En cada uno de los siguientes programas, identifique si la clase usa el patrón singleton o no y justifique el porqué.

1. **Conexión a base de datos**

```java
public class DatabaseConnection {
    private static DatabaseConnection instance;

    private static boolean resetFlag = true;

    private DatabaseConnection() {
        System.out.println("Conexión a la base de datos creada.");
    }

    public static DatabaseConnection getInstance() {
        if (instance == null || resetFlag) {
            instance = new DatabaseConnection();
            resetFlag = false;
        }
        return instance;
    }

    public void query(String sql) {
        System.out.println("Ejecutando consulta: " + sql);
    }
}

public class Main {
    public static void main(String[] args) {
        DatabaseConnection connection1 = DatabaseConnection.getInstance();
        connection1.query("SELECT * FROM usuarios");
    }
}
```

2. Registro de eventos en una aplicación.

```java
public class Logger {

    private static final Logger instance = new Logger();

    private Logger() {}

    public static Logger getInstance() {
        return instance;
    }

    public void log(String message) {
        System.out.println("Log: " + message);
    }
}

public class Main {
    public static void main(String[] args) {
        Logger logger = Logger.getInstance();
        logger.log("Aplicación iniciada");
    }
}
```

3.Conexión a una base de datos

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConfig {
    private static DatabaseConfig instance;
    private Connection connection;

    private static final String URL = "jdbc:mysql://localhost:3306/mi_base_de_datos";
    private static final String USER = "usuario";
    private static final String PASSWORD = "password";

    public DatabaseConfig() {
        connect();
    }

    public static DatabaseConfig getInstance() {
        if (instance == null) {
            instance = new DatabaseConfig();
        }
        return instance;
    }

    public Connection getConnection() {
        try {
            if (connection == null || connection.isClosed()) {
                connect();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }
```

```java
    public void closeConnection() {
        if (connection != null) {
            try {
                connection.close();
                System.out.println("Conexión cerrada exitosamente.");
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    private void connect() {
        try {
            connection = DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

4. Conexión a una base de datos

```java
public class DatabaseConnection {
    private static DatabaseConnection instance;

    private DatabaseConnection() {
        System.out.println("Conexión a la base de datos creada.");
    }

    public static DatabaseConnection createInstance() {
        return new DatabaseConnection();
    }

    public static DatabaseConnection getInstance() {
        if (instance == null) {
            instance = createInstance();
        }
        return instance;
    }

    public void query(String sql) {
        System.out.println("Ejecutando consulta: " + sql);
    }
}

public class Main {
    public static void main(String[] args) {
        DatabaseConnection connection1 = DatabaseConnection.getInstance();
        connection1.query("SELECT * FROM usuarios");
    }
}
```

5. Configuración de un procesador de imágenes

```java
public class ImageProcessorConfig {
    private static ImageProcessorConfig instance;

    private String imageFormat;
    private int imageSize;
    private String outputDirectory;

    private ImageProcessorConfig() {
        this.imageFormat = "PNG";
        this.imageSize = 1024;
        this.outputDirectory = "/default/output/directory/";
    }

    public static ImageProcessorConfig getInstance() {
        if (instance == null) {
            instance = new ImageProcessorConfig();
        }
        return instance;
    }

    public String getImageFormat() {
        return imageFormat;
    }

    public void setImageFormat(String imageFormat) {
        this.imageFormat = imageFormat;
    }

    public int getImageSize() {
        return imageSize;
    }

    public void setImageSize(int imageSize) {
        this.imageSize = imageSize;
    }
```

```java
    public String getOutputDirectory() {
        return outputDirectory;
    }

    public void setOutputDirectory(String outputDirectory) {
        this.outputDirectory = outputDirectory;
    }
}

public class ImageProcessor {
    public void processImage(String imagePath) {
        ImageProcessorConfig config = ImageProcessorConfig.getInstance();

        System.out.println("Procesando imagen: " + imagePath);
        System.out.println("Formato de imagen: " + config.getImageFormat());
        System.out.println("Tamaño de imagen: " + config.getImageSize() + " pix
        System.out.println("Guardando imagen en: " + config.getOutputDirectory(

    }
}

public class Main {
    public static void main(String[] args) {
        ImageProcessorConfig config = ImageProcessorConfig.getInstance();
        config.setImageFormat("JPEG");
        config.setImageSize(800);
        config.setOutputDirectory("/images/processed/");

        ImageProcessor processor = new ImageProcessor();
        processor.processImage("path/to/image.jpg");

        System.out.println("Configuración actual:");
        System.out.println("Formato: " + config.getImageFormat());
        System.out.println("Tamaño: " + config.getImageSize());
        System.out.println("Directorio: " + config.getOutputDirectory());
    }
}
```