: sólo pueden ser vistas por amigos
- semiPrivado: pueden ser vistas por seguidores y amigos.

Nota: la forma en que se establecen los vínculos de entre usuarios está fuera del alcance de este ejercicio.

El alumno pensará y modelará:

- **Modelo 1.** Un diagrama de secuencia para obtener una colección de las 20 historias más recientes que puede ver un usuario de los usuarios a los que sigue.
- **Modelo 2.** Un diagrama de clases que muestre las clases que intervienen en el Modelo 1, y las relaciones entre ellas.
- **Modelo 3.** Dos pruebas automatizadas que modelen el caso anterior.

Para evaluar conceptos teóricos: (20% cada pregunta)

2 - Un atributo de clase (o estático) es:

- A. Otro nombre para indicar que un método es abstracto
- B. Un atributo cuyo valor se puede calcular en base a los valores de otros atributos
- C. Un atributo que mantiene siempre el mismo valor
- D. Un atributo que tiene el mismo valor para todos los objetos de la clase, aunque pueda cambiar
- E. Todas las anteriores
- F. Ninguna de las anteriores

3 - La redefinición:

- A. Debería hacerse de modo tal que la semántica del método en la clase descendiente sea totalmente diferente a la de la ancestro
- B. Es un buen sustituto de la herencia múltiple
- C. Permite introducir nuevos atributos en una clase descendiente
- D. Se da en los lenguajes sin clases cuando dos objetos tienen el mismo comportamiento
- E. Todas las anteriores
- F. Ninguna de las anteriores

Nombre:	Padrón:
Corrigió: Edson	Nota: 10 (diez)

Cada ítem del examen se va a calificar con un valor entre 0 y 1. Para aprobar el examen se deberán cumplir todas las siguientes condiciones:

- En el primer ejercicio se debe tener al menos un puntaje de 0,6.
- Se debe sumar como mínimo el 60% del puntaje total obtenido mediante la ponderación de cada ejercicio (ver el coeficiente de ponderación en cada ejercicio).

1 - Ejercicio de modelado (se recomienda leer todo el ejercicio antes de comenzar). Se desea modelar parte de un sistema mediante el paradigma de objetos.

Ponderación: 70%

En el sistema de alquiler de patinetas FIUBineta, tenemos los siguientes elementos::

- Usuarios: las personas anotadas en el sistema de alquiler.
- Patinetas: Los vehículos a Alquilar, poseen una ubicación y un estado de conservación (%).

Además los usuarios pueden ser de tipo:

- normalucho: el sistema les ofrece vehículos con estado de conservación de 30% a 70%.
- groso: el sistema les ofrece vehículos con estado de conservación de 60% a 100%.

Las patinetas pueden ser de tipo::

- FIUBineta estándar: pueden ser alquiladas por cualquiera.
- FIUBineta subestándar: sólo pueden ser alquiladas por usuarios normaluchos.

Suponer la existencia de una clase ubicación a quien puede enviarse un mensaje del siguiente tipo:

unaUbicacion obtenerDistanciaEnMetrosA: otraUbicacion.

El alumno pensará y modelará: (colocar la respuesta a cada uno de estos puntos en una hoja separada)

- **Modelo 1.** Un diagrama de secuencia para obtener las FIUBinetas en un radio de N metros de distancia que pueden ser alquilados por un usuario.
- **Modelo 2.** Un diagrama de clases que muestre las clases que intervienen en el Modelo 1, y las relaciones entre ellas.
- **Modelo 3.** Dos pruebas automatizadas que modelen el caso anterior.

Para evaluar conceptos teóricos: (10% cada pregunta) (contestar sobre esta hoja, no es necesario justificar)

2 - Al programar una clase, conviene prever todo lo que se vaya a necesitar, para incluirlo desde el principio

→ Verdadero ✓
→ Falso

3 - TDD, al prescribir escribir las pruebas antes, facilita aislar la especificación de la implementación

→ Verdadero ✓
→ Falso

4 - Una clase abstracta no puede tener métodos

→ Verdadero ✓
→ Falso

Cada ítem del examen se va a calificar con un valor entre 0 y 1. Para aprobar el examen se deberán cumplir todas las siguientes condiciones:

- En cada uno de los dos primeros ejercicios se debe tener al menos un puntaje de 0,6.
- Se debe sumar como mínimo el 60% del puntaje total obtenido mediante la ponderación de cada ejercicio (ver el coeficiente de ponderación en cada ejercicio).

1) **Ejercicio de modelado** (se recomienda leer todo el ejercicio antes de comenzar). Se desea modelar parte de un sistema mediante el paradigma de objetos.

Ponderación: 30%

En la materia Algoritmos 3 de FIUBA, el régimen de aprobación de la misma es tal que:

- Se requiere aprobar primero la "cursada" y luego el examen integrador.
- Para aprobar la cursada, el alumno debe obtener un promedio ponderado mayor o igual a 4, en el cual pesa un 5% la nota del TP0, un 35% la nota del TP1, un 30% la nota del TP2 y un 30% la nota del examen parcial.
- También es obligatorio aprobar con 4 o más el TP1, el TP2 y el parcial.
- Para aprobar el examen parcial, el alumno contará con 3 oportunidades durante la cursada. La primera vez que obtenga una nota igual o superior a 4 habrá aprobado el parcial.
- Para aprobar el examen integrador, el alumno tiene 1 año y medio, en el cual hay 15 fechas disponibles y puede presentarse hasta en 3 oportunidades. La primera vez que obtenga una nota igual o superior a 4 habrá aprobado el integrador y con ello la materia.
- Una vez aprobada la materia, se calcula la nota final redondeando el promedio ponderado de 60% para la nota de cursada y 40% para la última nota de integrador.

