

PROGRAMACIÓN II

(Duración del examen: 2 horas y 30 minutos)

NOMBRE:

Cuestión 1. (0.5 puntos)

Dado el siguiente fragmento de código:

```
public interface Vigilante {  
    void hazTuTrabajo();  
}  
  
abstract public class AgentePrisiones implements Vigilante {  
}
```

¿Cuál de las siguientes afirmaciones es correcta?

- ☐ El código no compila porque el método hazTuTrabajo() de la interfaz Vigilante tendría que haber sido definido como abstract.
- ☐ El código no compila porque la clase agentePrisiones tiene que implementar el método hazTuTrabajo() de la interfaz Vigilante.
- ☐ El código compila sin errores.
- ☐ El código no compila porque en la declaración de la clase agentePrisiones debería utilizarse la palabra extends en lugar de implements.

Cuestión 2. (0.5 puntos)

Dado el siguiente código:

```
public class ClaseA {  
    private int atributo1;  
    public ClaseA (int a) {  
        atributo1 = a;  
    }  
    public ClaseA () {  
        this(0);  
    }  
    public String toString () {  
        return ("objeto A " + atributo1);  
    }  
}  
  
public class ClaseB extends ClaseA{  
    public ClaseB (int a) {  
        super(a);  
    }  
    public ClaseB () {  
    }  
    public String toString () {  
        return ("objeto B " + super.toString());  
    }  
}
```

```
public class ClaseC extends ClaseA{
    public ClaseC (int a) {
        super(a);
    }
    public ClaseC () {
        super();
    }
}
```

Explicar cuál es el resultado de la ejecución del siguiente fragmento de código:

```
public class Principal {
    public static void metodo1 (ClaseA objeto1){
        System.out.println(objeto1.toString());
    }

    public static void main(String[] args){
        ClaseA objeto1 = new ClaseA(1);
        ClaseB objeto2 = new ClaseB(2);
        ClaseC objeto3 = new ClaseC(3);

        metodo1(objeto1);
        metodo1(objeto2);
        metodo1(objeto3);
    }
}
```

Respuesta:

Cuestión 3. (0.5 puntos)

Completar la implementación de los métodos de la clase Viaje utilizando los huecos que se han dejado libres para ello:

```
public class Viaje {
    private String origen;
    private String destino;
    private int distancia;

    public Viaje (String origen, String destino, int distancia){

    }

    public boolean equals (          ) {

    }
}
```

Cuestión 4. (0.5 puntos)

Dadas las clases e interfaces siguientes:

```
public class PrimerPlato {
    public void pedirAgua() {
        System.out.println("Agua, por favor...");
    }
    public void pedirPan() {
        System.out.println("Pan, por favor...");
    }
}
public abstract class SegundoPlato {
    public abstract void pedirVino();
}
public interface PedirPostre {
    public void pedirTarta();
    public void pedirVinoDulce();
}
```

¿Es posible implementar una nueva clase MenuDelDia de tal forma que herede los métodos de las clases PrimerPlato y SegundoPlato, e implemente, a su vez, todos los métodos de la interfaz Postre? ¿Por qué?

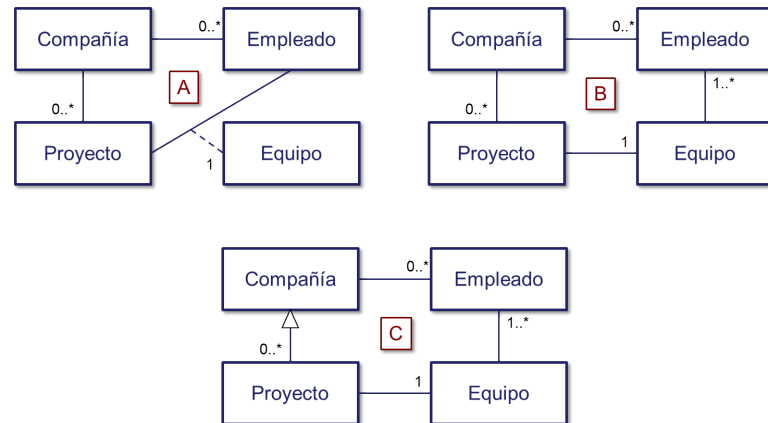
Respuesta:

Escribir el código que permita implementar la clase MenuDelDia. Incluya las modificaciones necesarias en las clases dadas.

