EXAMEN PARCIAL DE PDOO (2º Grado en Informática) **28/11/2013**

EXAMEN RESUELTO

A) (6 puntos) Basándote en el diagrama de clases y de secuencia proporcionados, responde brevemente a las siguientes cuestiones:

1. ¿Qué cambio habría que hacer en el diagrama de clases para modelar que un barrio puede tener varias asociaciones de vecinos?

   En la relación que une Barrio con Asociación_Vecinos, junto a Asociación_Vecinos debe aparecer la cardinalidad 1..*

2. ¿Desde un objeto de la clase Ciudadano puede saberse si es presidente de la asociación de vecinos de su barrio?¿Por qué?

   No, la relación no es navegable desde Ciudadano hasta Asociación_Vecinos.

3. ¿De qué tipo es la relación entre Ciudad y Barrio? ¿qué significa?

   Es una asociación de composición, relación fuerte. Las partes no tienen sentido sin el todo. Significa que no puede haber Barrios sin que formen parte de una Ciudad

4. ¿De qué tipo es la relación entre Barrio y Ciudadano? ¿qué significa? ¿podría ser de otro tipo?

   Es una asociación de agregación, relación débil. Significa que los ciudadanos son parte del barrio. Podría haber barrios sin ciudadanos (barrios nuevos sin habitar, por ejemplo). Podría haber ciudadanos que no fueran vecinos de un barrio, que fueran por ejemplo de un poblado, donde Poblado fuera otra entidad diferente de Ciudad.

5. ¿Qué significa la línea discontinua con la indicación "{subconjunto}" entre la asociaciones de Barrio---Ciudadano y Asociacion_Vecinos---Ciudadano?

   Significa que todos los ciudadanos socios de una asociación deben ser vecinos del barrio, es decir, un subconjunto de los vecinos de un barrio son socios de la asociación de vecinos.

6. Si la junta directiva de una asociación de vecinos está formada por exactamente 6 ciudadanos, ¿cómo lo indicarías en el diagrama de clases?

   En la línea que une asociación de vecinos con ciudadano y con la palabra juntaDirectiva, se sustituye 1..* por 6 para indicar la cardinalidad de la relación.

7. ¿Qué significa que la clase Cargo esté ligada a la asociación entre Asociacion_Vecinos y Ciudadano?

   Es una clase asociación. Se utiliza porque la asociación tiene atributos propios (la responsabilidad) y complementa la asociación indicando el cargo que tiene un ciudadano concreto en la juntaDirectiva de su asociación de vecinos.

8. Implementa en Ruby los consultores/modificadores básicos de la clase Asociación_Vecinos.

   Primera forma: attr_accessor :nombre, :cif

   Segunda forma:

```
#consultores
def getNombre()
        @nombre
end
```

```
def getCif()
        @cif
end

#modificadores
def setNombre(nom)
        @nombre=nom
end

def setCif(cif)
        @cif=cif
end
```

9. ¿Cómo debería ser el constructor de Ciudadano para que inicialice el estado completo de una instancia? Impleméntalo en Java y Ruby.

Ruby

```
def initialize(nom,cif)
        @nombre=nom
        @cif=cif
end
```

Java

```
Ciudadano(String nom, String cif){
        this.nombre=nom;
        this.cif=cif;
}
```

10. Suponiendo que la visibilidad de todas las clases del diagrama es pública y que estamos codificando la clase Voto en Java ¿A qué otras clases puede acceder ésta usando directamente su nombre, sin indicar el nombre del paquete al que pertenecen?

    Puede acceder a Ciudadano (porque lo importa) y a Mesa_Electoral (porque está en su mismo paquete)

11. Dadas las siguientes instancias en Java y suponiendo que en la clase Cargo está redefinido el método equals para que compare igualdad de estado:

    Cargo cargo1= new Cargo("Vocal");

    Cargo cargo2= new Cargo("Vocal");

    Cargo cargo3 = cargo1;

    Indica si las siguientes instrucciones se evaluarían como true o false

    cargo1 == cargo2; Falso, referencian a distintos objetos

    cargo1.equals(cargo2); Verdadero, tienen el mismo estado

    cargo1 == cargo3; Verdadero, referencian al mismo objeto

    cargo1.equals(cargo3); Verdadero, tienen el mismo estado

12. Define en Java los atributos de referencia de la clase Barrio.

    private ArrayList <Ciudadano> vecinos;

    private Asociacion_Vecinos asociacion;

    nombre es un atributo básico, no de referencia.

13. ¿Desde qué clases es accesible el atributo nombre de la clase Asociacion_Vecinos?

    El atributo nombre es público, es decir, es potencialmente accesible desde cualquier clase de su paquete o de otros paquetes.

14. En el diagrama de secuencia ¿por qué el objeto :Asociacion_Vecinos es un único objeto y no multiobjeto? Razona la respuesta.

    Hay que observar también el diagrama de clases. En él, un Barrio está relacionado solo con una única asociación de Vecinos, luego no tiene sentido que haya un multiobjeto.

15. ¿Podemos deducir del diagrama de secuencia el tipo de enlace que hay entre los objetos de la clase Ciudad y Barrio? ¿por qué?

    No se puede deducir esa información mirando únicamente el diagrama de secuencia, es necesario consultar también el diagrama de clases.

