PROGRAMACIÓN ORIENTADA A OBJETOS **14/02/2006**

NOMBRE:_____**Titulación:**_____

1. (**0'5 ptos**) Dadas las clases A, B y C, cuyas declaraciones expresadas en Eiffel aparecen abajo, indica qué versión se ejecuta para cada uno de los mensajes que aparecen a la derecha de las declaraciones de las clases. (Utiliza letras griegas para referirte a las distintas implementaciones de rutinas implicadas). En el caso de que se produzca un error indica el motivo.

clases A **feature**
  f is do … **end;**
  …
end;
 	
clases B **feature**
  f is do … **end;**
  …
end;
 	
class C **inherit**
  A
  rename f **as** g;
  export {NONE} g
  A
  rename f as h
  redefine h;
  select h
  B
  redefine f;
feature
  f **is do** .. **end;**
  h **is do** .. **end;**
End
	oa: A; ob: B; oc: C;
	  !!oa; !!ob; !!oc;
	
	  oa.f;
	  ob.f
	  oc.f;
	  oc.g;
	  oc.h
	
	oa:= oc;
	
	  oa.f;
	  oa.g;
	
	ob:= oc;
	
	  ob.f;
	  ob.h;	

2. Se la ***clase genérica*** **`Caja`,** implementada en Eiffel, la encarga de modelar una caja que almacena una lista objetos de valor (joyas, documentos, fotos, etc.).

```
class Caja[G] feature
      contenido: LIST [G]
      …
end
```

a) (**0'5 ptos**) Implementa el método **`max`** en la clase `Caja` que devuelve el objeto de más valor de los almacenados en la caja. Modifica la implementación propuesta de la clase `Caja` si lo consideras oportuno. (Puedes suponer la existencia de un método `getValor` que devuelve el valor de un objeto de valor).

b) (**0'5 ptos**) Supuesta la implementación de la clase `Caja` en Java y siendo `LinkedList` el tipo de la variable de instancia `contenido`, compara esta implementación con la propuesta en Eiffel desde el punto de vista de la genericidad.

3. Suponga la implementación de la clase `Caja` en el contexto de una aplicación bancaria donde existe una clase `Cliente` que modela el cliente del banco que tiene alquilada una caja de seguridad (clase `CajaSeguridad` subclase de `Caja`).

a) (**0'75 ptos**) ¿Cuál sería el nivel de visibilidad del atributo `contenido` en Eiffel para que sólo la clase `Cliente` tenga acceso total a dicho atributo? ¿y en Java?

b) (**0'75 ptos**) ¿Cuál sería el nivel de visibilidad del atributo `contenido` en Eiffel para que, además de la propia clase `Caja`, sólo la clase `CajaSeguridad` tenga acceso total a dicho atributo? ¿y en Java?

4. (**1 pto**) Al implementar la clase `Caja` en Java se ha decidido que herede de `LinkedList` ¿Sería posible ocultar algunos de los métodos públicos de la clase `LinkedList` para los clientes de la clase `Caja`? Compara la solución propuesta para Java con la que darías para C++.

5. Un **`Predicado`** evalúa un objeto y devuelve un valor de tipo `boolean` como resultado de dicha evaluación. Por su parte, la clase **`java.util.Collections`** consiste exclusivamente en métodos de clase que operan sobre colecciones (**`java.util.Collection`**, al final del examen puedes consultar los métodos definidos en esta interfaz). Se pide:

a) **(0'5 ptos)** Implementar un nuevo método para la clase `Collections` denominado **`retain`** encargado de mantener en la colección que se le pasa como parámetro sólo aquellos elementos para los que el predicado, que también se le pasa como parámetro, devuelve un valor verdadero.

b) **(0'5 ptos) ¿**El método `retain` del apartado a) es un ejemplo de código genérico? Razona la respuesta.

c) **(0'5 ptos)** Sea **`LinkedList alumnos;`** una variable que referencia a la lista de todos los alumnos matriculados en POO. Utilizar el método `retain` para mantener en la lista sólo los alumnos suspensos. Especifica todo lo que habría que hacer.