El alumno pensará y modelará:

- **Modelo 1.** Un diagrama de secuencia que muestre el cálculo de si un alumno aprobó o no la cursada, dadas todas sus notas. Ojo: tenga en cuenta que el alumno en cuestión podrá haber ya dado rendido todos los TPs y exámenes o no, y elija alguna forma de representación de un TP o examen no rendido.
- **Modelo 2.** Explique cómo modela un TP o examen no rendido.
- **Modelo 3.** Dos pruebas automatizadas que modelen el caso anterior.

(puede ayudarse con un diagrama de clases simplificado que ayude al corrector a entender mejor los anteriores)

Envia tus examenes a lawikifiuba@gmail.com

Para evaluar conceptos teóricos:

2) ¿Se pueden automatizar las pruebas de aceptación (UAT)? ¿Qué sentido tendría hacerlo? (máximo media carilla)
Ponderación: 20%

3) En un contexto de reutilización, cuando necesita que la clase nueva tenga la misma interfaz que la previa, usa delegación, dejando la herencia para los demás casos. ¿Es así o al revés? ¿Por qué? (máximo media carilla)

Ponderación: 20%

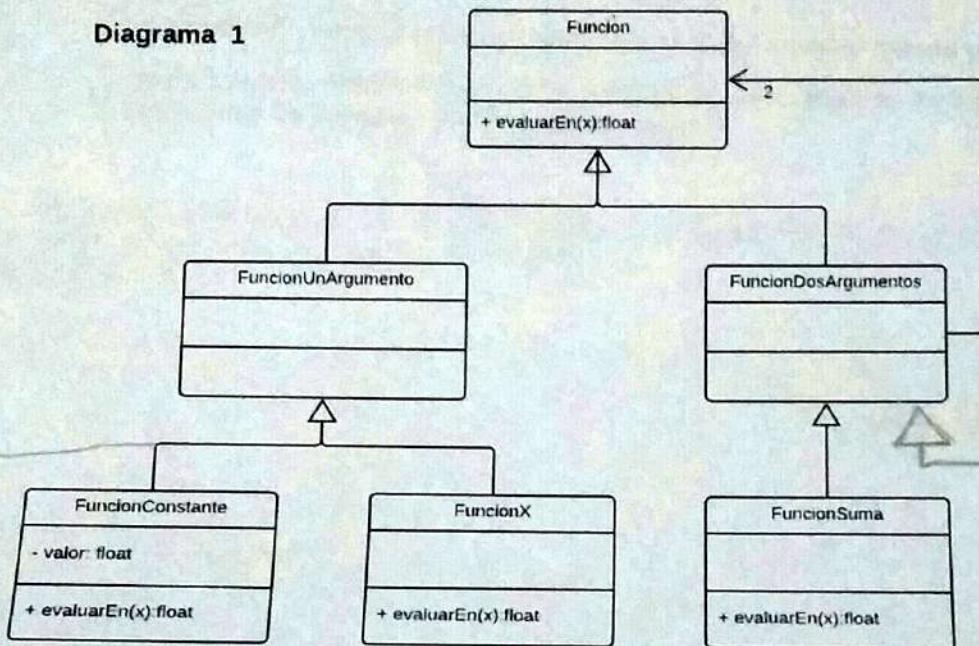
- 4) **Ejercicio conceptual** (se recomienda leer todo el ejercicio antes de comenzar). Se desea analizar la correspondencia entre el diagrama que se muestra y el enunciado que le sigue.

Ponderación: 30%

Envia tus examenes a lawikiifiuba@gmail.com

Implementando un modelo que busque representar funciones matemáticas, proponemos las siguientes opciones:

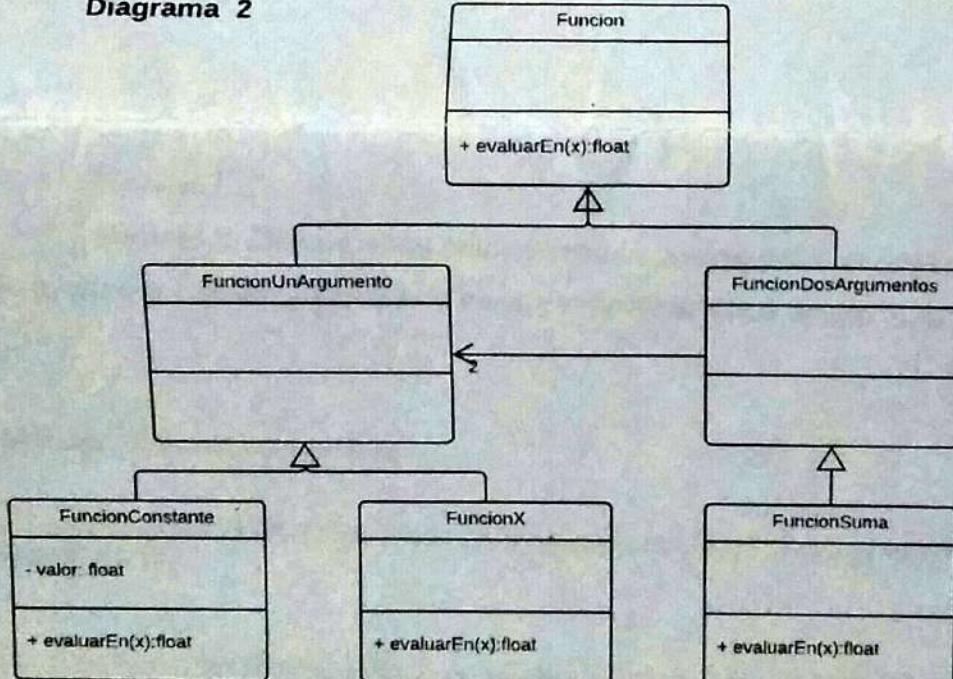
Diagrama 1



ConTESTAR en base a estos diagramas:

- Explique cual de los dos diagramas permite modelar mejor la composición de funciones.
- Explique que limitación ve en el otro diagrama (el que no eligió para el punto anterior).
- extienda el diagrama para incluir las funciones: raíz cuadrada de x, seno de x, y división.

