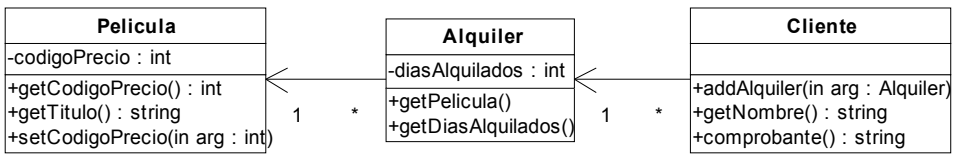
Programación orientada a objetos. Examen diciembre 2003.

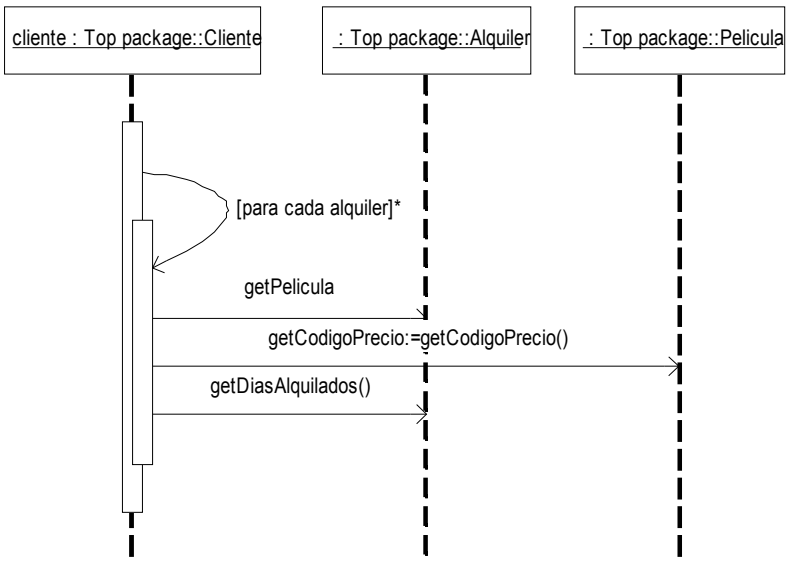
Martes 2 de diciembre de 2003.

Pongan el nombre sólo en la primera hoja. Escriban sólo del anverso de la hoja. Contesten las preguntas en orden en que están formuladas. Eviten doblar las hojas. Gracias y mucha suerte.

```
/* Pelicula.java */
public class Pelicula {
  public static final int INFANTIL = 2;
  public static final int REGULAR = 0;
  public static final int ESTRENO = 1;

  public String m_titulo;
  private int m_codigoPrecio;

  public Pelicula(String titulo, int codigoPrecio) {
       m_titulo = titulo;
       m_codigoPrecio = codigoPrecio;
  }

  public int getCodigoPrecio() {
       return m_codigoPrecio;
  }

  public String getTitulo() {
       return m_titulo;
  }

  public void setCodigoPrecio(int arg) {
       m_codigoPrecio = arg;
  }
}

/* Alquiler.java */
public class Alquiler {
  private Pelicula m_pelicula;
  private int m_diasAlquilados;

  public Alquiler(Pelicula pelicula, int diasAlquilados) {
    m_pelicula = pelicula;
    m_diasAlquilados = diasAlquilados;
  }

  public int getDiasAlquilados() {
    return m_diasAlquilados;
  }

  public Pelicula getPelicula() {
    return m_pelicula;
  }
}

/* Cliente.java */
import java.util.Enumeration;
import java.util.Vector;

public class Cliente {
  private String m_nombre;
  private Vector m_alquileres = new Vector();

  public Cliente(String nombre) {
    m_nombre = nombre;
  }

  public void addAlquiler(Alquiler arg) {
    m_alquileres.addElement(arg);
  }
```

```
  public String getNombre() {
    return m_nombre;
  }

  public String comprobante() {
    double importeTotal = 0;
    int puntosAlquilerFrecuente = 0;
    Enumeration alquileres = m_alquileres.elements();
    String result = "Registro de alquiler " + getNombre() + "\n";
    while (alquileres.hasMoreElements()) {
      double importe = 0;
      Alquiler alquiler = (Alquiler)alquileres.nextElement();

      // Determinar importe de cada línea
      switch(alquiler.getPelicula().getCodigoPrecio()) {
        case Pelicula.REGULAR:
          importe =+ 2;
          if (alquiler.getDiasAlquilados() > 2)
            importe += (alquiler.getDiasAlquilados() - 2) * 1.5;
           break;
        case Pelicula.ESTRENO:
          importe += alquiler.getDiasAlquilados() * 3;
          break;
        case Pelicula.INFANTIL:
          importe += 1.5;
          if (alquiler.getDiasAlquilados() > 3)
            importe += (alquiler.getDiasAlquilados() - 3) * 1.5;
          break;
      }

      // Agregar puntos de alquiler frecuente
      puntosAlquilerFrecuente++;

      // Agregar boons por alquiler de estrenos de dos días
      if ((alquiler.getPelicula().getCodigoPrecio() ==
        Pelicula.ESTRENO) && alquiler.getDiasAlquilados() > 1)
        puntosAlquilerFrecuente++;

      // Mostrar importes de este alquiler
      result += "\t" + alquiler.getPelicula().getTitulo() + "\t" +
        String.valueOf(importe) + "\n";
      importeTotal += importe;
    }
    // Agregar líneas al pie
    result += "Total es " + String.valueOf(importeTotal) + "\n";
    result += "Usted ganó " + String.valueOf(puntosAlquilerFrecuente) +
      " puntos de alquiler frecuente";
    return result;
  }
}
```





