Programación 4

PARCIAL FINAL EDICIÓN 2017

Por favor siga las siguientes indicaciones:

- Escriba con lápiz y de un solo lado de las hojas
- Escriba su nombre y número de documento en todas las hojas que entregue
- Numere las hojas e indique el total de hojas en la primera de ellas
- Recuerde entregar su número de parcial junto al parcial
- Está prohibido el uso de computadoras, tabletas o teléfonos durante el parcial

Problema 1 (35 puntos)

Parte I)

a) ¿Qué es un tipo asociativo?
b) En el contexto de la metodología de desarrollo vista en el curso, ¿cual es la diferencia entre un diagrama UML y un artefacto?

Parte II)

Una empresa de software le ha pedido a Ud. un sistema para gestionar las oportunidades comerciales que su Área Comercial tiene con diferentes clientes.

Toda oportunidad comienza siendo un Dato (ej: cuando la empresa de software se entera que una empresa cliente está buscando proveedores). Luego de recabar cierta información, y cuando el usuario lo desea, se convierte en Oportunidad Calificada, luego de lo cual puede terminar de dos formas: Oportunidad Ganada (cuando la empresa ganó el trabajo) u Oportunidad Perdida (cuando no):

Una Oportunidad Calificada tiene la misma información que un Dato, aunque agrega la fecha en que se convirtió en Calificada. La información en común es: nombre (que identifica), fecha de creación, contacto (ver más adelante), si hay presupuesto asignado por parte de la empresa cliente o no, si se realizó una presentación a la empresa cliente o no, y quienes están a cargo de tomar la decisión de compra dentro de la empresa cliente (ver más adelante). Luego, una Oportunidad Ganada contiene además el monto del proyecto y la fecha de inicio, mientras que una Oportunidad Perdida contiene el motivo por el cual la empresa de software perdió esa oportunidad.

Debe brindarse una trazabilidad de las oportunidades desde su comienzo (Dato) hasta su fin (ya sean Ganadas o Perdidas), no perdiendo información en el camino. Una oportunidad calificada deberá tener una sola persona de contacto (de la cual se debe conocer su nombre, mail, teléfonos y en qué empresa cliente trabaja) y podrá (o no)

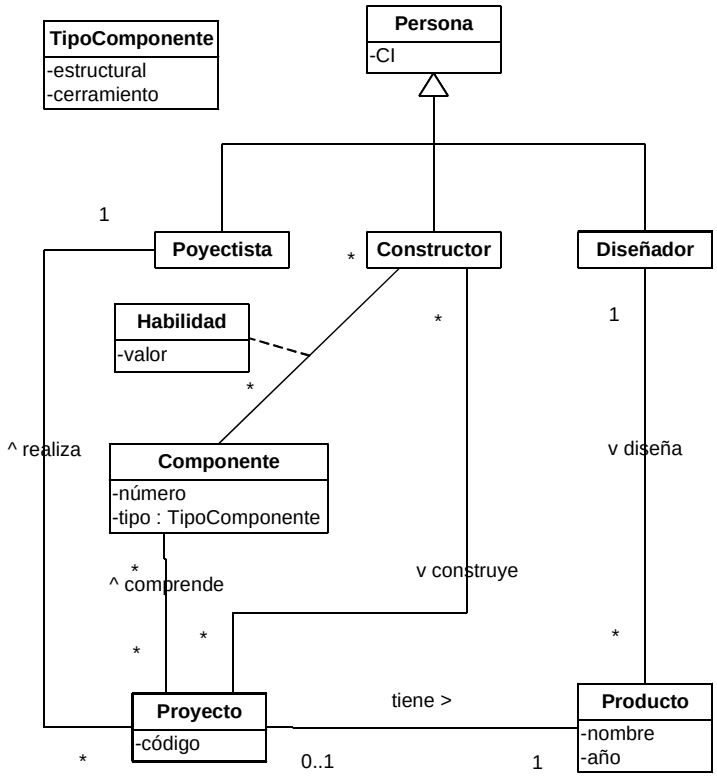
tener asociadas muchas personas que participan de la toma de decisión dentro de la empresa cliente (la misma del contacto).