Diagrama 2



FUNCION DIVISION
+evaluarEn(x)

FUNCION EXPONENCIAL
+evaluarEn(x)

FUNCION RAIZ
+evaluarEn(x)

Nombre:	Padrón:
Corrigió:	Nota:

- 1) **Ejercicio de modelado** (se recomienda leer todo el ejercicio antes de comenzar). Se desea modelar parte de un sistema mediante el paradigma de objetos.

Una empresa de envío de comidas a domicilio necesita calcular el monto a pagar por cada pedido. Los detalles de cómo se calculan los precios son:

- Cada comercio tiene una lista de productos con sus precios básicos.
- Los productos pueden ser platos, bebidas, postres y menús. Una bebida tiene un precio único. Los postres y los platos pueden incluir una guarnición (por ejemplo, un flan puede incluir "con dulce de leche" y una milanesa podría incluir un puré o papas fritas), que puede o no tener un precio que se añade al del postre o plato. Un menú consta de plato, postre y bebida, determinándose su costo mediante la suma de sus partes menos un porcentaje que depende del día de la semana: 5% viernes y sábados, 10% los jueves y domingos y 20% de lunes a miércoles.
- Finalmente, el total de la factura se afecta por un coeficiente según la fidelidad del cliente. Un cliente nuevo tiene un descuento del 10%, sólo en el primer pedido. Un cliente habitual, que ha hecho más de 10 pedidos y el último fue en los últimos 15 días tiene un 20% de descuento; si alguna de las dos condiciones no se cumple, paga el precio normal. Un cliente muy fiel, con más de 20 pedidos hechos y el último dentro de los 7 días, tiene un 30% de descuento.
- Los impuestos ya se encuentran incluidos en los precios.

El alumno pensará y modelará:

- **Modelo 1.** Un diagrama de secuencia que muestre la interacción de los objetos y mensajes en el caso del cálculo del precio de un pedido de un menú en día domingo.
 - **Modelo 2.** Escriba dos pruebas automatizadas que prueben el proceso de cálculo de precio de un pedido.
 - **Modelo 3.** Un diagrama de clases de la parte del sistema que sirva para calcular los precios. Incluya todos los métodos que le parezca necesarios en las clases, pero ninguno más. Los métodos que utilice en el diagrama de secuencia y en las pruebas, deben figurar en el diagrama.
- 2) El artículo "Continuous Integration", dice que debemos corregir los errores en el servidor de integración inmediatamente ("fix broken builds immediately"). ¿A qué se refiere? ¿Qué desventajas acarrearía no hacerlo en el marco de la práctica de integración continua? (máximo media carilla)
- 3) Cuando hablamos de calidad de código dijimos que los comentarios en el código no deberían abundar. ¿Para qué casos especiales sí se recomienda escribir comentarios? (máximo una carilla)

Nombre:		Padrón:	
Corrigió:	DIEGO SÁNCHEZ Ingeniero en Informática (UBA)	Nota:	

- 1) **Ejercicio de modelado** (~~Mat. COPITEC N° 5838~~ Se recomienda leer todo el ejercicio antes de comenzar). Se desea modelar parte de un sistema mediante el paradigma de objetos.

Una empresa de servicio de entrega de soda a domicilio (vía cañerías dedicadas) necesita calcular la facturación a cada cliente. Los detalles de cómo se calculan los precios son:

- Hay una tarifa fija (que no depende de la cantidad de litros), que es diferente según la categoría del cliente: cliente domiciliario, cliente comercial, gran comercio.
- La tarifa fija también varía en función del tipo de zona: si la zona es careniada queda en su tarifa fija, en zona media se aumenta un 30%, y en zona elite sube un 70%.
- El resto del precio se define por litro y es igual para todos los clientes de una misma categoría y una misma zona.
- Luego hay impuestos locales y regionales. Los locales son un porcentaje sobre el precio por litro. Los regionales son un porcentaje también sobre el precio del litro, el porcentaje que se aplica es el 50% del porcentaje de ajuste zonal.

El alumno pensará y modelará:

- **Modelo 1.** Un diagrama de secuencia que muestre la interacción de los objetos y mensajes en el caso del cálculo de facturación a un cliente comercial, en zona media.
- **Modelo 2.** Escriba dos pruebas automatizadas que prueben el proceso de cálculo de facturación de un cliente.
- **Modelo 3.** Un diagrama de clases de la parte del sistema que sirva para facturación a clientes. Incluya todos los métodos que le parezca necesarios en las clases, pero ninguno más. Los métodos que utilice en el diagrama de secuencia y en las pruebas, deben figurar en el diagrama.

