Universidad Nacional del Sur
Departamento de Ciencias e Ingeniería de la Computación

Tecnología de Programación

(Regular)

Ejercicio 1: Algunos desarrolladores sugieren que la operación getInstance() del patrón de diseño Singleton puede requerir ser una operación sincronizada. Explique claramente por qué. Describa una situación donde la no sincronización del método resulta en un problema.

Ejercicio 2: Teniendo en cuenta la siguiente descripción:
Una empresa de viajes ofrece viajes turísticos personalizados. Cada viaje posee un identificador alfanumérico, una descripción de texto, el nombre del pasajero y un costo en dólares, al que se pueden agregar adicionales a gusto del cliente. Por ejemplo, se pueden adicionar paseos o visitas guiadas, cenas o almuerzos, alquiler de vehículos, etc. Por supuesto, cada uno de estos adicionales incrementa el costo del viaje completo en una cierta cantidad. Cada adicional posee también un código alfanumérico y el nombre del pasajero. Un paquete turístico es un viaje turístico con o sin adicionales. El identificador de todo el paquete es la concatenación de todos los identificadores de los adicionales (si los hay) y el paquete.

¿Qué patrón de diseño considera útil para esta situación? Implemente en Java el esquema anterior, utilizando el patrón que considere apropiado. Incluya el diseño UML completo. No implemente getters o setters triviales pero indique la interfaz completa de cada clase. Indique cualquier suposición o interpretación que considere ambigua o incompleta en el enunciado.

Ejercicio 3: Teniendo en cuenta la siguiente descripción
La empresa registra todos los paquetes turísticos vendidos y tiene algunos puntos de venta en todo el país. Estas sucursales sólo venden ciertos paquetes turísticos pre-armados, que se venden con mucha frecuencia. Por ejemplo, son comunes los cruceros a Brasil con algunas cenas incluidas, o viajes a Bariloche con excursiones en barco. Las sucursales ofrecen sólo tres paquetes diferentes, identificados como Lujo, Premium o Económico, cuya estructura es determinada por la empresa, según lo que gusta a los clientes en cada zona. Cuando un cliente llega a la sucursal y adquiere un paquete, la sucursal únicamente debe registrar el nombre del pasajero y enviar la copia del paquete turístico deseado a la empresa para que la registre.

¿Qué patrón de diseño considera útil para esta situación? Implemente en Java el esquema anterior, utilizando el patrón que considere apropiado. Incluya el diseño UML completo. No implemente getters o setters triviales pero indique la interfaz completa de cada clase. Indique cualquier suposición o interpretación que considere ambigua o incompleta en el enunciado.

Ejercicio 4: Explique brevemente el principio de Inversión de Dependencias. Muestre además, un ejemplo en donde el principio NO se cumpla e indique una posible solución.

Ejercicio 5: Explique en qué consiste el refactoring y mencione al menos dos casos de refactoring que considere interesantes.

Ejercicio 6: ¿Cuáles son los tipos de herencia según Meyer?

Ejercicio 7: Explique claramente la relación entre el principio de sustitución de Liskov y el diseño por contrato.

Ejercicio 8: Explicar “Herencia por conveniencia”.

Ejercicio 9: Explique qué alternativas pueden ofrecen los lenguajes de programación en el tratamiento de las colisiones de nombre.