Respuesta:

Cuestión 5. (0.5 puntos)

Una compañía realiza proyectos. Cada proyecto es llevado a cabo por un equipo de empleados. ¿Cuál de los siguientes diagramas UML (A, B ó C) representa adecuadamente el sistema?

**Cuestión 6. (0.5 puntos)**

¿Cuál de las siguientes afirmaciones sobre este diagrama UML es cierta?



- ☐ Si se efectúan cambios en el paquete C, el paquete B debe comprobarse y si es necesario efectuar cambios en éste, entonces también habrá que comprobar el paquete A, por si es necesario realizar adaptaciones también.
- ☐ Si se efectúan cambios en el paquete B, entonces el paquete A y el C han de ser comprobados, por si es necesario realizar algún cambio también en ellos.
- ☐ Los paquetes A, B y C han de ser diseñados de forma que los cambios efectuados en uno no tengan efecto en el resto.
- ☐ Si se realizan cambios en el paquete C, entonces tanto el paquete A como el C tienen que revisarse, porque las relaciones de dependencia son transitivas.

Cuestión 7. (0.5 puntos)

Dado el siguiente código de un método

```
public static void XYZ(String m, int n) {  
    if (n >= 1) {  
        System.out.println(m);  
        XYZ(m, n - 1);  
    }  
}
```

Indicar qué tipo de método es y qué hace

Respuesta:

Cuestión 8. (0.5 puntos)

¿Cuáles de las siguientes afirmaciones sobre Métrica V3 son ciertas?

- ☐ Es el modelo de desarrollo software para la administración pública española, englobando la administración local, regional y nacional.
- ☐ Sólo permite el desarrollo de software de acuerdo al paradigma orientado a objetos.
- ☐ Utiliza otros estándares como UML para la representación gráfica de los modelos, ISO 12207 para la descripción de los ciclos de vida e IEEE 1074 para la identificación de las actividades de desarrollo y mantenimiento del software.
- ☐ Es un modelo Ágil como Scrum.

Problema 1. (1 punto)

Libro
- titulo: String - descripcion: String - prestado: boolean + id: int + tipoDeLibro: int
+ Libro(titulo:String, descripcion:String, tipo: int) + getTitulo(): String + getDescripcion(): String + getPrestado(): boolean + setPrestado(prestado: boolean): void

Dada la clase Libro representada en el diagrama UML, se pide:

1. **(0.5 puntos)** Escriba el código de la clase Libro (que pertenece al paquete examenJulio2012) sabiendo que el atributo tipoDeLibro podrá adoptar 3 valores: novela (valor 1); relatos (valor 2) y poesia (valor 3). Estos 3 tipos serán definidos como constantes públicas comunes para todos los objetos de la clase. Agregue un atributo visible por cualquier otra clase denominado

contador que contabilice el número de libros que se han creado y un método denominado getContador que permita obtener este valor. Finalmente, el atributo id será un número único para cada libro que le será asignado al crearlo como un valor consecutivo partiendo del número 200.

Respuesta:

2. **(0.5 puntos)** Implementar una interfaz denominada `Ordenable` que tiene las siguientes características:
- a) Pertenece también al paquete `examenJulio2012`
 - b) Tiene un método público denominado `getOrden` que no recibe parámetros y devuelve un número.

Haga que la clase `Libro` implemente dicha interfaz tal que el orden de un libro sea igual a su identificador.