- 2) El artículo "Unit Testing Guidelines" dice que debemos cubrir los casos de borde ("cover boundary cases"). ¿A qué se refiere? ¿Por qué le parece importante esta recomendación? (máximo media carilla)
- 3) El artículo "Principios de diseño de Smalltalk" dice que el propósito del lenguaje es proveer un esquema para la comunicación. ¿Qué quiere decir Ingalls con esta afirmación? ¿En qué impacta esto en su práctica cotidiana de programación? (máximo media carilla)

Nombre:	Padrón:
Corrigió:	Nota:

Cada ítem del examen se va a calificar con un valor entre 0 y 1. Para aprobar el examen se deberán cumplir todas las siguientes condiciones:

- En cada uno de los dos primeros ejercicios se debe tener al menos un puntaje de 0,6.
- Se debe sumar como mínimo el 60% del puntaje total obtenido mediante la ponderación de cada ejercicio (ver el coeficiente de ponderación en cada ejercicio).

1) **Ejercicio de modelado** (se recomienda leer todo el ejercicio antes de comenzar). Se desea modelar parte de un sistema mediante el paradigma de objetos.

Ponderación: 30%

En el área metropolitana de Buenos Aires, el transporte público utiliza una tarjeta prepaga para abonar los viajes, conocida como "tarjeta SUBE". Además, existen las siguientes cuestiones a tener en cuenta:

- Cada usuario tiene una tarjeta que almacena el saldo de la misma. En líneas generales, cada vez que un viaje se realiza se vuelve a guardar el saldo en la tarjeta.
- Las tarifas base disponibles dependen del medio de transporte, tren, subte, colectivo. Para cada medio se calcula distinto:
 - o En el caso de los colectivos, el pago de un viaje se realiza antes de comenzarlo, y se basa en el pedido del pasajero del viaje que va a realizar.
 - o En el caso del subte también se paga al inicio y es fijo para toda la red.
 - o En el caso de los trenes, el viaje se calcula y se cobra al salir de la estación de destino, en base al registro que el pasajero hace con su tarjeta al ingresar en la estación de origen del viaje y la de destino.
- La tarifa final para un viaje en un transporte no depende sólo de la tarifa base, sino que también influye lo que el usuario de la tarjeta ha hecho en las últimas dos horas: si en el transcurso de 2 horas de comenzado un viaje, el titular de la tarjeta comienza un segundo viaje, éste costará un 50% del valor normal. Si además comienza un tercer viaje, costará un 25%.
- La tarjeta SUBE se carga mediante distintos mecanismos, y se admite un saldo negativo de hasta 30 pesos. Si no hay siquiera -30 pesos en la tarjeta, el pasajero no podrá viajar.

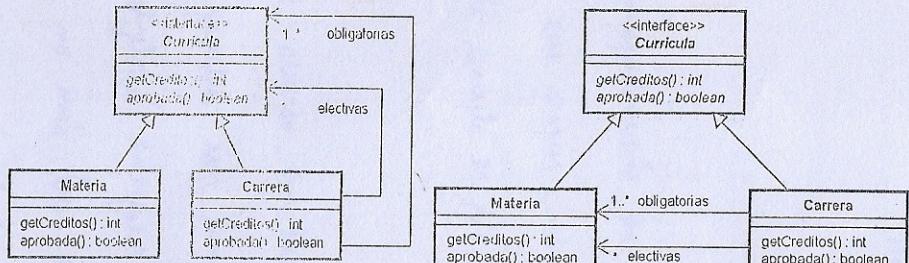
El alumno pensará y modelará:

- **Modelo 1.** Un diagrama de secuencia que muestre el cálculo del valor de tres viajes sucesivos en colectivo, subte y colectivo, en ese orden, con la interacción de los objetos y mensajes para obtener el precio de cada pasaje. Ojo: debe verificar el saldo de la tarjeta cada vez.
- **Modelo 2.** Dos pruebas automatizadas que modelen el caso anterior.

(puede ayudarse con un diagrama de clases simplificado que ayude al corrector a entender mejor los anteriores)

2) **Ejercicio conceptual** (se recomienda leer todo el ejercicio antes de comenzar). Se desea analizar la correspondencia entre el diagrama que se muestra y el enunciado que le sigue.
Ponderación: 30%

En una universidad, las currículas de las carreras se establecen mediante una serie de materias, algunas de ellas obligatorias y otras electivas, cada una con una cantidad de créditos asociadas. Las carreras no pueden ser parte de otras carreras. Para aprobar una carrera se deben aprobar todas las materías obligatorias y se debe llegar a una suma de créditos, entre obligatorias y electivas, que es específica de cada carrera. Analice los siguientes diagramas de clases, teniendo en cuenta que el mensaje "getCreditos" se refiere a los créditos que otorga una materia en la clase "Materia", mientras que en la clase "Carrera" se refiere a la cantidad de créditos necesarios para graduarse. Responda las preguntas que siguen:



- ¿Cuál de ambos diagramas le parece que se ajusta mejor al problema, el de la izquierda o el de la derecha? ¿Por qué?
- El diagrama que usted NO eligió, ¿qué otro problema resuelve?
- ¿Qué significa, en el contexto de este diagrama, el sentido de las flechas en las asociaciones?
- ¿Desde el punto de vista de la calidad de código, ¿es correcto afirmar: «el mensaje "getCreditos" se refiere ... en la clase "Materia", mientras que en la clase "Carrera" se refiere a ...»? ¿Por qué?

