



Universidad Rey Juan Carlos
Departamento de Lenguajes y Sistemas Informáticos I
23 de Junio de 2009

Duración: 2 horas.

Lea atentamente todo el enunciado de cada pregunta antes contestar.

Especifique nombre y apellidos en todas las hojas que entregue.

Cualquier suposición o decisión sobre el enunciado del examen debe ser detallada y justificada convenientemente.

Apellidos y Nombre:

Número de expediente:

PROGRAMACIÓN ORIENTADA A OBJETOS – JUNIO 2009

1. Se dice que las aplicaciones realizadas en lenguaje Java son portables ¿Por qué? Justifique razonadamente su respuesta [0,5 puntos]

Solución:

Un programa Java se puede ejecutar en cualquier lugar debido a que la salida del compilador de Java NO es un ejecutable, es bytecode, un conjunto de instrucciones de alto nivel que se ejecutarán en una máquina virtual de Java (JVM). La JVM interpreta los 'bytecodes' creados por el compilador y los convierte a código particular de la CPU utilizada.

2. Suponiendo que se encuentra definida la clase *Pagina*. ¿Qué es lo que hacen los métodos *annadeCapitulo* y *modificaVersion*? Justifique razonadamente su respuesta [0,5 puntos]

```
class Documento extends Pagina {
    static int version = 1;
    int numero_de_capitulos;
    static void annadeCapitulo() { numero_de_capitulos++;}
    static void modificaVersion( ) { version++; }
}
```

Solución:

En el código del método *annadeCapitulo* hay un error de compilación ya que se intenta acceder a una variable no estática desde un método estático.

El método *modificaVersion* se encarga de incrementar la variable estática que contiene el número de la versión.

3. Implementar una interfaz llamada *Pelicula* que contenga los métodos necesarios para reproducir una película, pausarla en un momento determinado y detenerla. **[0,75 puntos]**

Solución:

```
public interface Pelicula {
    boolean reproducir();
    boolean pausar();
    boolean detener();
}
```

4. ¿El siguiente código es correcto? Justifique razonadamente su respuesta **[0,5 puntos]**

```
try {
    ...
} finally {
    ...
}
```

Solución:

El código es correcto. El control de excepciones se realiza con bloques try – catch – finally. El código susceptible de generar errores debe estar acotado en un bloque try. Cuando se produce un error en el bloque try se genera un objeto del tipo de la excepción que se ha producido. Esta excepción se puede recoger en un bloque catch, pero es necesario. Se pueden incluir tantos bloques catch como sean necesarios, cada uno de los cuales atrapará un tipo de excepción. El bloque finally es opcional: Las sentencias de este bloque se ejecutan siempre (se produzca o no la excepción).

5. Cuando un usuario interactúa con un programa Java que tiene interfaz gráfica, se producen eventos. ¿Qué pasos son necesarios para realizar una determinada acción cuando se produce un evento? **[0,5 puntos]**

Solución:

Para programar respuestas a los eventos que se producen sobre un determinado objeto, es necesario utilizar oyentes. Un oyente es un objeto de ejecutar una respuesta cuando se produce un evento. Los pasos a seguir son: i) declarar un oyente y ii) registrar un objeto oyente para un objeto fuente.

6. Implementar una clase llamada Vehículo en la que se guarde información sobre la velocidad del vehículo, el número de ruedas que tiene y la gasolina que le queda. Esta clase deberá tener los métodos necesarios para:

- Permitir acelerar un número de kilómetros/hora la velocidad del vehículo
- Parar el vehículo
- Echar gasolina

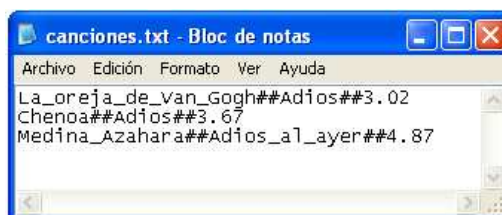
[0,75 puntos]

