

# **Programación y Diseño Orientado a Objetos**

## **Curso 2021-2022 - Examen de Teoría – 17 de enero de 2022**

Apellidos: Ruiz Gomez Nombre: Soledad  
DNI: \_\_\_\_\_ Grupo o Profesor de teoría: \_\_\_\_\_

Nombre: Soledad

## TIPO A

## ABAJO EL DIAGRAMA

1. (5 puntos) Dado el diagrama de clases proporcionado, responda a las siguientes cuestiones poniendo V (verdadero) o F (falso) en la tabla de más abajo. Tenga en cuenta que una respuesta correcta suma 0,25 puntos, una respuesta incorrecta resta 0,25 puntos y una respuesta en blanco ni suma ni resta. **Las dos primeras respuestas incorrectas no restan nota.**

- F** 1. La clase Camara tiene un atributo de tipo Empleado. ✓

**V** 2. El valor del atributo NumCamaras es compartido por todos los objetos de la clase Camara y también de sus subclases. F

**F** 3. En Java, si en la clase Camara no definimos un método consultor y modificador (get/set) del atributo “marca”, éste no será accesible para objetos de la clase CamaraFoto. ✓

**F** 4. En Ruby, es posible cambiar la visibilidad del atributo de instancia “marca” para que sea público.

**V** 5. Un objeto de la clase CamaraReflex es capaz de ejecutar tres funcionalidades diferentes: establecerResolucion, cambiarObjetivo y calibrar; y algunas otras heredadas de la clase Object.

**V** 6. El tipo Imprimible se puede usar para declarar variables pero no para crear objetos.

**F** 7. Un objeto de la clase CamaraFoto tiene solo un atributo de instancia.

**V** 8. En Java, usando el modificador "default" se podría asignar un código por defecto al método imprimir() de la interfaz Imprimible.

**V** 9. La clase Video tiene un atributo de tipo Camara que podría ser un objeto del tipo CamaraFoto, CamaraVideo o CamaraReflex.

**F** 10. Se pueden crear objetos de la clase Camara.

**F** 11. El método realizaEncargo de Empleado sería accesible en Java sólo por las posibles subclases de Empleado, pero no por las otras clases del paquete Encargos. ✓

**F** 12. Al hacer ob1 = ob2; se está creando una copia de ob2 y asignándola a ob1.

**V** 13. Si eliminamos el método calibrar de la clase CamaraVideo, dicha clase sería abstracta.

**V** 14. Si se declara en Java una nueva clase: class Equipamiento<T extends Camara>{...}, podría hacer posteriormente esta declaración: Equipamiento<CamaraReflex> equiporeflex;

**V** 15. En Java, si declaro una variable de tipo Imprimible podría asignarle un objeto de tipo Foto.

**V** 16. En Java, el siguiente código daría error de compilación al invocar al método voltear:  
Imprimible imp = new Foto(); imp.voltear();

**V** 17. En Java, es posible añadir métodos nuevos a la clase Empleado en tiempo de ejecución. Reflexión

**F** 18. En Ruby, dentro de un método de instancia de la clase CamaraFoto no se podría modificar el valor del atributo formatoSensor para otro objeto distinto del receptor. depende de lo que sea establecerResolucion,

**V** 19. En Java, un objeto de la clase CamaraReflex si recibe la petición “calibrar” ejecutará el método definido en la clase CamaraReflex, aunque haya sido declarado con tipo estático CamaraFoto. Esto es así porque existe ligadura dinámica.

**V** 20. En Java, si un objeto de la clase CamaraVideo modifica el valor de NumCamaras, el cambio no afecta a los objetos del resto de subclases de Camara.