Se pide: realizar el Modelo de Dominio de la realidad anterior, con restricciones en lenguaje natural.

Problema 2 (30 puntos)

Parte I)

En una compañía de producción de muebles existe un departamento que diseña los productos. Cada producto tiene un proyecto asociado, del que se conoce su código y su proyectista. Un proyecto comprende varios componentes, identificados por un número y que pueden ser de tipo estructural o cerramiento. Un proyecto es llevado a cabo por constructores, que deben tener habilidad para la construcción de los componentes que comprenden al proyecto. Para cada constructor se sabe su grado de habilidad en cada componente, representado como un número en el rango [0..5]; se considera que un constructor tiene habilidad sobre un componente si su valor es mayor a 2. Finalmente, las personas son identificadas por su documento de identidad.



Se pide:

Realice el diagrama de comunicación de la siguiente operación:

`void agregarComponenteAProy(String proy, int numComp, TipoComponente tipo, Map<String, int> habilidades)`	
Descripción	Agrega un componente al proyecto. Crea el componente y le asocia la habilidad correspondiente a todos los constructores utilizando la información de habilidades.
Parámetros	• `proy`: identifica al proyecto • `numComp`: identifica al componente • `tipo`: tipo de componente que se desea agregar • `habilidades`: mapeo de <clave, valor> para constructores y habilidades <ci, valor> respectivamente
Precondiciones	• Existe en el Sistema una única instancia de `Proyecto` con código `proy`. • Existe en el Sistema una única instancia de `Constructor` para cada clave de `habilidades`. • Los valores de `habilidades` están en el rango [2..5]. • No existe en el Sistema un `Componente` identificado por `numComp`.
Postcondiciones	• Se crea una nueva instancia de `Componente` asociada a todos los `Constructores` con su respectiva `Habilidad` definida. • Se agrega el `Componente` al `Proyecto`.

Parte II)

Se desea poder monitorear los estados [PENDIENTE, EN CURSO, COMPLETO] de construcción de un componente específico para cada proyecto; adicionalmente se desea encapsular la lógica de transición entre estados en clases individuales. Por otro lado, se desea tener un mecanismo de suscripción de personas para que estas reciban notificaciones de cualquier cambio de estado en la construcción de un componente específico.

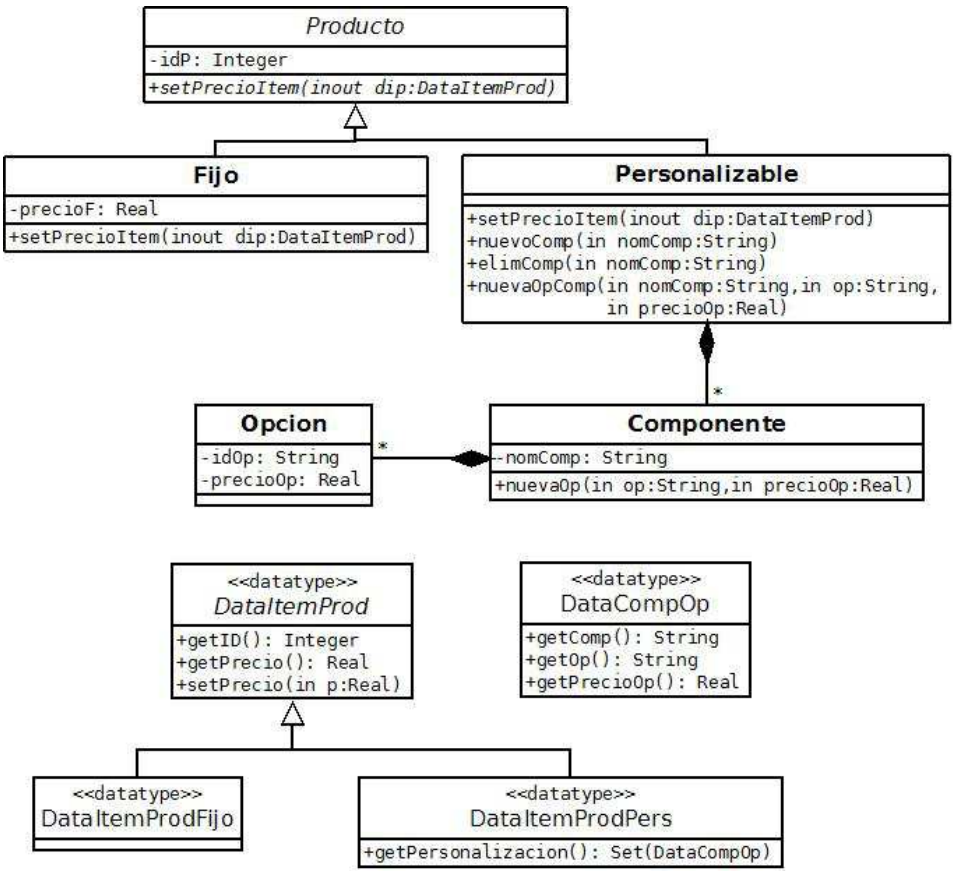
Se pide:
Realizar un DCD con el diseño propuesto para soportar estos requerimientos siguiendo los criterios GRASP y procurando realizar la menor cantidad de modificaciones posibles al modelo de dominio dado. En caso de utilizar patrones especifique cómo han sido aplicados.