Para evaluar conceptos teóricos:

3) ¿Qué significa el encapsulamiento en POO? Explíquelo en 3 líneas. Marque una X sobre el código de abajo que mejor respeta el encapsulamiento (mover el punto 5 pixeles en sentido horizontal):

- punto := Punto new inicializarEnX: (punto getX + 5) enY: punto getY
 punto trasladarEnX: 5
 punto trasladarHorizontal: 5
 punto x := punto x + 5
 punto setX: (punto getX + 5)

Ponderación: 20%

4) Supongamos que alguien hace la siguiente afirmación: si usamos TDD en el desarrollo, vamos a poder realizar más fácilmente el control de calidad y nos quedará algo de documentación ya generada desde el comienzo. ¿Está de acuerdo con estas dos afirmaciones? Explique por qué. (máximo media carilla)

Ponderación: 20%

Nombre:	Padrón:
Corrigió:	Nota:

Cada ítem del examen se va a calificar con un valor entre 0 y 1. Para aprobar el examen se deberán cumplir todas las siguientes condiciones:

- En cada uno de los dos primeros ejercicios se debe tener al menos un puntaje de 0,6.
- Se debe sumar como mínimo el 60% del puntaje total obtenido mediante la ponderación de cada ejercicio (ver el coeficiente de ponderación en cada ejercicio).

1) **Ejercicio de modelado** (se recomienda leer todo el ejercicio antes de comenzar). Se desea modelar parte de un sistema mediante el paradigma de objetos.

Ponderación: 30%

En el área metropolitana de Buenos Aires, el transporte público utiliza una tarjeta prepaga para abonar los viajes, conocida como "tarjeta SUBE". Además, existen las siguientes cuestiones a tener en cuenta:

- Cada usuario tiene una tarjeta que almacena el saldo de la misma. En líneas generales, cada vez que un viaje se realiza se vuelve a guardar el saldo en la tarjeta.
- Las tarifas base disponibles dependen del medio de transporte, tren, subte, colectivo. Para cada medio se calcula distinto:
 - o En el caso de los colectivos, el pago de un viaje se realiza antes de comenzarlo, y se basa en el pedido del pasajero del viaje que va a realizar.
 - o En el caso del subte también se paga al inicio y es fijo para toda la red.
 - o En el caso de los trenes, el viaje se calcula y se cobra al salir de la estación de destino, en base al registro que el pasajero hace con su tarjeta al ingresar en la estación de origen del viaje y la de destino.
- La tarifa final para un viaje en un transporte no depende sólo de la tarifa base, sino que también influye lo que el usuario de la tarjeta ha hecho en las últimas dos horas: si en el transcurso de 2 horas de comenzado un viaje, el titular de la tarjeta comienza un segundo viaje, éste costará un 50% del valor normal. Si además comienza un tercer viaje, costará un 25%.
- La tarjeta SUBE se carga mediante distintos mecanismos, y se admite un saldo negativo de hasta 30 pesos. Si no hay siquiera -30 pesos en la tarjeta, el pasajero no podrá viajar.

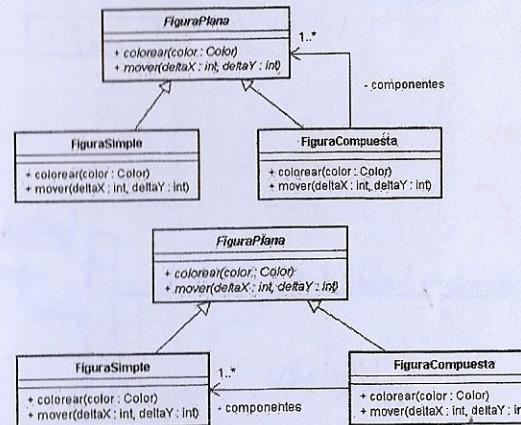
El alumno pensará y modelará:

- **Modelo 1.** Un diagrama de secuencia que muestre el escenario del cálculo del valor de un viaje en tren, que depende de la estación de entrada y de salida (use algún servicio externo para obtener los precios), con la interacción de los objetos y mensajes para obtener el precio del pasaje. Ojo: debe verificar el saldo de la tarjeta al inicio.
- **Modelo 2.** Un diagrama de clases que modele y sirva para comprender el diagrama anterior.
- **Modelo 3.** Dos pruebas automatizadas que modelen el caso del diagrama de secuencia.

- 2) **Ejercicio conceptual** (se recomienda leer todo el ejercicio antes de comenzar). Se desea analizar la correspondencia entre el diagrama que se muestra y el enunciado que le sigue.

Ponderación: 30%

En un programa de dibujo de figuras planas, se permite colorear y mover figuras. También se definen figuras compuestas por agrupación, las cuales también pueden colorearse o moverse en forma conjunta. Las figuras compuestas no pueden contener otras figuras compuestas. Analice los siguientes diagramas de clases y responda las preguntas que siguen:



- a. ¿Cuál de ambos diagramas le parece que se ajusta mejor al problema, el de la izquierda o el de la derecha? ¿Por qué?
- b. El diagrama que usted NO eligió, ¿qué otro problema resuelve?
- c. Tomando el diagrama que haya elegido, y suponiendo que las clases "FiguraPlana" y "FiguraSimple" ya han sido implementadas, implemente la clase "FiguraCompuesta", escribiendo previamente las pruebas automáticas que permitan probar luego la calidad de la solución..
- d. ¿Qué debería agregar en el diagrama para permitir que un usuario agregue figuras a una figura compuesta (inótese que no lo hemos previsto!)?

