Simulacro de Coloquio Final

Ejercicio 1: Programación Orientada a Objetos (POO)

Dado el siguiente programa:

```java
abstract class Empleado {
    private String nombre;
    private double salario;

    public Empleado(String nombre, double salario) {
        this.nombre = nombre;
        this.salario = salario;
    }

    public String getNombre() {
        return nombre;
    }

    public double getSalario() {
        return salario;
    }

    public abstract double calcularBonificacion();
}

class EmpleadoTiempoCompleto extends Empleado {
    public EmpleadoTiempoCompleto(String nombre, double salario) {
        super(nombre, salario);
    }

    @Override
    public double calcularBonificacion() {
        return getSalario() * 0.1;
    }
}
```

```java
class EmpleadoMedioTiempo extends Empleado {
    public EmpleadoMedioTiempo(String nombre, double salario) {
        super(nombre, salario);
    }

    @Override
    public double calcularBonificacion() {
        return getSalario() * 0.05;
    }
}

public class Main {
    public static void main(String[] args) {
        Empleado e1 = new EmpleadoTiempoCompleto("Alice", 50000);
        Empleado e2 = new EmpleadoMedioTiempo("Bob", 20000);

        System.out.println(e1.getNombre() + " tiene una bonificación de " + e1.calcularBonificacion());
        System.out.println(e2.getNombre() + " tiene una bonificación de " + e2.calcularBonificacion());
    }
}
```

Ejercicio 1.a) Se tiene un programa que recibe desde el main dos tareas empleado 1 y 2, a los que por parámetros se les pasa un nombre y un salario. Desde la clase empleado, se definen y retornan los valores para nombre y salario, que componen a la clase, y a su vez esta se extiende a dos clases más (siendo empleado la superclase), donde en base al sueldo que se les pase y el estatus al que se corresponda (si es un empleado de medio tiempo o de tiempo completo), se le calculará su incremento. De esta manera, Alice (empleado1) recibirá en su sueldo un incremento de $5000, mientras que Bob (empleado2) recibirá uno de $1000.

b) No, no se viola open/closed ya que el programa esta abierta a la extensión del código y sus cálculos, pero no se esta modificando código existente, ya que empleado es una superclase que se extiende al estatus de cada uno, y en base a la información recibida por cada uno, el incremento se esta realizando con el metodo calcularBonificacion, abierto a la extensión de cada nueva clase. Aun asi, la justificación es repetitiva y poco técnica, omitiendo términos clave como "polimorfismo" o "sobrescritura de métodos". Ademas, el diseño podría mejorarse aún más (por ejemplo, utilizando interfaces en lugar de una clase abstracta).

Ejercicio 2: Programación Orientada a Eventos (POE)

Dado el siguiente programa:

```java
import java.util.Timer;
import java.util.TimerTask;

public class Main {
    public static void main(String[] args) {
        Timer timer = new Timer();
        TimerTask tarea = new TimerTask() {
            int contador = 0;

            @Override
            public void run() {
                contador++;
                System.out.println("Tarea ejecutada " + contador + " veces");
                if (contador == 5) {
                    System.out.println("Tarea completada");
                    timer.cancel();
                }
            }
        };
        timer.scheduleAtFixedRate(tarea, 0, 1000);
    }
}
```

a) Explique el funcionamiento del programa y su salida al ejecutarlo.
b) Determine la duración total de la ejecución.

Ejercicio 2.a) Se tiene un programa en java que contiene una clase principal main, a la cual se les esta implementando un temporizador, y una clase anónima (por el uso de azúcar sintáctica en TimerTask), donde se implementa al método run. Este timer programa la ejecución del run, con una frecuencia de a un segundo, y respecto a la funcionalidad del método run, esta evalúa, incrementa y detiene el temporizador de la tarea, y devuelve la cantidad de veces ejecutada, hasta que llegue a un tiempo de 5.

b) Como bien dijimos, Se inicia, evalúa, incrementa y se detiene un contador inicia en timer, que actualiza las tareas cada un segundo, y se propone un límite de 5 repeticiones/actualizaciones de las tareas, de manera automática y cada un segundo, por lo que con esto, podemos suponer que se finalizará la ejecución al cumplir este tiempo, y mostrará el incremento final de la tarea, junto al mensaje de tarea completada.

Ejercicio 3: Principios de Diseño

Analice el siguiente código y conteste las preguntas:

```
class Calculadora {
    public int sumar(int a, int b) {
        return a + b;
    }

    public int restar(int a, int b) {
        return a - b;
    }

    public String concatenar(String a, String b) {
        return a + b;
    }
}
```

a) Identifique si este diseño viola algún principio de diseño. Justifique.
b) Proponga una solución para mejorar el diseño.

Ejercicio 3.a) Si, se violan principios de diseño como el DRY (Don't Repeat Yourself), ya que se esta de manera innecesaria repitiendo un método de suma. Además en consecuencia se violan SOC y SRP, ya que cada clase debe tener su enfoque en una funcionalidad diferente, y no repetir.

b)Una solución adecuada sería separar las responsabilidades en diferentes clases (una para cálculos matemáticos y otra para manipulación de cadenas).

Ejercicio 4: Programación Concurrente en Java

Dado el siguiente código en Clojure:

```java
public class Main {
    public static void main(String[] args) {
        Contador contador = new Contador();

        Thread hilo1 = new Thread(() -> {
            for (int i = 0; i < 5; i++) {
                contador.incrementar();
            }
        });

        Thread hilo2 = new Thread(() -> {
            for (int i = 0; i < 5; i++) {
                contador.incrementar();
            }
        });

        hilo1.start();
        hilo2.start();
    }
}
```

```java
class Contador {
    private int valor = 0;

    public synchronized void incrementar() {
        valor++;
        System.out.println("Valor: " + valor);
    }
}
```

a) Analice el funcionamiento del programa y describa su salida esperada.
b) ¿Qué ocurre si se elimina la palabra clave synchronized del método incrementar? Justifique.

Ejercicio 4.a) Se tiene una clase main que recibe dos tareas de manera concurrente y en hilos separados. Estas tareas lo que hace es implementar un contador, que previamente se inicializa con un valor definido en 0, los cuales son evaluados en los threads, y en base a esta verificación, incrementa el valor definido hasta el límite establecido de 5 incrementos

en un bucle fijo en cada tarea. De esta manera, ambas tareas empezaran con un incremento hasta su final, y se mostrará finalizada la ejecución, el valor al cual llegaron.

b) Sin synchronized, ambos hilos podrían acceder al método incrementar al mismo tiempo, causando condiciones de carrera (race conditions) y resultados impredecibles en el valor del contador. La sincronización es fundamental para garantizar que cada incremento se realice de manera atómica.