UNNOBA

Programación Orientada a Objetos
2020

Práctica 1: Diseño OO

1. Presentación y objetivos

Se desea repasar en forma práctica los conceptos de orientación a objetos y Java desarrollados en las materias anteriores, así como aplicar los conceptos de diseño orientado a objetos y patrones explicados en la teoría.
El objetivo de esta actividad es entonces la resolución de problemas utilizando los conceptos de objetos, Java y diseño orientado a objetos.

2. Materiales y recursos

Para el desarrollo de la actividad deberán utilizarse los siguientes materiales y recursos:

- Plataforma ED:
  - Clase 1.
  - Clase 2.
  - Clase Patrones de Diseño.
  - Libro Design Patterns.
  - Foro de actividades prácticas.
- Teoría y práctica de las materias:
  - Introducción a Objetos
  - Introducción a la Programación Orientada a Objetos
- Sitios:
  - http://docs.oracle.com/javase/tutorial/index.html
- Bibliografía disponible en biblioteca:
  - An introduction to object-oriented programming.
  - Análisis y diseño orientado a objetos con aplicaciones.
  - Fundamentos del diseño y la programación orientada a objetos.
  - Head first object-oriented analysis and design.
  - The java programming language.
- Software de desarrollo:
  - Netbeans
  - Eclipse.

3. Actividades

Ejercicio 1

Un pedido está representado por un número, una fecha de entrega, un cliente y una colección de artículos cada uno con cantidad y precio unitario. Los pedidos son cancelados por una serie de remitos que contienen uno o varios artículos del pedido. El pedido puede estar abierto (se le pueden agregar artículos, pero no remitos), cerrado (se le pueden agregar remitos, no artículos) o entregado (no se le pueden agregar artículos ni remitos).

a. Realice un modelo de clases que permita representar estos requerimientos adecuadamente. Piense patrones de diseño que puedan ayudarlo y describa por qué.
b. Escriba en Java los métodos para agregar artículos y remitos a un pedido.
c. Modele ahora la opción de reabrir un pedido cerrado. ¿Qué modificaciones debe hacer en el modelo de objetos?

Ejercicio 2

Un Pool de Impresoras es un objeto que administra un conjunto de impresoras. Este pool es único en cada aplicación y debe asegurarse que no se pueda crear más de una instancia del mismo. Siempre que se solicite el pool de impresoras debe responderse enviando el mismo objeto.
El Pool de impresoras sabe responder a los siguientes métodos:
public Impresora obtenerImpresora(String nombre) que retorna un objeto Impresora.
public void liberarImpresora(Impresora impresora) que libera la impresora antes solicitada.

a. Realice un modelo de clases que permita representar estos requerimientos adecuadamente. Piense patrones de diseño que puedan ayudar y describa por qué.
b. Escriba en Java el código de los métodos para instanciar el Pool de Impresoras y el código de los métodos obtenerImpresora(String nombre) y liberarImpresora(Impresora impresora).
c. ¿A qué clase o instancia debe enviarse el mensaje para obtener el Pool de Impresoras?

Ejercicio 3

Se quiere realizar un software para administración de venta de chocolates de una fábrica. La fábrica vende cajas de diferentes unidades con un precio unitario por caja. Además la fábrica propone bolsas promocionales compuestos por varias cajas de diferentes productos aplicando un 10% de descuento al precio total.

a. Realice un modelo de clases que permita representar estos requerimientos adecuadamente. Piense qué patrón de diseño puede ayudarle y describa por qué.
b. Escriba el método precio() de una caja y de una bolsa.

Ejercicio 4

Se quiere modelar la forma de pago de una venta de productos a través de una aplicación web.
La forma de pago puede ser transferencia bancaria, pago por Internet (PayPal) o con tarjeta de crédito. Para cada caso se dispone de un procedimiento diferente: transferencia bancaria requiere de CBU, pago por Internet requiere nombre de usuario

de Paypal y para tarjeta de crédito se requiere los datos de nombre, número y los 3 dígitos de seguridad de la tarjeta.
Para ello existe en el modelo la siguiente funcionalidad:

- Banco.transferir(cbuorigen, importe)
- Tarjeta.cobrar(nombreTarjeta, numero, digitos, importe)
- PayPal.cobrar(usuario,importe)

a. Realice un modelo de clases que permita representar estos requerimientos adecuadamente. Piense qué patrón de diseño puede ayudarle y describa por qué.
b. Escriba el método pagar() de la venta y haga un bosquejo en pseudocódigo de cómo se resolvería en cada caso.
c. Realice ahora una adaptación al modelo de objetos considerando la posibilidad de que la venta puede incluir más de una forma de pago.