Para evaluar conceptos teóricos:

- 3) En un contexto de reutilización, cuando necesita que la clase nueva tenga la misma interfaz que la previa, usa delegación, dejando la herencia para los demás casos. ¿Es así o al revés? ¿Por qué? (máximo media carilla)
- Ponderación: 20%
- 4) ¿Qué son las pruebas de regresión? ¿Para qué sirven? ¿Qué relación tienen con la idea de automatización?
- Ponderación: 20%

Nombre:	<i>Z. Llorente</i>	Padrón:
Corrigido:		Nota:

- 1) Ejercicio de modelado (se recomienda leer todo el ejercicio antes de comenzar). Se desea modelar la obtención del precio de un servicio de Playa de un bañear mediante el paradigma de objetos. Algunas entidades de dominio podrían ser:

Prestación: una carpeta, sombrilla o comida que vende el bañear.

Combo: un conjunto de uno o más prestaciones que puede venderse al público. Tener en cuenta que no cualquier conjunto de prestaciones puede venderse como combo.

Categoría: cada prestación puede tener categorías. Por ejemplo, las carpas pueden tener vista interna, vista lateral o vista al mar. Las comidas pueden ser sólo bebidas, media pensión o pensión completa, etc.

Servicio de playa: un combo por una quincena, incluyendo categorías, vendido a un bañista. Un servicio de playa tiene un precio, cuya calculo es el objetivo de este ejercicio.

El precio de un servicio de playa no es único ni es la suma de los precios de las prestaciones. Se calcula de esta manera:

- Cada prestación tiene un precio básico para cada categoría. Cada combo tiene un precio básico para cada categoría, que es la suma de los básicos de las prestaciones.
- Todo servicio de playa se vende al precio básico del combo desde 180 días antes de la fecha de comienzo del servicio hasta 45 días antes.
- Si 45 días antes de la fecha de comienzo del servicio ya se ha vendido el 80% de la prestación más cara, el precio del servicio se incrementa en un 20% sobre el básico. Si no, se reduce en un 20%. Esto se mantiene hasta 15 días antes de la fecha de comienzo del servicio.
- Si 15 días antes de la fecha de comienzo del servicio ya se ha vendido el 90% de la prestación más cara, el precio del servicio se incrementa en un 30% sobre el básico. Si no, se reduce en un 30%. Esto se mantiene hasta 5 días antes de la fecha de comienzo del servicio.
- Si 5 días antes de la fecha de comienzo del servicio ya se ha vendido el 95% de la prestación más cara, el precio del servicio se incrementa en un 40% sobre el básico. Si no, se reduce en un 40%. Esto se mantiene hasta el día anterior al de la fecha de comienzo del servicio.
- El día anterior a la fecha de comienzo del servicio, si quedan prestaciones sin vender, se reduce el precio del servicio al 50% del básico.

El alumno pensará y modelará un escenario simple de obtención del precio de un servicio de playa. Para ello, deberá entregar:

0,2 • **Modelo 1.** Un diagrama de secuencia de ese escenario. Es importante mantener el nivel de abstracción adecuado, sin indicar cuestiones de implementación de mensajes ajenos a este escenario. Por ejemplo, no tiene que modelar nada que implique calcular la cantidad de prestaciones sin vender: use correctamente la abstracción para evitarlo.

0,1 • **Modelo 2.** Escriba dos pruebas automatizadas no triviales y distintas, necesarias para probar el comportamiento modelado en el diagrama de secuencia, usando SUnit.

0,1 • **Modelo 3.** Un diagrama de clases de la parte del sistema que sirva para calcular precios de servicios. Use nombres adecuados para todas las clases, métodos y asociaciones que defina. Incluya todos los métodos que le parezca necesarios en las clases, pero ninguno más. Los métodos que utilice en los puntos a y b, deben figurar en el diagrama.

2) El artículo "Unit Testing Guidelines", dice que hay que escribir pruebas para reproducir fallas ("write tests to reproduce bugs"). ¿A qué se refiere? ¿Por qué le parece que es importante? (máximo media canilla)

3) El código repetido suele mencionarse como un mal síntoma de diseño. Eliminarlo implicaría usar alguna refactorización, que dependerá de si la repetición se da dentro de una misma clase, entre dos clases hermanas o entre clases no vinculadas.

- 1 • Explique qué es una refactorización, y qué se busca con la práctica de refactoring (3 líneas).
- 1 • Relate las 3 refactorizaciones que haría en los 3 casos de código repetido mencionados más arriba usando código o diagramas para mostrarlas.
- 0 • ¿Qué hace que una refactorización sea tal y no otro tipo de cambio de código? ¿Cómo puede garantizar esto? (3 líneas)

Nombre:

Corrigió:

- 1) **Ejercicio de modelado** (se recomienda leer todo el ejercicio antes de comenzar). Se desea modelar la obtención de la disponibilidad de prestaciones de playa de un balneario mediante el paradigma de objetos. Algunas entidades de dominio podrían ser:

Prestación: una carpa, sombrilla o comida que vende el balneario.

Combo: un conjunto de uno o más prestaciones que puede venderse al público. Tener en cuenta que no cualquier conjunto de prestaciones puede venderse como combo.

Categoría: cada prestación puede tener categorías. Por ejemplo, las carpas pueden tener vista interna, vista lateral o vista al mar, las comidas pueden ser sólo bebidas, media pensión o pensión completa, etc.

Servicio de playa: un combo por una quincena, incluyendo categorías, vendido a un bañista. Un servicio de playa tiene un precio.

Quincena: periodo de tiempo de 2 semanas del calendario para el cual se venden servicios.