Este programa es muy simple. Es un programa que calcula e imprime un comprobante para un cliente de un videoclub. Al programa se le indica las películas que alquila un cliente y por cuanto tiempo. El programa calcula el importe dependiendo del tiempo por el que es alquilada la película e identifica el tipo de película. Hay tres tipos de película: regulares, de niños y estrenos. Además de calcular el importe, el comprobante también calcula los puntos de alquiler frecuente, que varían dependiendo de si la película es un estreno.

El programa es correcto; cumple con los requisitos. Compila y funciona.

1. Critique este programa. Justifique su respuesta.

20 puntos.

2. Indique cómo resolvería cada problema encontrado. Justifique su respuesta. Escriba nuevamente el programa.

20 puntos.

```
/* Perro.java */
public class Perro {
  public static int PERRO = 1;
  public static int PERRA = 2;
  private String m_nombre;
  private int m_sexo;

  public Perro(…) {…}
  public String getNombre() {…}
  public void setNombre(…) {…}
  public int getSexo() {…}
  public Cruza cruzar(…) {…}
}


/* Cruza.java */
import java.util.*;

public class Cruza {
  private Perro m_perro;
  private Perro m_perra;
  private Collection m_prole = new ArrayList();
  public Cruza(Perro perro, Perro perra) {…}
  public void addCachorro(Perro cachorro) {
    m_prole.add(cachorro);
  }
  public Iterator getProle() {
    return m_prole.iterator();
  }
}
```

3. Sean las clases `Perro` y `Cruza` para registrar las cruzas entre perros de raza. Completen los métodos que dicen "…". Tengan en cuenta que:

   - Un perro no puede cambiar de raza.
   - Un perro no puede cambiar de sexo.
   - No se puede cambiar el perro y la perra de una cruza.
   - No se puede quitar cachorros de una cruza.

   10 puntos

4. Definan en forma breve y precisa **precondición** y **poscondición**. Agreguen a las clases `Perro` y `Cruza` del ejercicio anterior las precondiciones y poscondiciones que consideren necesarias para asegurar que:

   - Un perro no se puede cruzar consigo mismo.
   - Un perro se debe cruzar con otro de su misma raza.
   - Un pero no se puede cruzar con otro del mismo sexo.
   - Un perro no se puede cruzar con sus padres ni con ninguno de sus hermanos.
   - Una cruza no puede cambiar los perros que intervinieron en ella.

   Agreguen estas afirmaciones al código del ejercicio anterior usando la cláusula **`assert`** de Java. No tienen porqué escribir nuevamente toda la clase, mientras indiquen claramente dónde se inserta cada modificación. Pueden agregar otros métodos además de los que ya existen.

   10 puntos

5. Programen un caso de prueba `TestPerro` en JUnit para la clase `Perro`.

   10 puntos

```
public class Animal {
  public void comer() {
  …
  };
}
```

6. Usando las clases `Perro` y `Animal` anteriores creen una nueva clase que tenga los métodos de ambas simultáneamente. Muestren todas las formas diferentes de hacerlo si hubiera. No pueden escribir los métodos; deben usar los que ya están implementados.

   Hagan un programa en Java en el que crean un objeto al que le envían el mensaje `comer()` y el mensaje `cruzar()`.

   10 puntos

```
public class Carnivoro extends Animal {
  public void comer() {
  …
  };
}
```

7. Expliquen en forma clara y precisa cómo funciona el **encadenamiento estático** y el **encadenamiento dinámico** en esos lenguajes. ¿El concepto es aplicable a los lenguajes en los que no se declara el tipo de las variables y argumentos?

   Muestren las diferencias entre encadenamiento estático y dinámico, con un pequeño programa en Java usando las clases provistas, suponiendo sólo por un instante que Java soportara encadenamiento estático.

   10 puntos

8. Un objeto puede tener más de un **tipo** y varios objetos de diferentes clases pueden tener el mismo **tipo**.

   Muestren con un pequeño programa en Java, usando las clases provistas, un ejemplo de un objeto con más de un tipo y otro con varios objetos de diferentes clases y el mismo tipo.

10 puntos