ENCAPSULAMIENTO

Ejemplos de Escenarios Reales

Objetivos de la actividad

- Comprender el concepto de encapsulamiento en la programación orientada a objetos.
- Identificar situaciones del mundo real en las que el encapsulamiento es esencial.
- Evaluar si un escenario dado requiere encapsulamiento o no.

Instrucciones:

A continuación, se presentan ejemplos de escenarios reales, lee y analiza si es esencial el encapsulamiento o no, dependiendo de la situación.

Escenario 1. Aplicación de Gestión de Correos Electrónicos

Problema: En una aplicación de gestión de correos electrónicos, necesitas rastrear la dirección de correo electrónico de un usuario.

Pregunta. ¿De acuerdo con el problema planteado, analiza si correoElectronico necesita encapsulamiento o no?

- Si es necesario el encapsulamiento
- No es necesario el encapsulamiento

La respuesta correcta es: **“Si es necesario el encapsulamiento”**. Justificación: La dirección de correo electrónico es información privada y sensible. El encapsulamiento es necesario para proteger la privacidad del usuario y garantizar que solo se acceda y modifique de manera autorizada.

```java
private String correoElectrónico;
```

Escenario 2. Sistema de Registro de Eventos Públicos

Problema: En un sistema que registra eventos públicos, como conferencias o conciertos, necesitas rastrear la fecha de un evento.

Pregunta. ¿De acuerdo con el problema planteado, analiza si fechaEvento necesita encapsulamiento o no?

- Si es necesario el encapsulamiento
- No es necesario el encapsulamiento

La respuesta correcta es: **“No es necesario el encapsulamiento”**. Justificación: La fecha y hora del evento suelen ser información que se divulga ampliamente y no necesita protección especial. En este caso, el acceso público a esta información es apropiado y no hay necesidad de ocultarla.

```java
public Date fechaEvento;
```

Escenario 3. Sistema de Control de Inventario en una Tienda de Ropa

Problema: En un sistema de control de inventario en una tienda de ropa, necesitas rastrear la cantidad de existencias de un artículo de ropa.

Pregunta. ¿De acuerdo con el problema planteado, analiza si cantidadExistencias necesita encapsulamiento o no?

- Si es necesario el encapsulamiento
- No es necesario el encapsulamiento

La respuesta correcta es: **“Si es necesario el encapsulamiento”**. Justificación: La cantidad de existencias afecta la disponibilidad de productos para los clientes y la gestión de pedidos de reposición. El encapsulamiento es necesario para evitar que las existencias se actualicen de manera incorrecta y para garantizar un control adecuado del inventario.

```java
private int cantidadExistencias;
```

Escenario 4. Sistema de Control de Temperatura en una Industria de Alimentos

Problema. En un sistema de control de temperatura en una fábrica de alimentos, necesitas rastrear la temperatura de almacenamiento de un producto.

Pregunta. ¿De acuerdo con el problema planteado, analiza si temperaturaAlmacenamiento necesita encapsulamiento o no?

- Si es necesario el encapsulamiento
- No es necesario el encapsulamiento

La respuesta correcta es: **“Si es necesario el encapsulamiento”**. Justificación: La temperatura de almacenamiento es crítica para garantizar la seguridad y la calidad de los alimentos. El encapsulamiento es necesario para controlar la temperatura de manera precisa y evitar cambios que puedan comprometer la calidad del producto.

```java
private double temperaturaAlmacenamiento;
```

Escenario 5. Clase de Constantes Matemáticas

Problema. Suponga que tiene una clase donde las variables son constantes de matemáticas, por ejemplo, PI.

Pregunta. ¿De acuerdo con el problema planteado, analiza si las constantes de matemáticas necesitan encapsulamiento o no?

- Si es necesario el encapsulamiento
- No es necesario el encapsulamiento

La respuesta correcta es: **“No es necesario el encapsulamiento”**. Justificación: Las constantes matemáticas son valores conocidos y públicos que no deben cambiar. No hay necesidad de encapsular estas constantes, ya que se espera que sean accesibles públicamente.

Escenario 6. Sistema de Registro de Calificaciones en una Escuela

Problema. En un sistema de registro de calificaciones escolares, se necesita rastrear las calificaciones de un estudiante en una asignatura.

Pregunta. ¿De acuerdo con el problema planteado, analiza si calificación necesita encapsulamiento o no?

- Si es necesario el encapsulamiento
- No es necesario el encapsulamiento

La respuesta correcta es: **“Si es necesario el encapsulamiento”**. Justificación: Las calificaciones son datos sensibles y deben estar encapsuladas para garantizar la precisión y evitar la manipulación no autorizada.

```java
private double calificación;
```

Escenario 7. Sistema de Configuración de Software

Problema. En un sistema de configuración de software, se almacenan opciones de configuración, como el idioma, que los usuarios pueden modificar según sus preferencias.