Ante un pedido de un cliente, que pide un combo con prestaciones con categoría para una quincena determinada, el sistema analizará si todas las prestaciones están disponibles con las categorías pedidas y luego:

- El sistema calcula el precio y le informa al cliente.
- Si el cliente acepta, cambia la disponibilidad de las prestaciones e informa al administrador del sistema la cantidad y el porcentaje desociables
- Si el cliente no acepta, aplica un algoritmo (ajeno a este escenario) para ofrecerle un combo alternativo (por ejemplo, cambiando fechas, categorías o prestaciones) y vuelve a informar al cliente para que pueda aceptar o no. Y así sucesivamente.

Si alguna prestación de las solicitadas no tiene disponibilidad, aplica un algoritmo (ajeno a este escenario) para ofrecerle un combo alternativo (por ejemplo, cambiando fechas, categorías o prestaciones) y procede como en el caso anterior.

El alumno pensará y modelará un escenario de pedido de un combo por un cliente para el caso en que no haya disponibilidad del mismo y acepta el primer combo alternativo que le ofrece el sistema. Para ello, deberá entregar:

- **Modelo 1.** Un diagrama de secuencia de ese escenario. Es importante mantener el nivel de abstracción adecuado, sin indicar cuestiones de implementación de mensajes ajenos a este escenario. Por ejemplo, no tiene que modelar nada que implique calcular precios, disponibilidades o cómo manejar las notificaciones al usuario: use correctamente la abstracción para evitarlo.
- **Modelo 2.** Escriba dos pruebas automatizadas no triviales y distintas, necesarias para probar el comportamiento modelado en el diagrama de secuencia, usando SUnit.
- **Modelo 3.** Un diagrama de clases de la parte del sistema que sirva para el escenario. Use nombres adecuados para todas las clases, métodos y asociaciones que defina. Incluya todos los métodos que le parezca necesarios en las clases, pero ninguno más. Los métodos que utilice en los puntos a y b, deben figurar en el diagrama.

- 2) Explique en – a lo sumo – una carilla el planteo de Martin Fowler en su artículo "What's a Model For" ("Para qué sirve un modelo").
- 3) En un contexto de reutilización, cuando necesita que la clase nueva tenga la misma interfaz que la previa, usa delegación, dejando la herencia para los demás casos. ¿Es así o al revés? ¿Por qué? (máximo media carilla)

Nota:

Corrig.

- 1) **Ejercicio de modelado** (se recomienda leer todo el ejercicio antes de comenzar). Se desea modelar parte del sistema de un servicio de guardia de un centro de salud mediante el paradigma de objetos.

En el centro de salud, cuando un paciente llega a la guardia:

- Se lo deriva a un médico de guardia, quien evalúa al paciente y decide si requiere o no un estudio de diagnóstico (laboratorio, radiología, etc.).
- Si no requiriese ningún estudio, se le indica un tratamiento y/o una medicación y se emite un informe que queda archivado en el centro de salud. El paciente paga, en caso de corresponder, y se retira.
- Si el paciente requiriese un estudio, se lo deriva al área correspondiente (radiología, laboratorio, etc.), donde le harán el estudio en forma ambulatoria. Una vez obtenido el estudio, el paciente vuelve al médico de guardia, quien le indica un tratamiento y/o una medicación y se emite un informe que queda archivado en el centro de salud. El paciente paga, en caso de corresponder, y se retira.

El cálculo del valor de la prestación será la suma de la consulta y el estudio de diagnóstico, y dependerá de si el paciente tiene o no obra social y el tipo de afiliación:

- Si no tiene obra social, deberá pagar los aranceles que se obtienen de un sistema externo, al que se accede mediante un objeto instancia de la clase ServicioAranceles.
- Si tiene obra social con cobertura completa, se factura un arancel a la obra social, que también se obtiene del mismo sistema externo, al cual accedemos mediante la clase ServicioAranceles, y que podría ser distinto para cada obra social (tip: enviar la obra social como argumento).
- Si tiene obra social con copago, se facturará una parte a la obra social y otra al paciente. Ambos valores se obtendrán también del sistema externo al cual accedemos mediante la clase ServicioAranceles.

Recordar que siempre se deberá calcular el arancel de la consulta, y si hubo, el del estudio.

El alumno no pensará y modelará:

- **Modelo 1.** Un diagrama que modele los distintos estados por los que pasa un paciente desde que ingresa al centro de salud hasta que sale, con indicación clara de las transiciones y los eventos.
 - **Modelo 2.** Un diagrama de secuencia del cálculo del arancel en el caso de obra social con copago. Es importante mantener el nivel de abstracción adecuado, sin indicar cuestiones de implementación de mensajes ajenos a este escenario. Por ejemplo, no tiene que modelar nada del funcionamiento interno de la instancia de la clase ServicioAranceles.
 - **Modelo 3.** Escriba dos pruebas automatizadas no triviales y distintas, necesarias para probar el comportamiento modelado en el diagrama de secuencia.
 - **Modelo 4.** Un diagrama de clases de la parte del sistema que sirva para calcular aranceles. Use nombres adecuados para todas las clases, métodos y asociaciones que defina. Incluya todos los métodos que le parezca necesarios en las clases, pero ninguno más. Los métodos que utilice en el diagrama de secuencia y en las pruebas, deben figurar en el diagrama.
- 2) En el artículo "What's a Model For" / "Para qué sirve un modelo?", Fowler dice que prefiere usar los modelos como "modelos esqueléticos". ¿A qué se refiere? ¿Cuánto se acerca esa visión de Fowler al uso que hacemos de UML en la materia? (máximo media carilla)
- 3) ¿Qué es el encapsulamiento? ¿Cómo lo asegura? ¿Qué ventajas tiene? (máximo media carilla)

Nombre:

Corrigió:

Ortiz

Padrón:

Nota:

10 (dic 2) JFJ

- 1) **Ejercicio de modelado** (se recomienda leer todo el ejercicio antes de comenzar). Se desea modelar parte del sistema de facturación de un supermercado mediante el paradigma de objetos.