Problema 3 (35 puntos)

Una empresa de venta online ofrece productos tanto fijos como personalizables. Una vez que el cliente selecciona un producto, se genera su correspondiente ítem de compra. Dado un producto concreto, si este es fijo, sus ítems son todos idénticos y tienen un precio fijo. Un producto personalizable tiene una lista de componentes, cada uno de los cuales tiene varias opciones que permiten configurar el producto a comprar (ítem), cuyo precio final dependerá de la (única) opción seleccionada para cada componente.

Por ejemplo, los ítems de compra del producto "cargador para celular X" son todos idénticos. Por otro lado, el producto "celular X" tiene dos componentes personalizables: la pantalla (que tiene dos opciones de calidad diferente y por lo tanto de precio diferente) y la carcaza (que tiene tres opciones, también con diferente calidad y precio). Cuando un cliente compra un "celular X", debe seleccionar una (y solo una) opción, para cada uno de sus componentes.

El diagrama de la figura muestra el diseño parcial de la solución. El datatype `DataItemProd` representa una compra concreta (ítem) de un producto por parte de un cliente; puede tener información de un producto fijo como de uno personalizable. En el caso que sea personalizable, se tiene una lista de opciones, una por cada componente del producto.



Considerar que:

- Las operaciones `nuevoComp` y `elimComp` de la clase `Personalizable` agregan y eliminan respectivamente un nuevo componente (identificado por un string) a un producto personalizable.
- La operación `nuevaOpComp` de la clase `Personalizable` agrega una nueva opción (mediante su identificador y precio) a un componente de un producto personalizable.
- La operación `setPrecioItem` recibe un `DataItemProd`, calcula su precio y se lo setea. Si es un ítem de producto `Fijo`, el precio equivale al valor `precioF`. Si es `Personalizable`, el precio equivale a la suma de los precios de cada opción, información que viene almacenada en el datatype `DataCompOp` para cada componente. Asuma que en esta operación, la clase `Fijo` siempre recibe un `DataItemProdFijo`, mientras que la clase `Personalizable` siempre recibe un `DataItemProdPers`.

Se pide:

a) Implemente los .h de todas las clases y datatypes del diagrama.
b) Implemente los .cpp de las clases `Fijo`, `Personalizable` y `Componente`.

Aclaraciones:

- No incluya directivas al precompilador.
- No implemente operaciones get y set, salvo las que figuran explícitamente en el diagrama.
- Implemente constructores y destructores cuando corresponda.
- Puede utilizar colecciones genéricas o paramétricas (STL), con sus respectivos iteradores.
- Asumir que todas las operaciones se invocan con datos válidos.