B) (1,5 puntos) Sigue respondiendo brevemente a las siguientes cuestiones (no relacionadas con los diagramas):

16. Indica qué tipo de reflexión tienen Java y Ruby, explicando qué significa cada una de ellas.

    Existen dos tipos de reflexión: de introspección y de modificación.

    En Java hay introspección, se puede consultar la estructura de una clase o de un objeto en tiempo de ejecución. En Ruby es de modificación porque además de consultar también se pueden modificar y crear clases, métodos y variables en tiempo de ejecución.

17. Dada la siguiente clase Java:

```
class Comercio {
private static String m1(){   return "compra en mi tienda";  }
 static String m2(){ return m1();}
}
```

¿Daría error alguna de las tres siguientes líneas de código? Justifica tu respuesta en cada caso.

```
//Código en otra clase del mismo paquete
Comercio.m1();
        Error:m1 es privado y solo puede accederse desde la misma clase
Comercio.m2();
        Correcto: m2 tiene visibilidad package por defecto y puede accederse desde otra
        clase de su mismo paquete

// Código en otra clase de otro paquete
Comercio.m2();
```

Error: m2 tiene visibilidad package por defecto y no puede accederse desde otro paquete

18. ¿Es equivalente este código en Ruby a la declaración de la clase en Java de la pregunta anterior? Justifica tu respuesta suponiendo que todo el código está en un mismo fichero.

```
class Comercio
    def Comercio.m1
        'compra en mi tienda'
    end
    private_class_method :m1

    def self.m2
        m1
    end
end
```

En Ruby no existe la visibilidad package, los métodos tienen visibilidad pública o privada.

En el código de arriba, m2 tiene visibilidad pública, esa es la única diferencia entre los dos lenguajes.

Algunas aclaraciones más:

- Es lo mismo poner Comercio.m1 que self.m1 y también Comercio.m2 que self.m2
- En Ruby no se necesita poner return en la última instrucción de un método
- No es lo mismo 'compra en mi tienda' que puts 'compra en mi tienda'

¿Daría error de ejecución alguna de las tres líneas de código siguientes? Justifica tu respuesta en cada un caso.

#Código en el mismo módulo que la clase

Comercio.m1();

Error:m1 es privado y solo puede accederse desde la misma clase, no de otra

Comercio.m2();

Correcto: m2 tiene visibilidad publica por defecto y puede accederse desde otra clase de su mismo paquete o diferente paquete

// Código en otra clase de otro paquete

Comercio.m2();

Correcto: m2 tiene visibilidad publica por defecto y puede accederse desde otros paquetes

C) (2,5 puntos) Problemas relacionados con los diagramas proporcionados:

19. Implementa en Java y Ruby las clases Asociacion_Vecinos y Mesa_Electoral completas (Ojo: no incluir nada que no esté en el diagrama de clases proporcionado).

Java

```
package ciudadania;

class Asociacion_Vecinos{
        public String nombre;
        private String cif;
        private Barrio barrio;
        private ArrayList <Ciudadano> socios;
        private ArrayList<Cargo> cargos;
        public Boolean esMiembro(Ciudadano vecino) { }
}
package elecciones;

import ciudadania.Ciudadano;

class Mesa_Electoral { }
```

Ruby

```
module ciudadania

class Asociacion_Vecinos
        @ nombre
        @ cif
        @barrio
        @socios
        @cargos
        def esMiembro(vecino)
        end
end


module elecciones

require_relative 'ciudadania.Ciudadano'

class Mesa_Electoral

end
```

20. Partiendo del diagrama de secuencia proporcionado, implementa en Java y Ruby el método *noMiembrosAV()* de la clase Barrio.

Java

```
public ArrayList<String> noMiembrosAV(){
        ArrayList<String> noMiembros = new ArrayList();
        Boolean pertenece;
        for (Ciudadano vecino: vecinos){ // vecinos es atributo de referencia de Barrio
                pertenece=asociacion.esMiembro(vecino);
```

```
                    //asociacion es atributo de referencia de Barrio
            if (!pertenece)
                    noMiembros.add(vecino.getNombre());
                            //se presupone que existe el consultor
        }
        return noMiembros;
}
```

Algunos errores cometidos que penalizan la nota:

- se olvida el return
- no se añaden nombres a noMiembros sino a ciudadanos
- se declaran las variables asociacion y vecinos como variables locales al método
- se declaran las variables locales pertenece y noMiembros como variables de instancia.
- se incluye el paso 1.2.2.1 dentro del código.

Ruby

```
def noMiembrosAV()
        noMiembros = Array.new
        @vecinos.each do |vecino|
                    # vecinos es atributo de referencia de Barrio y por ello lleva @
            pertenece= @asociacion.esMiembro(vecino);
                    # asociacion es atributo de referencia de Barrio y por ello lleva @
            if  !pertenece
                    noMiembros << vecino.getNombre()
                            # se presupone que existe el consultor o se usa vecino.nombre
            end
        end
        noMiembros
end
```

each do es uno de los iteradores de Ruby que pueden usarse sobre las colecciones. Podrían utilizarse otros alternativos.

Algunos errores cometidos que penalizan la nota:

- se olvida la última línea: noMiembros
- no se añaden nombres a noMiembros sino a ciudadanos
- se declaran las variables asociacion y vecinos como variables locales sin poner @
- se declaran las variables locales pertenece y noMiembros como variables de instancia poniendo una @ o incluso poniendo @@ como variables de clase.
- se incluye el paso 1.2.2.1 dentro del código.