1. Se sabe que los pilares de la programación orientada a objetos son: _____, _____, _____ y _____

2. Es correcto afirmar sobre el siguiente fragmento de código: `class Persona extends Auditoria { }`, que la palabra `extends` corresponde al pilar _____ 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( ).

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( ). No es obligatorio que una clase implemente todos los métodos definidos en una interfaz cuando esta es implementada. Responda con (x) True( ) 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;
    _____ void metodoUno() {
        // Este método es accesible solo dentro de la clase MiClase.
        System.out.println("Método uno ejecutado.");
    }

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

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

    _____ 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 {
```

```
    __________
    public __________ operacion(_____ _____, _____ _____) {
        Double 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 _____________ 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
    ____________________________

}

// 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 = ____________________________________________;
```

```
        Double mensajeNumero = ____________________________________________;

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