Pregunta. ¿De acuerdo con el problema planteado, analiza si configuracionIdioma necesita encapsulamiento o no?

- Si es necesario el encapsulamiento
- No es necesario el encapsulamiento

La respuesta correcta es: **"No es necesario el encapsulamiento"**. Justificación: Las opciones de configuración son datos que los usuarios esperan modificar fácilmente. No es necesario encapsular esta información, ya que se supone que debe ser accesible y modificable públicamente.

```java
public String configuracionIdioma;
```

Escenario 8. Sistema de Registro de Eventos de Uso de Aplicaciones

Problema. En un sistema de registro de eventos de uso de aplicaciones como los registros de eventos y las acciones realizadas por los usuarios es indispensable tener acceso para el análisis.

Pregunta. ¿De acuerdo con el problema planteado, analiza si estadoLucer necesita encapsulamiento o no?

- Si es necesario el encapsulamiento
- No es necesario el encapsulamiento

La respuesta correcta es: **"No es necesario el encapsulamiento"**. Justificación: Los registros de eventos son esenciales para comprender el comportamiento de las aplicaciones y solucionar problemas. No es necesario ocultarlos.

```java
public List<Evento> registroEventos;
```

Clasificación

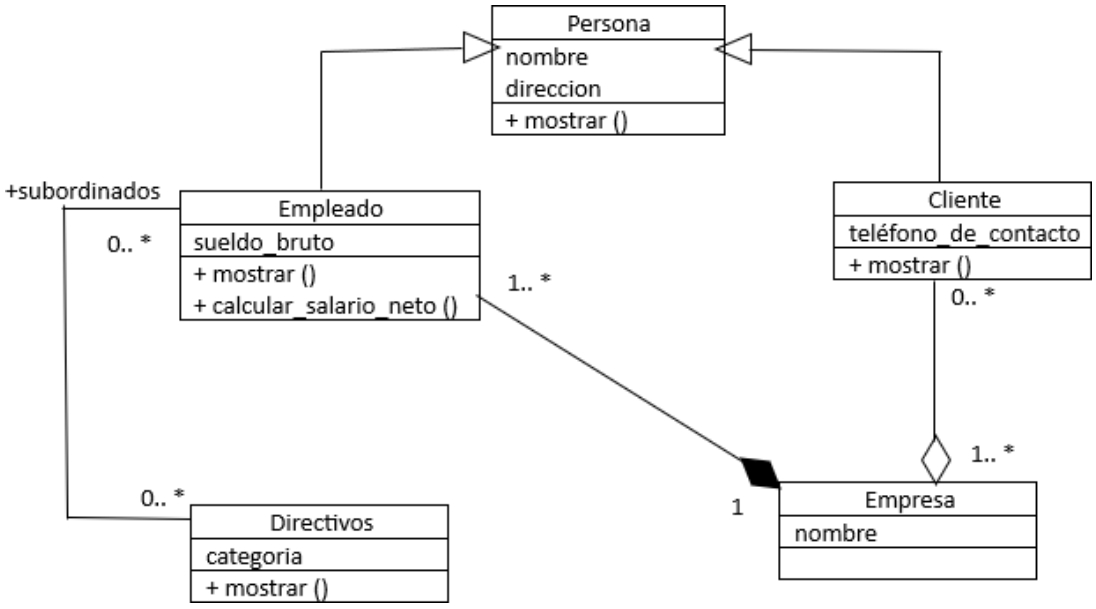
Objetivo

- Ayudar a los estudiantes a comprender y practicar la clasificación de propiedades en programación en función de su nivel de acceso, es decir, si deben ser declaradas como "public", "private" o "protected".

Instrucciones

De acuerdo con el siguiente diagrama, clasifica cada propiedad como "public", "private" o "protected" seleccionando la opción correspondiente. Piensa en la necesidad de proteger cierta información y a considerar qué propiedades deben ser accesibles desde fuera de la clase o el objeto y cuáles deben ser restringidas.

Diagrama.