Solución:

```
class vehiculo {
    private int velocidad;
    private int ruedas;
    private double gasolina;
    public vehiculo(int r){
        this.velocidad = 0;
        this.gasolina = 0;
        this.ruedas = r;
    }
    public void parar() { velocidad = 0;}
    public void acelerar(int kms) { velocidad += kms;}
    public void repostar(double litros) { gasolina += litros;}
}
```

Nota: Se asume que también existirían los métodos set y get correspondientes con cada uno de los atributos de la clase

7. Se desea realizar un programa que permita gestionar una base de datos de música. Para ello se deberán implementar dos clases: Cancion y BDCanciones. La clase Cancion contiene información sobre el intérprete, duración y título de la canción. La clase BDCanciones contiene información sobre una serie de canciones, que se encargará de leer de un fichero de texto con el siguiente formato.



El programa principal que lee el fichero de texto "canciones.txt", carga la información almacenada en un objeto de tipo BDCanciones y visualiza tanto el número de canciones del fichero como la información asociada a cada canción es el siguiente:

```
public static void main(String[] args) throws IOException
{
    BDCanciones bd = new BDCanciones();
    bd.cargaFicheroCanciones("canciones.txt");

    System.out.println("Numero de canciones: " + bd.getNumCanciones());
    bd.visualizaCanciones();
}
```

Se pide implementar el código de la clase Cancion y BDCanciones para que el código del programa principal cargue la información del fichero canciones.txt en un objeto de la clase BDCanciones y visualice la información de cada una de ellas. [4 puntos]

Solución:

```
import java.util.ArrayList;

public class BDCanciones {
    private ArrayList<Cancion> canciones;

    public BDCanciones(){ this.canciones = new ArrayList<Cancion>();}

    public ArrayList<Cancion> getCanciones(){ return canciones;}

    public int getNumCanciones(){ return getCanciones().size(); }
    public void insertaCancion(Cancion c){ getCanciones().add(c);}

    public void visualizaCanciones()
    {
        for (Cancion c: getCanciones()){ c.visualizaCancion();}
    }

    public void cargaFicheroCanciones(String f) throws IOException
    {
        BufferedReader in = new BufferedReader( new FileReader(f));
        String linea;
        while((linea = in.readLine())!= null)
        {
            Cancion c = new Cancion();
            c.datosCancion(linea);
            this.insertaCancion(c);
        }
        in.close();
    }
}

public class Cancion {
    private String interprete;
    private String titCancion;
    private double duracion;

    public Cancion(){}

    public String getInterprete(){ return this.interprete ;}
    public String getTitCancion(){ return this.titCancion ;}
    public double getDuracion(){ return this.duracion ;}

    public void setInterprete(String i){ this.interprete = i;}
    public void setTitCancion(String t){ this.titCancion = t;}
    public void setDuracion(double d){ this.duracion = d;}

    public void datosCancion(String l){
        String[] campos = l.split("##");
        setInterprete(campos[0]);
        setTitCancion(campos[1]);
        setDuracion(Double.valueOf(campos[2]));
    }

    public void visualizaCancion(){
        System.out.println("Interprete: " + getInterprete() +
            " Titulo: " + getTitCancion() +
            " Duracion: " + getDuracion());
    }
}
```

8. Se quiere diseñar un sistema de envío de mensajes a móviles. Existen dos tipos de mensajes que se pueden enviar entre móviles, mensajes de texto (SMS) y mensajes que contienen imágenes (MMS). Por un lado, los mensajes de texto contienen un mensaje que se desea enviar de un móvil a otro. Por otro lado, los mensajes que contienen imágenes almacenan información sobre la imagen a enviar, la cual se representará por el nombre del fichero que la contiene. Independientemente del tipo de mensaje, cada mensaje tendrá asociado un remitente de dicho mensaje y un destinatario. Ambos estarán definidos obligatoriamente por un número de móvil, y opcionalmente se podrá guardar información sobre su nombre. Además, los métodos *enviarMensaje* y *visualizarMensaje* deben estar definidos tanto para mensajes de texto como para mensajes que contienen imágenes. Se pide realizar el diagrama de clases para este sistema, indicando las clases, atributos de cada una de ellas, métodos, relaciones y multiplicidades (si las hubiera). **[2,5 puntos]**

Solución:

