	I.E.S ARCIPRESTE DE HITA. DEPARTAMENTO DE INFORMÁTICA. Curso 2022/2023 **Módulo:** PROG – 1EDAW 1ª Ordinaria segunda parte	**NOTA:**
	Nombre y apellidos:	Fecha: 24/05/2023

1. (1 punto) Crea un programa que lea una serie de valores numéricos enteros desde el teclado y los guarde en un ArrayList de tipo Integer. La lectura de números termina cuando se introduzca el valor -99. Este valor no se guarda en el ArrayList.

2. (0,75 puntos) Muestra por pantalla todos los valores guardados en el ArrayList.

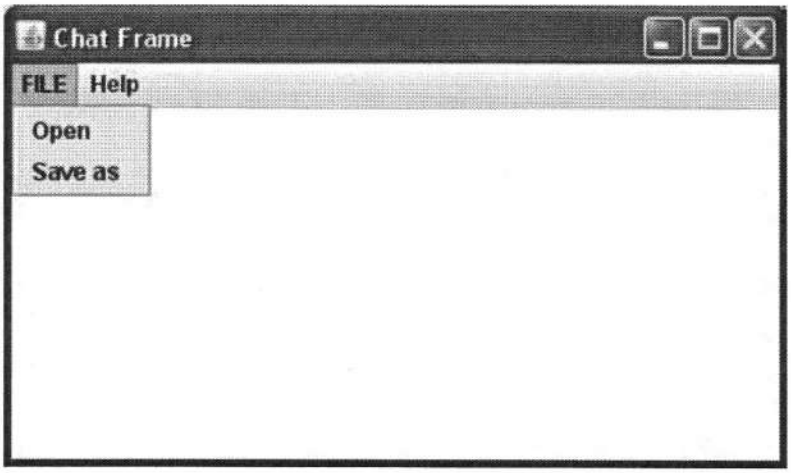
3. (0,75 puntos) Dada la siguiente clase:

```
public class Vehiculo
{

     private String matricula;
     private String marca;
          private String tamañoDeposito;
     private String modelo;
     private static int contador=0;


}
```

Incluye las modificaciones necesarias para que los objetos de dicha clase sean serializables. Ten en cuenta que el atributo tamañoDeposito no queremos que se almacene en el archivo.

4. (1 punto) Crear un método main que cree un objeto, lo guarde en un fichero y lo lea del fichero.

5. (2 puntos) Establece los elementos necesarios para crear la siguiente interfaz:



	I.E.S ARCIPRESTE DE HITA. DEPARTAMENTO DE INFORMÁTICA. Curso 2022/2023 **Módulo:** PROG – 1EDAW 1ª Ordinaria segunda parte	**NOTA:**
	Nombre y apellidos:	Fecha: 24/05/2023

6. (2 puntos) Genera la siguiente estructura de clases:

   - Escriba una clase Multimedia para almacenar información de objetos de tipo multimedia (películas, discos, mp3...). Esta clase contiene título, formato y duración como atributos. El formato puede ser uno de los siguientes: wav, mp3, midi, avi, mov y dvd. El valor de todos los atributos se pasa por parámetro en el momento de crear el objeto, para lo que debes generar el constructor correspondiente.

     Esta clase tiene, además, un método toString() que devuelve la información del objeto. Esta clase nunca va a poder ser instanciada.

   - Escriba una clase Película que herede de la clase Multimedia anterior. La clase Película tiene, además de los atributos heredados, un actor principal. La clase debe tener un método para obtener el nuevo atributos y debe sobreescribir el método toString() para que devuelva, además de la información heredada, la nueva información.

1. (0,5 puntos) Genera un método llamado conexión que nos permita establecer una conexión con una base de datos orientada a objetos, como, por ejemplo, objectDB.

2. (2 puntos) Crea un método que reciba como parámetros un nombre y un objeto de conexión y ejecute una consulta a la base de datos de aquellos datos cuyo nombre coincida con el dado. Debe devolver el objeto generado por dicha consulta.

Ejercicio 1:

```
import java.util.ArrayList;
import java.util.Scanner;

public class LecturaNumeros {
    public static void main(String[] args) {
        ArrayList<Integer> numeros = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);

        System.out.println("Introduce los números enteros (termina con -99):");

        try {
            int numero;
            while ((numero = scanner.nextInt()) != -99) {
                numeros.add(numero);
            }
        } catch (Exception e) {
            System.out.println("Error: Asegúrate de ingresar solo valores enteros.");
        }
```

Ejercicio 2:

```
        if (numeros.isEmpty()) {
            System.out.println("No se ingresaron números.");
        } else {
            System.out.println("Números introducidos:");
            for (int num : numeros) {
                System.out.println(num);
            }
        }

        scanner.close();
    }
}
```

Ejercicio 3:

```
public class Vehiculo implements Serializable
{
private String matricula;
private String marca;
private transient String tamañoDeposito;
private String modelo;
private static int contador=0;
}
```