- 1- Empleado tiene un atributo de tipo cámara
- 2- Porque es privado
- 3- Verdadero porque marca es privado
- 4- En Ruby todos los atributos son privados, sin excepciones
- 5- Porque hereda los métodos públicos de CamaraFotos
- 6- En interfaces y c. abstractas pueden utilizarse para declarar tipos ("a la izq. del igual" tipo estat), lo que no pueden es instanciarse (usarse como tipo dinámico). En un objeto declarado con el nombre de una interfaz
- 7- No hereda los de cámara porque son privados
- 8:
- 9- No estoy segura
- 10- NO SE PUEDEN CREAR OBJETOS DE UNA CLASE ABSTRACTA 
- 11- En un método PROTECTED permite el acceso desde subclases y miembros del paquete en un método PRIVATE solo pueden acceder los de la clase en la que se declaran.
- 12- Esa declaración lo que hace es asignar la referencia de obj1 a obj2, esto hace que apunten a un mismo objeto en memoria.
- 13- Una clase es abstracta si tiene al menos un método abstracto (NO seguro)
- 14-
- 15- Si se puede ya que Imprimible se implementa en foto
- 16- No denía error, pero no sé porqué
- 17-
- 18-
- 19- Se usaría el de la clase hija que se sobrescribe (NO seguro)
- 20- Al ser numerosos estaticos afecta a todos los objetos de las subclases.

2. (2 puntos) Atendiendo al diagrama de clases anterior, indique en las siguientes sentencias Java el tipo de error, si es que lo hay, y su solución, si la tuviera. En el tipo de error escriba C si es de Compilación, E si es de Ejecución u OK si no hay error. Considere que siempre se pasan todos los argumentos necesarios a los constructores y que dichas sentencias están en un método *main* en una clase nueva dentro del paquete “Encargos”.

CÓDIGO (secuencial e incremental)	ERROR (C, E, OK)	CORRECCION (si no es posible, indique por qué)
Camara c, c1, c2, c3;	OK	
c = new Camara();	C	no puedo instanciar una abstracta
c1 = new CamaraReflex();	OK	
c1.establecerResolucion();	C	((CamaraReflex) c1).establecer R();
c1.calibrar();	OK	No está implementado pero no hace falta
c2 = new CamaraFotos();	OK	
((CamaraReflex) c2).cambiarObjetivo(3);	E	He prometido CamaraReflex
c3 = new CamaraVideo();	OK	
((CamaraFotos) c3).establecerResolucion("Alta");	E	Cará no coincide con tipo dinámico
ArrayList<Camara> maquinas = new ArrayList<Camara>();	OK	
maquinas.add(c1);	OK	
maquinas.add(c2);	OK	
maquinas.add(c3);	OK	
maquinas.get(0).calibrar();	OK	
maquinas.get(1).establecerResolucion("Alta");	OK/C	cará ((CamaraFotos) m.get(1)).setR(" ");
Encargo e = new Video();	C	video no hereda de encargo (es comp)
c = c3;	OK	

3. (1,5 puntos) Implemente en Java:

3.A La cabecera de la clase Camara. (0.25 puntos)	<pre>public abstract class Camara public abstract class Camara { }</pre>
3.B El constructor de la clase CamaraFotos considerando que los constructores de las clases del diagrama reciben argumentos para inicializar todos sus atributos. (0.75 puntos)	<pre>CamaraFotos (String ma, String mo, String f) {     super (ma, mo);     formatoSensor = f; }</pre>
3.C La clase Foto completa, sin constructor y dejando el código de los métodos vacío. (0.5 puntos)	<pre>public class Foto implements Imprimible {     Encargo e;     CamaraFotos tomadaCon;     public void imprimir () {}     public void voltear () {} }</pre>

4. (1,5 puntos) Implemente en Ruby:

<p>4.A El código necesario a incluir en la clase Camara, para simular que dicha clase es abstracta. (0.5 puntos)</p>	<pre> class Camara   private_class_method :new end  class Camara   <del>class_method :new</del>   <del>(--&gt; private_class_method :new</del> end  en el resto: <del>class_method :new</del> public_class_method :new   <small>puedo acceder a los privados de la clase base desde la Hija</small> </pre>
<p>4.B: El método calibrar() de las clases CamaraReflex y CamaraFotos, teniendo en cuenta que calibrar en CamaraFotos consiste en establecer dos resoluciones, primero la resolución "Alta" y luego la "Baja". Mientras que calibrar en CamaraReflex es igual que la calibración en CamaraFotos pero, además, estableciendo previamente la resolución a "Media". (1 punto)</p>	 <p>NADA SEGURA</p>

```

class CamaraFotos < Camara
  ...
  def calibrar
    RALTA = "Alta"
    RBAYA = "Baja"
    establecerResolucion(RALTA)
    establecerResolucion(RBAYA)
  end
end

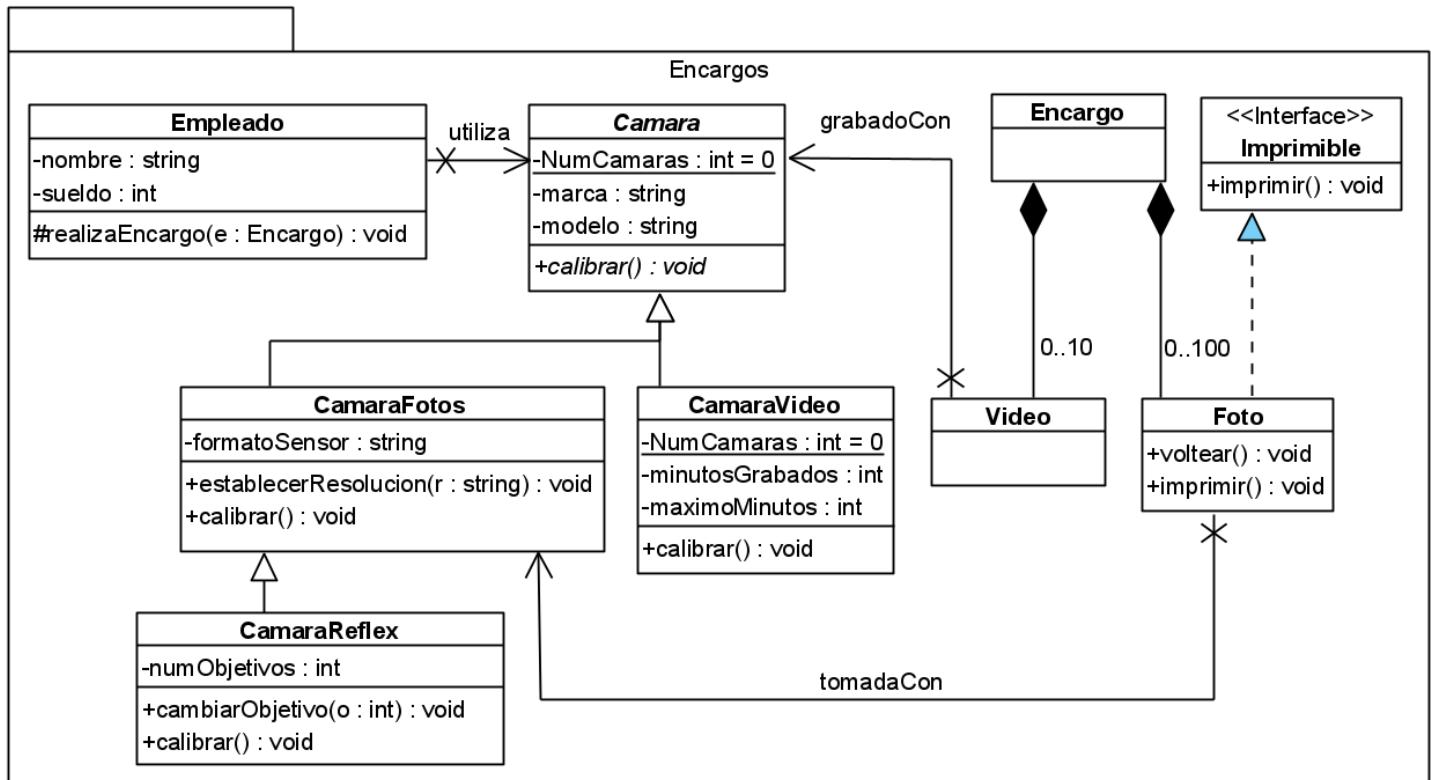
```

podemos  
ahí

```

class CamaraReflex < CamaraFotos
  ...
  def calibrar
    RMEDIA = "Media"
    establecerResolucion(RMEDIA)
    super
  end
end

```



Aclaraciones:

- El paquete al que pertenecen todas las clases se llama “Encargos”.
- Están en cursiva el nombre de la clase Cámara y el método calibrar de Camara.