Respuesta:

Problema 2. (3 puntos)

Se tiene la definición de la clase Guerrero como sigue :

```
public abstract class Guerrero {
    private final String nombre;
    private int edad, fuerza;
    private boolean herido, muerto;
    // Aquí métodos get y set adecuados para los atributos anteriores

    protected static boolean comprobarEdad (int e) {
        return ((e >= 15) && (e <= 60));
    }
    protected static boolean comprobarFuerza (int f) {
        return ((f > 0) && (f <= 10));
    }
    public abstract boolean retirarse();
}
```

1. **(0.5 puntos)** Crear tres constructores para la clase Guerrero:

- a) El primero recibirá valores para todos los atributos, excepto herido y muerto que, obviamente, serán inicializados como falsos. Deberá comprobar que los valores dados son válidos y, en caso contrario, establecer como edad 25 y fuerza 5.
- b) El segundo, que deberá utilizar el primero, no recibirá ningún valor, y creará un guerrero cuyo nombre sea guerreroX y cuya edad y fuerza valgan 15 y 1, respectivamente.
- c) El tercero recibirá un objeto de tipo Guerrero y un nombre y creará una copia del guerrero que recibe como parámetro con el nuevo nombre.

Respuesta:

2. **(0.5 puntos)** Crear una clase `Troyano` y una clase `Griego`, heredando ambas de `Guerrero`, que no tienen ningún nuevo atributo.
- a) Crear un constructor para cada una de ellas que reciba el nombre, edad y fuerza y que utilice alguno de los constructores de la clase `Guerrero`.
 - b) Implementar los métodos necesarios para que ninguna de las dos clases sea abstracta, teniendo en cuenta que los métodos devolverán `true` si se ha podido realizar la acción y `false` si no ha sido así. Los troyanos no pueden retirarse. Los griegos sí pueden hacerlo siempre y cuando estén heridos y, obviamente, no estén muertos.

Respuesta:

3. **(0.5 puntos)** Implementar un constructor de copia para las tres clases anteriores (`Guerrero`, `Troyano` y `Griego`) de tal forma que los constructores de copia de las clases derivadas **sólo** utilicen el constructor de copia de la clase padre.

Respuesta:

4. **(0.5 puntos)** Crear la clase CaballoDeTroya que tendrá como atributos públicos:
- a) capacidad (tipo int) que representa el número de guerreros griegos que puede haber dentro del caballo. Una vez asignado un valor no se podrá modificar
 - b) ocupacion (tipo int) que representa el número actual de guerreros griegos dentro del caballo
 - c) ocupantes, un array de objetos de la clase Guerrero, especificando quiénes están dentro del caballo

Crear para esta clase crear un constructor que reciba un array de Guerreros y compruebe que todos ellos son Griegos. Si es así se creará el caballo con capacidad igual a la del número de guerreros recibidos; en caso contrario creará un caballo sin ocupantes con capacidad 100.

Respuesta:

4. **(0.5 puntos)** Definir la interfaz `PuedeMontarse` que especifica cómo montar en los objetos de tipo `CaballoDeTroya`. Ha de tener los dos métodos siguientes:
- a) `int montar (Guerrero g)`: que monta un guerrero en el caballo y devuelve el número de ocupantes que hay ó -1 si el caballo ya está lleno. Deberá comprobar que sólo los griegos pueden montar en el caballo; si un guerrero troyano intenta subir deberá devolver un valor -2.
 - b) `void desmontar()`: que fuerza a todos los guerreros a bajar del caballo.

Modifique la clase `CaballoDeTroya` para que implemente esta interfaz.

Respuesta:

5. **(0.5 puntos)** La clase `Guerrero` tiene dos métodos cuyo modificador de acceso es `protected`. Explique para qué se utiliza este modificador y compárelo con el resto de los modificadores de acceso que conozca.

Respuesta:



Problema 3. (2 puntos)

Una parte del programa de gestión de la universidad de Vigo, programado en Java, se ocupa de la gestión del personal. Desafortunadamente, un becario al que se le había encargado optimizar el código de cálculo de las nóminas ha borrado una parte importante del mismo.

```
public class Universidad {  
  
    public static void main(String[] args) {  
  
        Persona[] personas = ...; // Se leen de una base de datos las personas registradas en la  
        Universidad de Vigo  
  
        for (int i = 0; i < personas.length; i++) {  
  
            if (personas[i] instanceof Cobrador) System.out.println("Sueldo de " +  
            personas[i].getId () + " es " + ((Cobrador)personas[i]).getNomina() + "Euros");  
  
        }  
  
    }  
  
}
```

Sabiendo que los profesores y los administrativos son empleados y cobran nómina y que los alumnos no, pero los alumnos becarios sí, se pide.