En el supermercado, cuando un cliente llega a la línea de cajas:

- Debe marcar en un monitor si desea un ticket común o una factura con IVA discriminado. En función de ello, el sistema le indicará a qué caja debe dirigirse.
- Una vez que llega a la caja respectiva, y sólo si es consumidor final, debe informar a quien lo atiende si tiene una tarjeta de puntos o no. Si la tuviera, podrá obtener un descuento por su compra, que se aplica mediante porcentajes diferentes sobre diferentes productos.
- Luego de ello, en ambos casos, se procesarán los productos a comprar, se calcularán sus precios, el comprador pagará con algún medio de pago (efectivo, tarjetas, tickets de descuento) y se retira del local con la mercadería.

El cálculo del total a pagar, se calculará de la siguiente manera:

- Si el cliente desea factura con IVA discriminado, se irán calculando los precios de los productos y se sumará el total al final, discriminando el IVA (tip: esto último no importa para este ejercicio). Luego, antes de calcular la suma total, el sistema deberá adicionar percepciones de impuestos al total, que se obtienen de enviar los detalles de la factura a un sistema externo, al que se accede mediante un objeto instancia de la clase ServicioPercepciones.
- Si el cliente es consumidor final y no tiene tarjeta de puntos, el total de la factura es simplemente la suma de los precios de los productos que compra.
- Si el cliente tiene tarjeta de puntos, cada uno de los productos que compra puede tener asociado un descuento. Para calcularlo, el sistema se comunica con otro sistema del supermercado, enviando mensajes a un objeto instancia de la clase DescuentosTarjetaDePuntos. La suma de los precios de los productos, con sus descuentos en cada caso, es el total a pagar.

✓ El alumno pensará y modelará:

- **Modelo 1.** Un diagrama que modele los distintos estados por los que pasa un cliente desde que llega a la línea de cajas hasta que sale del supermercado, con indicación clara de las transiciones y los eventos.
- **Modelo 2.** Un diagrama de secuencia del cálculo del total de una factura en el caso de ser consumidor final con tarjeta de puntos. Es importante mantener el nivel de abstracción adecuado, sin indicar cuestiones de implementación de mensajes ajenos a este escenario. Por ejemplo, no tiene que modelar nada del funcionamiento interno de la instancia de la clase DescuentosTarjetaDePuntos.
- **Modelo 3.** Escriba dos pruebas automatizadas no triviales y distintas, necesarias para probar el comportamiento modelado en el diagrama de secuencia.
- **Modelo 4.** Un diagrama de clases de la parte del sistema que sirva para calcular totales a pagar. Use nombres adecuados para todas las clases, métodos y asociaciones que defina. Incluya todos los métodos que le parezca necesarios en las clases, pero ninguno más. Los métodos que utilice en el diagrama de secuencia y en las pruebas, deben figurar en el diagrama.

- ✓ 2) En el artículo "What's a Model For" / "Para qué sirve un modelo?", Fowler marca una diferencia entre "resaltar los detalles importantes" e "ignorar los detalles". ¿A qué se refiere? ¿Por qué le parece que es importante en el contexto del artículo? (máximo media carilla)
- ✓ 3) Supongamos que alguien hace la siguiente afirmación: si usamos TDD en el desarrollo, vamos a poder realizar más fácilmente el control de calidad y nos quedará algo de documentación ya generada desde el comienzo. ¿Está de acuerdo con estas dos afirmaciones? Explique por qué. (máximo media carilla)

1) Un banco contiene cuentas bancarias de diversos tipos, que difieren en el límite de extracción que se les puede hacer. Para ello, se han definido y desarrollado 3 clases "Cuenta", "CuentaCorriente" y "CajaAhorro" (la primera es la madre de las otras dos), con métodos "getSaldo", "getTitular", "depositar" y "extraer", además de un método abstracto "getLimite" en "Cuenta", redefinido en las dos clases hijas. Hay también una clase "ListaCuentas", que contiene una colección de todas las cuentas del banco. Se pide:

- a. Hacer un diagrama de clases de lo ya desarrollado.
- b. Se pide una modificación. A partir de la misma, cada vez que se haga un depósito de más de 10000 pesos, se deberá depositar 10 pesos en todas las demás cuentas que ese mismo cliente tenga en el banco. Haga el diagrama de secuencia del nuevo método "depositar", escriba las pruebas unitarias que permitirán probar la modificación y haga también el nuevo diagrama de clases. No hace falta que escriba el código que hace funcionar las pruebas.

2) El artículo "Unit Testing Guidelines", dice que hay que escribir pruebas para reproducir fallas ("write tests to reproduce bugs"). ¿A qué se refiere? ¿Por qué le parece que es importante? (máximo media carilla)

3) ¿Por qué, en el procedimiento de TDD (Test Diven Development), se hace énfasis en que probemos que las pruebas fallen apenas se escriben los tests? (máximo media líneas, citando al ejercicio 1 para exemplificar)

De las siguientes afirmaciones, diga si son verdaderas o falsas, directamente sobre esta hoja. No es necesario justificar las respuestas, pero puede hacerlo si lo cree necesario.

- 4) Antes de agregar un comentario para aclarar, debería tratar de aclarar el código. Verdadero
- 5) Durante una refactorización, conviene hacer la mayor cantidad posible de cambios seguidos, y recién luego correr las pruebas falso .