1. Los pilares de la programación orientada a objetos son: **encapsulamiento**, **herencia**, **polimorfismo** y **abstracción**.

2. Es correcto afirmar sobre el siguiente fragmento de código: `class Persona extends Auditoria { }`, que la palabra `extends` corresponde al pilar **herencia** de POO. ¿Este pilar establece que se pueden agregar cualquier cantidad de clases luego de `extends`, por ejemplo, `class Persona extends Auditoria, Usuario, Cliente { }`? Marque con (x) True( ) o False(x).

3. No es correcto afirmar sobre el siguiente fragmento de código: `class Persona implements Serializable { }`, que la palabra `implements` se utiliza para implementar una interfaz. Este mecanismo no permite que una clase implemente múltiples interfaces, por ejemplo, `class Persona implements Serializable, Cloneable, Comparable { }`. Marque con (x) True( ) o False(x). Adicionalmente, es obligatorio que una clase implemente todos los métodos definidos en una interfaz cuando esta es implementada. Responda con (x) True(x) o False( ).

4. Complete el siguiente fragmento de código: Indique el modificador de acceso adecuado para cada método (metodoUno, metodoDos, metodoTres, metodoCuatro) en la clase MiClase.

```
public class MiClase {
    private int atributoPrivado;

    private void metodoUno() {
        // Este método es accesible solo dentro de la clase MiClase.
        System.out.println("Método uno ejecutado.");
    }

    default void metodoDos() {
        // Este método es accesible dentro del mismo paquete.
        System.out.println("Método dos ejecutado.");
    }

    public void metodoTres() {
        // Este método es accesible desde cualquier clase.
        System.out.println("Método tres ejecutado.");
    }

    protected void metodoCuatro() {
        // Este método es accesible solo dentro de la misma clase y subclases.
        System.out.println("Método cuatro ejecutado.");
    }
}
```

5. Complete el siguiente fragmento de código:

```
public interface OperacionMatematica {
    Double operacion(Double n1, Double n2);
}
```

```
public class ImplementacionOperacion implements OperacionMatematica {
    @Override
    public Double operacion(Double n1, Double n2) {
        Double x = n1 * n2;
        return x;
    }

    Double x = operacion(2.0, 3.0);
    // X es igual a 15.0
}
```

6. Complete el siguiente fragmento de código:

```
// Clase abstracta para capturar datos por consola
import java.util.Scanner;
public abstract class CapturaDatos {
    private static final Scanner scanner = new Scanner(System.in);

    // Método para capturar texto
    public static double CapturarTexto(String mensaje) {
        System.out.print(mensaje + ": ");
        return scanner.nextLine();
    }

    // Método para capturar número
    public static    CapturarNumero(String mensaje) {
        while (true) {
            String input = CapturarTexto(mensaje);
            try {
                return Double.parseDouble(input);
            } catch (NumberFormatException e) {
                System.out.println("¡Error! Debe ingresar un número válido.");
            }
        }
    }
}


// Clase intermedia para la capa de funcionalidades
public class Funciones extends CapturaDatos {

    // Constructor por defecto
    public Funciones() {
        super(); // Llama al constructor de la clase base CapturaDatos
    }

}

// Clase principal
public class Main {
    public static void main(String[] args) {
```

```
        // Creación del objeto Funciones
        Funciones funciones = new Funciones();

        // Uso del método RealizarOperacion para obtener una cadena y un número
        String mensajeTexto = funciones.CapturarTexto("Ingrese un texto");
        Double mensajeNumero = funciones.CapturarNumero("Ingrese un número");

        // Imprimir resultados
        System.out.println("Texto ingresado: " + mensajeTexto);
        System.out.println("Número ingresado: " + mensajeNumero);
    }
}
```

7. Complete el siguiente fragmento de código:

```
class Estudiante {
    private String codigo;
    private String nombre;

    public Estudiante(String codigo, String nombre) {
        this.codigo = codigo;
        this.nombre = nombre;
    }

    public String getCodigo() {
        return codigo;
    }

    public String getNombre() {
        return nombre;
    }
}


import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // Crear una lista para almacenar estudiantes
        List<Estudiante> listaEstudiantes = new ArrayList<>();

        // Crear un objeto Scanner para capturar datos desde la consola
        Scanner scanner = new Scanner(System.in);

        // Pedir al usuario que ingrese los datos de 5 estudiantes
        for (int i = 0; i < 5; i++) {
            System.out.println("Ingrese el código del estudiante " + (i + 1) + ":
");
            String codigo = scanner.nextLine();
```

```
            System.out.println("Ingrese el nombre del estudiante " + (i + 1) + ":
");
            String nombre = scanner.nextLine();

            // Crear un objeto Estudiante con los datos ingresados y agregarlo a
la lista
            Estudiante estudiante = new Estudiante(codigo, nombre);
            listaEstudiantes.add(estudiante);
        }

        // Imprimir la lista de estudiantes
        System.out.println("Lista de estudiantes:");
        for (Estudiante estudiante : listaEstudiantes) {
            System.out.println("Código: " + estudiante.getCodigo() + ", Nombre: "
+ estudiante.getNombre());
        }
    }
}
```

8. Complete el siguiente fragmento de código:

```
iimport java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // Crear un HashMap para almacenar los estudiantes
        Map<String, String> estudiantes = new HashMap<>();

        // Crear un objeto Scanner para capturar datos desde la consola
        Scanner scanner = new Scanner(System.in);

        // Pedir al usuario que ingrese los datos de 5 estudiantes
        for (int i = 0; i < 5; i++) {
            System.out.println("Ingrese el código del estudiante " + (i + 1) + ":
");
            String codigo = scanner.nextLine();
            System.out.println("Ingrese el nombre del estudiante " + (i + 1) + ":
");
            String nombre = scanner.nextLine();

            // Agregar el estudiante al HashMap
            estudiantes.put(codigo, nombre);
        }

        // Imprimir la lista de estudiantes
        System.out.println("Lista de estudiantes:");
        for (Map.Entry<String, String> entry : estudiantes.entrySet()) {
            System.out.println("Código: " + entry.getKey() + ", Nombre: " +
entry.getValue());
        }
```

```
        }
    }
```