Programación orientada a objetos. Examen julio 2004.

Viernes 30 de julio de 2004.

Pongan el nombre sólo en la primera hoja. Contesten las preguntas en orden en que están formuladas. Escriban sólo en los anversos de las hojas y eviten doblarlas. Todas las preguntas valen lo mismo. Gracias y mucha suerte.

1. Digan cómo funcionan el **encadenamiento estático** y el **encadenamiento dinámico**. En un lenguaje que soporta tanto encadenamiento estático como dinámico ¿qué cosas malas o negativas podrían llegar a ocurrir si se perdiera el encadenamiento dinámico? En un lenguaje que soporta sólo encadenamiento dinámico ¿qué cosas buenas o positivas podría llegar a tener si se le agregara encadenamiento estático?

   Sean las clases siguientes:

```
/* Ancestro */
public class Ancestro {
  void metodo() {
    System.out.println("Soy el ancestro");
  }
}

/* Sucesor */
public class Sucesor {
  public void metodo() {
    System.out.println("Soy el sucesor");
  }
}

/* Simple */
public interface Simple {
  public void metodo();
}

/* Programa */
public class Programa {
  public static void main(String[] args) {
    Sucesor o;
    o = new Sucesor();
    o.metodo();
  }
}
```

   La clase `Programa` compila y es correcto. ¿Qué se imprime en la consola? Asuman por un instante que Java usara encadenamiento estático para los mensajes enviados a los objetos. ¿Qué se imprime en la consola?

   Si contestaron que en ambos casos en la consola se imprime lo mismo, hagan la menor cantidad de modificaciones -borrar, agregar o cambiar líneas de código- posible, para que en la consola se imprima algo diferente con encadenamiento estático que con encadenamiento dinámico.

   Si contestaron que en ambos casos en la consola se imprime algo diferente, hagan la menor cantidad de modificaciones posible, para que la consola imprima lo mismo con encadenamiento estático que con encadenamiento dinámico.

2. Definan **polimorfismo**. Describan el **principio de sustitución**. ¿Qué relación hay entre el polimorfismo y el principio de sustitución?

   ¿Tiene sentido hablar de polimorfismo y de principio de sustitución cuando los tipos son definidos por clases? Si contestaron afirmativamente, hagan la menor cantidad de modificaciones posibles en las clases del primer ejercicio, para mostrar claramente un ejemplo de polimorfismo y principio de sustitución usando clases para definir los tipos.

   ¿Tiene sentido hablar de polimorfismo y de principio de sustitución cuando los tipos son definidos con interfaces cuando son definidos por interfaces? Usando las clases del primer ejercicio, hagan la menor cantidad de modificaciones posibles en las clases del primer ejercicio, para mostrar

claramente un ejemplo de polimorfismo y principio de sustitución usando interfaces para definir los tipos.

3. Sea una interfaz `Printable` y una clase `Printer` definidas así:

```
interface Printable {
  void printOn(PrintStream s);
}

class Printer {
  public void printOn(Object o, PrintStream s) {
    s.println(o);
  }
}
```

El método `printOn(Stream s)` imprime el receptor en la `PrintStream s`. Se pide a dos programadores que implementen una nueva clase `Cualquiera` que implementa la interfaz `Printable`. Hay más de una forma de hacerlo. Impleméntenlas usando las clases anteriores y discutan las ventajas de cada una.

4. Un objeto puede tener más de un tipo y varios objetos de diferentes clases pueden tener el mismo tipo. Hagan la menor cantidad de modificaciones posibles en las clases del primer ejercicio, para tener:
   a. Un objeto con más de un tipo;
   b. Varios objetos de diferentes clases y el mismo tipo, declarando los tipos mediante clases;
   c. Varios objetos de diferentes clases y el mismo tipo, declarando los tipos mediante interfaces.

   Es importante que digan los nombres de las variables que contienen o hace referencia a dichos objetos.

5. Sea la siguiente clase:

```
public class ReproductorCD {
  private boolean encendido = false;
  private boolean reproduciendo = false;
  public boolean isEncendido() {
    return encendido;
  }
  public boolean isReproduciendo() {
    return reproduciendo;
  }
  public void setEncendido(boolean b) {
    encendido = b;
  }
  public void reproducir() {
    reproduciendo = true;
  }
  public void detener() {
    reproduciendo = false;
  }
}

public class Programa {
  public static void main(String[] args) {
    ReproductorCD r = new ReproductorCD();
    r.setEncendido(true);
    r.reproducir();
    r.detener();
    r.setEncendido(false);
  }
}
```

Sean las afirmaciones: “el reproductor debe estar encendido mientras está reproduciendo”, “no se puede encender el reproductor cuando ya está encendido” y “no se puede apagar el reproductor cuando ya está apagado”. Decidan si estas afirmaciones son precondiciones, poscondiciones o invariantes. Prográmenlas en Java indicando, además, dónde las agregarían -en qué método y si van al comienzo o al final-.

6. a. Hagan la menor cantidad de modificaciones posibles en las clases del ejercicio anterior, para que sea el cliente quien viola la invariante, precondición o poscondición.

   b. Hagan la menor cantidad de modificaciones posibles en las clases del ejercicio anterior, para que sea el servidor quien viola la invariante, precondición o poscondición.

   Indiquen claramente cuál es la afirmación que falla y en qué método de qué clase falla.

7. Sean las siguientes clases:

```
public class UnosDedos {
  public void índice() {
    System.out.println("Índice");
  }
  public void pulgar() {
    System.out.println("Pulgar");
  }
}

public class OtrosDedos {
  public void anular(){
    System.out.println("Anular");
  }
  public void mayor() {
    System.out.println("Mayor");
  }
  public void meñique() {
    System.out.println("Meñique");
  }
}

public class Programa {
  public static void main(String[] args) {
    Mano m = new Dedos();
    m.anular();
    m.índice();
    m.mayor();
    m.meñique();
    m.pulgar();
  }
}
```

   La clase `Programa` no compila pues no están definidos `Mano` ni `Dedos`. Hagan lo que sea necesario para que el programa compile e imprima los nombres de los cinco dedos en la consola. Si hay más de una solución posible impleméntenlas todas.