Ejercicio 4:

```
import java.io.*;

public class Ejercicio4 {
    public static void main(String[] args) {
        // Crear objeto Vehiculo
        Vehiculo vehiculo = new Vehiculo();
        vehiculo.setMatricula("ABC123");
        vehiculo.setMarca("Toyota");
        vehiculo.setTamañoDeposito("50L");
        vehiculo.setModelo("Corolla");

        // Guardar objeto en un archivo
        String fileName = "vehiculo.dat";
        guardarVehiculo(vehiculo, fileName);

        // Leer objeto desde el archivo
        Vehiculo vehiculoGuardado = leerVehiculo(fileName);

        // Imprimir información del objeto leído
        if (vehiculoGuardado != null) {
            System.out.println("Vehículo leído del archivo:");
            System.out.println("Matrícula: " + vehiculoGuardado.getMatricula());
            System.out.println("Marca: " + vehiculoGuardado.getMarca());
            System.out.println("Tamaño del depósito: " +
vehiculoGuardado.getTamañoDeposito());
            System.out.println("Modelo: " + vehiculoGuardado.getModelo());
        }
    }

    private static void guardarVehiculo(Vehiculo vehiculo, String fileName) {
        try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(fileName))) {
            outputStream.writeObject(vehiculo);
            System.out.println("Vehículo guardado en el archivo: " + fileName);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static Vehiculo leerVehiculo(String fileName) {
        try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(fileName))) {
            Vehiculo vehiculo = (Vehiculo) inputStream.readObject();
            System.out.println("Vehículo leído del archivo: " + fileName);
            return vehiculo;
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

Ejercicio 5:

```
import javax.swing.*;

public class ChatFrame extends JFrame {
    public ChatFrame() {
        // Configurar JFrame
        setTitle("Chat Frame");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 300);

        // Crear JMenuBar
        JMenuBar menuBar = new JMenuBar();

        // Crear JMenu "FILE"
        JMenu fileMenu = new JMenu("FILE");

        // Crear JMenuItem "Open"
        JMenuItem openItem = new JMenuItem("Open");
        fileMenu.add(openItem);

        // Crear JMenuItem "Save as"
        JMenuItem saveAsItem = new JMenuItem("Save as");
        fileMenu.add(saveAsItem);

        // Agregar JMenu "FILE" al JMenuBar
        menuBar.add(fileMenu);

        // Crear JMenu "Help"
        JMenu helpMenu = new JMenu("Help");

        // Agregar JMenu "Help" al JMenuBar
        menuBar.add(helpMenu);

        // Establecer JMenuBar en el JFrame
        setJMenuBar(menuBar);
    }

    public static void main(String[] args) {
        ChatFrame chatFrame = new ChatFrame();
        chatFrame.setVisible(true);
    }
}
```

Ejercicio 6:

```
public abstract class Multimedia {
    private String titulo;
    private String formato;
    private int duracion;

    public Multimedia(String titulo, String formato, int duracion) {
        this.titulo = titulo;
        this.formato = formato;
        this.duracion = duracion;
    }

    public abstract String toString();
}
```

```
public class Pelicula extends Multimedia {
    private String actorPrincipal;

    public Pelicula(String titulo, String formato, int duracion, String
actorPrincipal) {
        super(titulo, formato, duracion);
        this.actorPrincipal = actorPrincipal;
    }

    public String getActorPrincipal() {
        return actorPrincipal;
    }

    public void setActorPrincipal(String actorPrincipal) {
        this.actorPrincipal = actorPrincipal;
    }

    @Override
    public String toString() {
        return "Película - Título: " + getTitulo() + ", Formato: " + getFormato() +
                ", Duración: " + getDuracion() + " minutos, Actor Principal: " +
actorPrincipal;
    }
}
```

Ejercicio 7:

```
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class Ejercicio 7 {
    private EntityManagerFactory emf;
    private EntityManager em;

    public void conexión() {
        // Establecer la configuración de conexión
        String persistenceUnitName = "$objectdb/db/nombre_base_datos.odb";

        // Crear el EntityManagerFactory
        emf = Persistence.createEntityManagerFactory(persistenceUnitName);

        // Crear el EntityManager
        em = emf.createEntityManager();

        // Realizar las operaciones en la base de datos...

        // Cerrar el EntityManager y el EntityManagerFactory
        em.close();
        emf.close();
    }
}
```

Ejercicio 8:

```
import java.sql.*;

public class Ejercicio8 {
    public ResultSet ejecutarConsulta(String nombre, Connection connection) {
        ResultSet resultSet = null;
        String consulta = "SELECT * FROM tabla WHERE nombre = ?";

        try {
            // Crear la sentencia preparada con el parámetro
            PreparedStatement statement = connection.prepareStatement(consulta);
            statement.setString(1, nombre);

            // Ejecutar la consulta y obtener el resultado
            resultSet = statement.executeQuery();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return resultSet;
    }
```