20 de diciembre de 2012

PROGRAMACIÓN II **(Duración del examen: 2 horas)**

NOMBRE:

Cuestión 1. **(1 punto)**

Indicar si las siguientes afirmaciones son verdaderas o falsas. Es imprescindible justificar adecuadamente el porqué de la decisión.

- *"Una clase abstracta puede no tener ningún método abstracto"*

Respuesta:

- *"Al pasar un objeto como parámetro de entrada a un método todos los cambios que se efectúen en sus atributos se mantienen cuando termina la ejecución de dicho método"*

Respuesta:

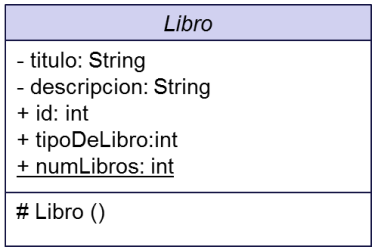
- *"Un objeto de una clase A que hereda de otra clase B siempre puede acceder a todos los atributos heredados de B."*

Respuesta:

- *"Al igual que es posible que distintas clases implementen una única interfaz, una única clase puede implementar dos o más interfaces."*

Respuesta:

Cuestión 2. **(0.5 puntos)**



Implementar la clase cuya representación UML se adjunta en la figura de la izquierda.

Respuesta:

Cuestión 3. **(0.5 puntos)**

Dado el código siguiente:

```
public class Mamifero {
	private String habitat;

	public Mamifero (String h){
		habitat=h;
	}
	public Mamifero(){
		this("terrestre");
	}
	public String tipo (){
		return "mamífero "+ habitat;
	}
}

public class Cetaceo extends Mamifero {

	public Cetaceo (String h){
		super(h);
	}

	public Cetaceo(){
		this("acuático");
	}
	public String tipo(){
		return super.tipo() + " (cetáceo)";
	}
}
```

¿Cuál será el resultado de ejecutar la siguiente secuencia de instrucciones del método principal?

```
public class prueba {
	public static void imprimirTipo(Mamifero m){
		System.out.println(m.tipo());
	}

	public static void main(String[] args) {
		Mamifero elefante = new Mamifero();
		Cetaceo delfin = new Cetaceo();
		imprimirTipo(elefante);
		imprimirTipo(delfin);
	}
}
```

Respuesta:

Cuestión 4. (0.5 puntos)

Dadas las siguientes clases:

```
public class Clase1 {
	private int atr1;
	private String atr2;

	public Clase1 (int a, String b){
		atr1=a;
		atr2=b;
	}
	public Clase1(){
		this(-1,"vacío");
	}
}

public class Clase2 extends Clase1 {
	boolean atr3;
}
```

Añadir el código necesario para que el resultado de ejecutar la siguiente secuencia de instrucciones:

```
Clase2 objeto = new Clase2(3,"hola",true);
System.out.println(objeto.toString());
```

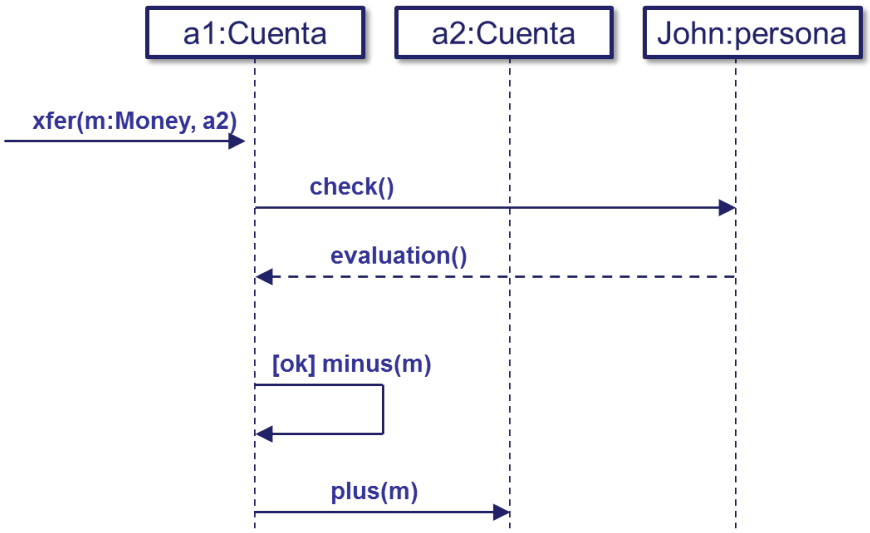
Sea:

```
Atributo 1 = 3
Atributo 2 = hola
Atributo 3 = true
```

Respuesta:

Cuestión 5. **(0.5 puntos)**

Dado el siguiente diagrama UML, ¿qué método debe implementar la clase Cuenta?