d) **(0'5 ptos)** A partir de la definición de la clase Eiffel **`LINEAR_ITERATOR`** vista en clase implementa el iterador **`ItRetieneSuspensos`** que itera sobre la colección de alumnos borrando aquellos que estén aprobados. De manera que el método **`retieneSuspensos`** quedaría:

```
retieneSuspensos(alumnosPOO:LIST[Alumno]) is
local
      it: ItRetieneSuspensos
do
      !!it.make(alumnosPOO);
      it.doIf
end
```

NOTA: En la clase **`Alumno`** existe un método **`getNota`** que devuelve la nota de POO.

6. La empresa de embutidos MorciMur S.A. ha desarrollado un software en Java para PDA de manera que los visitadores de las granjas porcinas puedan enviar los datos directamente a la sede central. Sea el método **`sendToCentral`** el encargado de dicho envío, que a su vez invoca al método **`sentTo`** de la clase `Granja` que abrirá la conexión vía satélite de acuerdo a la dirección que se le pasa como parámetro y transmitirá los datos recogidos. Esto es:

```
public void sentToCentral(){
     granjaActual.sentTo(dirMorciMur);
}
```

a) (**1 ptos**) El método `sentTo` lanzará una excepción en el caso de que no se haya podido abrir la conexión o se interrumpa la transmisión. ¿De qué tipo será dicha excepción? Razona la respuesta. Explica todas las formas posibles de modificar el código propuesto para manejar la excepción.

b) (**0'5 pto**s) El método `sentToCentral` debe reintentar el envío al menos 5 veces, en el caso de no éxito debe notificarlo al cliente. Modifica el código propuesto para que se adapte a esta circunstancia.

7. (**1 pto**) En una empresa existen dos tipos de empleados: los contratados por horas y los empleados fijos. En el primer caso el sueldo se calcula como el producto del número de horas por el precio de la hora y en el segundo caso los empleados tienen un sueldo fijo mas comisiones. Utiliza el patrón de diseño del ***Método Plantilla*** para implementar el método `getNomina` que devuelva una cadena de texto con la información de la nómina que tiene que percibir el empleado. En la cabecera de la nomina se debe imprimir el tipo de contrato del empleado

("Empleado temporal por horas" o "Empleado fijo") junto con el nombre y el DNI. A continuación el detalle de su sueldo. Por ejemplo, en el empleado por horas se deberá imprimir el número de horas trabajadas, el precio de la hora y el total que va a percibir ("20 horas x 9 euros/hora = 180 euros"). Especifica e implementa todo lo que sea necesario para resolver el ejercicio (jerarquía de herencia, atributos y métodos de cada clase).

8. En el club deportivo FitnessMur se pueden contratar cada una de las actividades que ofertan por separado (Gimnasio, Natación, Tenis, etc.), cada una con una tarifa diferente, o se puede contratar un Bono. Un Bono es un tipo de contrato que incluye otros contratos individuales o incluso otros bonos. La tarifa en el caso de los bonos se calculan como la suma de la tarifa de todos los contratos que lo componen menos un 10% de descuento.

a) (**1 pto**) Especifica la jerarquía de contratos que modele el problema planteado e implemente el método `getTarifa` en la clase que represente los bonos del club.

b) (**0'5 ptos**) Implementa en Eiffel un método `imprimeSocorristas` que reciba como argumento la lista de los posibles contratos e imprima los monitores de natación. (Suponga que la clase que representa la actividad de natación cuenta con un método `getMonitores` que devuelve una cadena de texto con los nombres de los monitores)

NOTA:
```
interface java.util.Collection{
        Iterator iterator();
        int size();
        boolean isEmpty();
        boolean contains(Object obj);
        boolean containsAll(Collection c);
        boolean add(Object obj);
        boolean addAll(Collection c);
        boolean remove(Object obj);
        boolean removeAll(Colecction c);
}
```