1. **(0.75 puntos)** Crear una jerarquía de clases (con clases abstractas e interfaces) de forma fiel a las indicaciones anteriores, que permita de la forma más sencilla posible mantener el código anterior con un funcionamiento adecuado. Téngase en cuenta los métodos necesarios.

Respuesta:

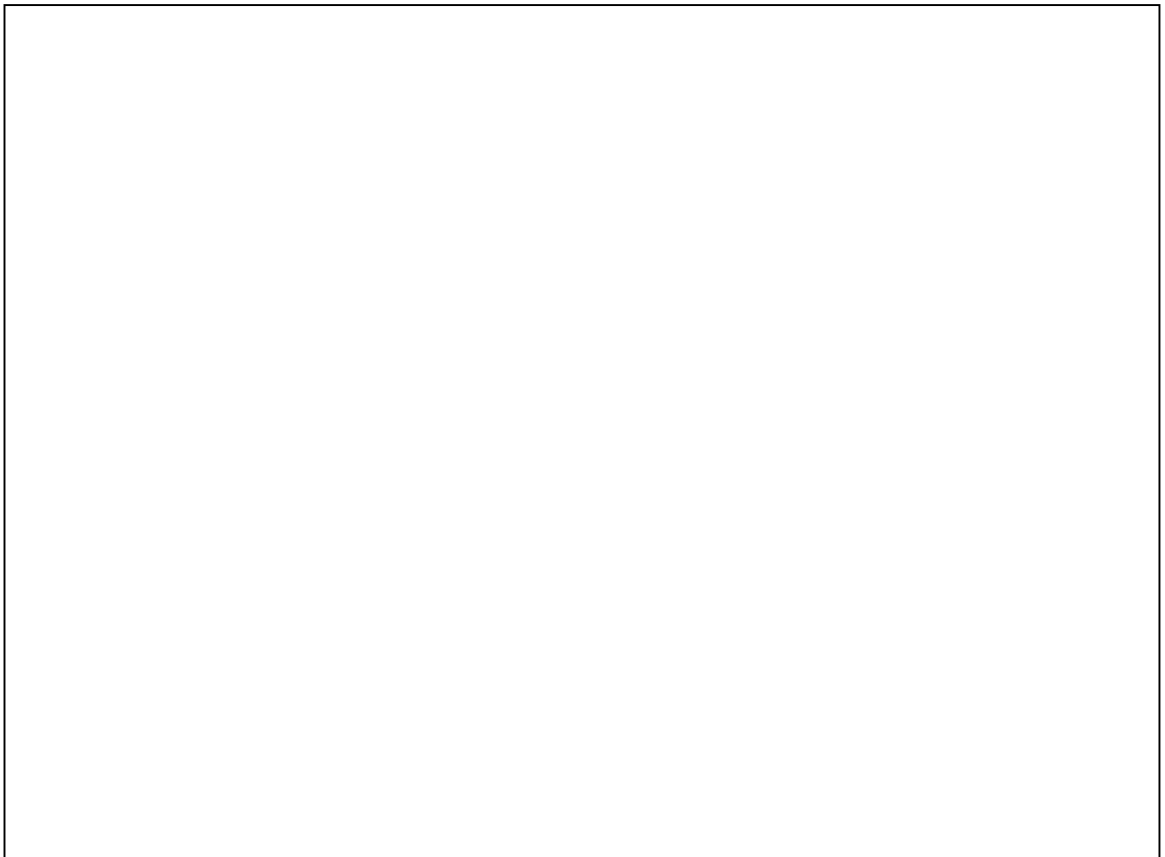
2. **(0,5 puntos)** Representar en un diagrama de clases UML las clases e interfaces identificadas y las relaciones entre ellas.

Respuesta:

A large, empty rectangular box with a thin black border, intended for drawing a UML class diagram.

3. **(0,75 puntos)** Modificar el programa anterior para que se escriban primero las nóminas de los profesores, a continuación las de los administrativos y por último las de los becarios.

Respuesta:

A large, empty rectangular box with a thin black border, intended for writing the modified program code.