- xfer()
- xfer(); plus(); minus()
- check(); plus(); minus()
- xfer(); evaluation(); plus(); minus()

Cuestión 6. **(0.5 puntos)**

Indicar cuál de las siguientes afirmaciones es correcta sobre clases concretas, clases abstractas e interfaces en Java:

- En Java sólo se pueden crear objetos de clases concretas, pero pueden existir objetos de clases abstractas e interfaces
- En Java sólo se pueden crear objetos de clases concretas y por tanto no pueden existir objetos de clases abstractas ni de interfaces
- En Java sólo se pueden crear objetos de clases concretas, pero pueden existir objetos de clases abstractas aunque no de interfaces
- En Java se pueden crear objetos de clases concretas, abstractas e interfaces

Cuestión 7. **(0.5 puntos)**

Indicar cuales de las siguientes afirmaciones sobre los métodos recursivos de cola en Java son correctas:

- El compilador de Java cambia de forma automática su implementación de la forma recursiva a una forma iterativa
- Son aquellos métodos en los que la última llamada que se ejecuta del método recursivo es la llamada al propio método
- Pueden incluir más de un caso base
- En Java no existen los métodos recursivos de cola

Problema 1. (4 puntos)

Dada la clase `Vehiculo` cuyo código se indica a continuación:

```
public class Vehiculo {
	private String marca;
	private String color;
	protected static int unidadesKm0Concesionario = 0;
	private int id;

	public Vehiculo (String m, String c){
		marca = m;
		color = c;
		unidadesKm0Concesionario ++;
		id = unidadesKm0Concesionario;
	}
}
```

1. **(0.5 puntos)** Implemente un constructor sin parámetros que, utilizando el constructor facilitado, permita crear un objeto de tipo `Vehiculo` cuya marca sea "`indeterminado`" y cuyo color sea "`desconocido`".
   Respuesta:

2. **(0.5 puntos)** Implemente el método `toString` para esta clase.
   Respuesta:

3. **(0.25 puntos)** Implemente la clase `Coche` que hereda de la clase facilitada `Vehiculo` que tiene dos atributos privados `precio` (tipo `int`) y `descapotable` (tipo `boolean`)
   Respuesta:

4. **(0.5 puntos)** Implemente un constructor para la clase `Coche` que reciba valores para todos los parámetros (incluidos los heredados) y que haga uso del constructor de su superclase.
   Respuesta:

5. **(0.25 puntos)** Implemente un constructor sin parámetros para la clase `Coche` que, utilizando los constructores disponibles, cree un objeto de tipo `Coche` cuyo precio sea 0 y que, por defecto, no sea descapotable.
   Respuesta:

6. **(0.5 puntos)** Introduzca los mínimos cambios posibles para que sea posible serializar los objetos de ambas clases, `Vehiculo` y `Coche`.
   Respuesta:

7. **(0.5 puntos)** Implementar el método `toString` para la clase `Coche`, utilizando el método `toString` de su superclase.
   Respuesta

8. **(0.5 puntos)** Implementar el constructor de copia para la clase `Vehiculo`.
   Respuesta

9. **(0.5 puntos)** Implementar el método `equals` para la clase `Vehiculo` y para la clase `Coche` (en la implementación de este último invocar al método `equals` de su superclase). Dos objetos se considerarán idénticos si los son todos sus atributos, menos el campo `id`.
   Respuesta

Problema 2. (2 puntos)

En una aplicación Java para gestión educativa se utilizan distintos tipos de clases para dar soporte a las actividades educativas que se pueden incluir en un curso. En concreto se tienen las siguientes clases (ver código siguiente): `Documento estático`, `Vídeo`, `Documento interactivo`, `Test`, `Wiki` y `Trabajo`. A continuación se muestran las partes esenciales de estas clases en Java y las relaciones entre ellas, con algunas clases adicionales.

```
import java.util.Date;
public class AplicacionEducativa {
	public class ActividadEducativa {
		Date fechaInicio;
		Date fechaFin;
		boolean completada;
	}

	public ActividadEducativa(){
	}
}

public class Estatica extends ActividadEducativa {
}

public class Interactiva extends ActividadEducativa {
}

public class Documento extends Estatica {
}

public class Video extends Estatica {
}

public class Test extends Interactiva {
}

public class Trabajo extends Interactiva {
}
```

Se quiere modificar estas clases para permitir distinguir entre actividades de dos tipos:

- Evaluables, categoría a la que pertenecen el `Test` y el `Trabajo`. Estas actividades deben incluir funcionalidades que permitan el establecimiento de una nota y la obtención de la nota. Tómese la nota como un valor real. Por tanto este tipo de actividades deben tener un atributo de tipo `Float` para guardar la nota.
- Revisables, categoría a la que pertenecen el `Trabajo` y el `Wiki`. Estas actividades deben incluir funcionalidades que permitan introducir un comentario, ver la lista de comentarios y obtener un comentario. Tómese el comentario como un objeto de una clase específica `Comentario` (no es necesario realizar esta clase). Por tanto este tipo de actividades deben tener una lista de `Comentarios` entre sus atributos.

1. **(0.5 puntos)** Represente en un diagrama UML las clases (incluyendo la especificación de los atributos y métodos) y las relaciones entre ellas, incorporando las modificaciones necesarias para dar soporte a la funcionalidad que se quiere extender. Debe tratar de introducir los mínimos cambios sobre la programación en Java ya realizada. Además, debe considerarse que en el futuro pueden incluirse nuevos tipos de actividades (p.ej. clips de audio, exámenes), algunas de ellas

evaluables y otras revisables y por tanto el diseño y la programación debería facilitar dichos cambios

Respuesta:

2. **(0.75 puntos)** Realice la programación en Java con los cambios que se han incluido para dar soporte a las modificaciones.

   Respuesta:

3. **(0.75 puntos)** El programa de gestión educativa permite mantener alumnos y grupos. Tanto los alumnos como los grupos contendrán un listado de todas las Actividades Educativas que han experimentado. Por otro lado un alumno puede pertenecer a varios grupos. Esta estructura se muestra en el siguiente código. Completar el mismo de forma que para permitir calcular la nota media de un alumno como la media aritmética de todas las actividades evaluables en las que ha participado, bien de forma individual o en base a su pertenencia a un grupo.