UNIVERSIDAD
NACIONAL
DE COLOMBIA
SEDE BOGOTÁ
FACULTAD DE INGENIERÍA

Taller Teórico
Programación Orientada a Aspectos

Neill Rolando Giraldo Corredor
Iván Darío Vanegas Pérez

Responda a las siguientes Preguntas:

1. Una incumbencia transversal o 'crosscutting concern' es:

    a. Un lenguaje de programación
    b. Una legacy app
    c. La conceptualización de responsabilidades de uso común en un sistema
    d. La implementación de códigos que se ejecutan a través del flujo de ejecución de los requerimientos funcionales básicos de una aplicación.

2. Algunos problemas considerados a lo largo de la historia del diseño de software que llevan a pensar en éste nuevo paradigma son:

    a. Redundancia de código
    b. Baja escalabilidad del sistema
    c. Código enredado y/o diseminado
    d. Código transversal

3. Algunos ejemplos de Aspectos de un software pueden ser

    a. Trazabilidad de flujo de ejecución de un programa
    b. Mantenimiento, escalabilidad, cohesión
    c. Transacciones, logging, reusabilidad
    d. Sincronización, logging, transacciones, seguridad

4. Un modelo de JoinPoints contiene la cantidad exacta de CutPoints que van dentro de los atributos de los objetos de la funcionalidad básica y que son iguales a los llamados a sus métodos:

    a. Verdadero
    b. Falso

5. La programación orientada a aspectos nos permite alterar el comportamiento de las entidades y los módulos del sistema en tiempo de ejecución

    a. Verdadero
    b. Falso

6. Los tejedores en AspectJ solo se programan a través de etiquetas de XML

    a. Verdadero
    b. Falso

7. El protocolo de metaobjetos, la programación adaptativa, y la reflexión son:

    a. Tecnologías fundamentales de algunos paradigmas de programación tales como POO y POA
    b. Los pilares de la programación orientada a aspectos
    c. Aplicaciones desarrolladas en Java bajo el paradigma de POO en conjunto con POA
    d. Ninguna de las anteriores

Considere el siguiente código para responder las preguntas de la 7 a la 10

```
public class SomeBusinessClass extends OtherBusinessClass {
   // Core data members
   // Other data members: Log stream, data-consistency flag
   // Override methods in the base class
   public int per_joinPoint1 = 0;
   public int attr_joinPoint2 = 1;

   public void performSomeOperation(OperationInformation info) {
      // Ensure authentication
      // Ensure info satisfies contracts
      // Lock the object to ensure data-consistency in case other
      // threads access it
      // Ensure the cache is up to date
      // Log the start of operation
```

```
        // ==== Perform the core operation ====
        // Log the completion of operation
        // Unlock the object
    }
    public Object performOtherOperation(OperationInformation info) {
      return new Object();
    }
    // More operations similar to above
    public void save(PersitanceStorage ps) {
    }
    public void load(PersitanceStorage ps) {
    }
}
```

8. La cantidad de JointPoints presentes en el programa son
    a. 2
    b. 4
    c. 0
    d. Ninguno ya que los joinpoint solo se pueden recrear en tiempo de ejecución

9. Una posible etiqueta para realizar un match desde un pointcut de un aspecto del programa es:

    a. `@Pointcut("execution(* SomeBussinessClass.performSomeOperation(String))") public void performSomeOperation() { }`
    b. `@After("somePointCut()") public void performSomeOperation() { //Do what you want to do before the joint point is executed }`
    c. `@Pointcut(return new Object());`
    d. Ninguna de las anteriores

10. Se puede inferir que

    a. El código no representa un programa bajo el paradigma POA ya que carece de tejedores.
    b. El código muestra un ejemplo clásico de lo que sería una funcionalidad orientada a objetos en la cual se limita el control de responsabilidad transversales al programa como son: Hilos, autenticación y Log de actividades.
    c. El código representa un lenguaje estructural
    d. Ninguna de las anteriores