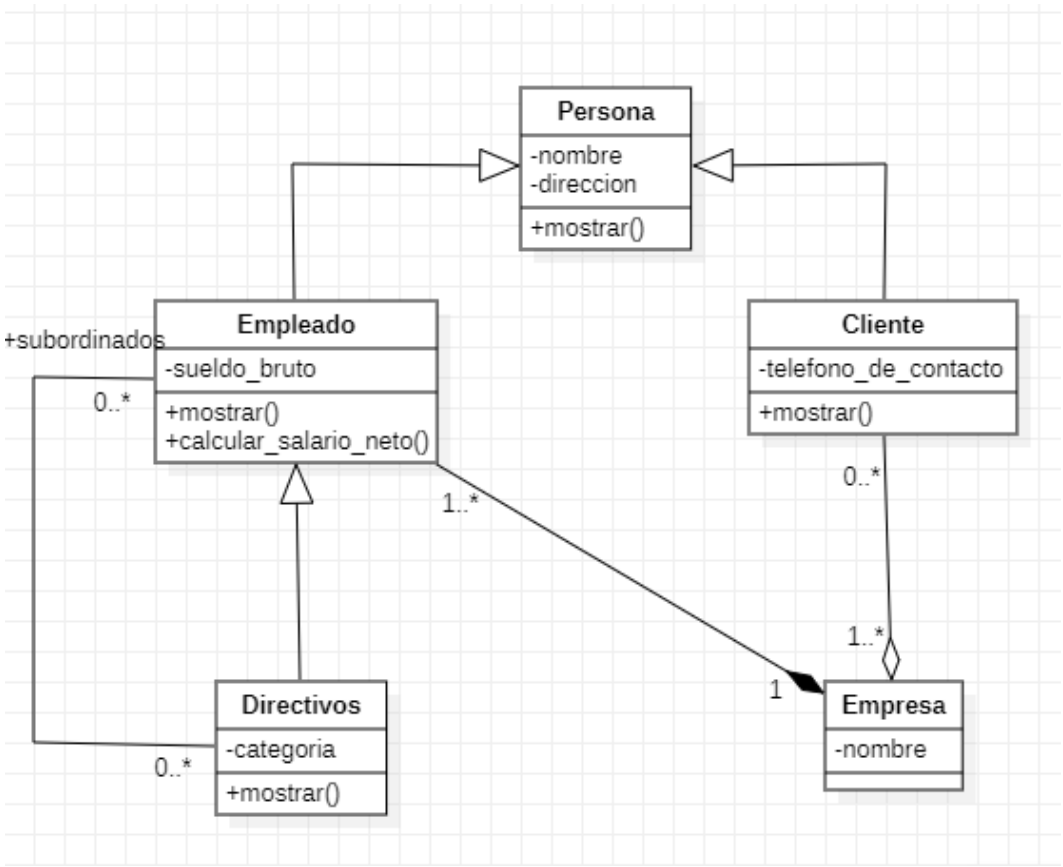


1. Nombre
    - Private
    - Public
    - Protected
2. Dirección
    - Private
    - Public
    - Protected

3. Sueldo_bruto
    - Private
    - Public
    - Protected
4. Teléfono_de_contacto
    - Private
    - Public
    - Protected

5. Categoría
    - Private
    - Public
    - Protected

Respuestas: Para este caso, todas las variables deben ser privadas(-) ya que la información es sensible la cual no debe ser accedida ni modificada desde fuera de la clase en la que se encuentra, por lo tanto debe ser declarada como "private". Esto significa que solo los métodos dentro de la misma clase pueden acceder a esta variable. Esto proporciona un alto nivel de protección para la dirección.



NOTAS:

Como se puede observar en este pequeño diagrama de clases hay herencia, nuestra superclase es Persona y las clasesHijas son Empleado y Cliente, RECUERDA que, aunque tus variables son privadas en la superclase, estas son heredadas automáticamente por las subclases, y para poder acceder a ella, necesitas hacerlo con los métodos getters and setters.

Cuestionario de conocimientos

Objetivo de la actividad

- Evaluar el nivel de conocimiento de los estudiantes sobre el tema de encapsulamiento a través de un cuestionario.

Instrucciones:

De acuerdo con los conocimientos adquiridos en este tema, responde las siguientes preguntas.

Pregunta 1. ¿Cuál es el concepto de Encapsulamiento?

a. El encapsulamiento es un principio de la programación orientada a objetos que consiste en ocultar el estado de los datos, con el fin de que solo se pueda cambiar mediante las operaciones definidas para el mismo.

b. El encapsulamiento se refiere a la creación de múltiples instancias de una clase.
c. El encapsulamiento se utiliza para heredar propiedades y métodos de una clase padre.

d. El encapsulamiento solo se aplica a las clases abstractas.

a. **Respuesta correcta: a.** El encapsulamiento es un principio de la programación orientada a objetos que consiste en ocultar el estado de los datos, con el fin de que solo se pueda cambiar mediante las operaciones definidas para el mismo.

Justificación: El encapsulamiento se refiere a ocultar los detalles internos de una clase y exponer una interfaz pública para interactuar con la clase.

Pregunta 2. ¿Cuál de las siguientes no es una razón para usar encapsulamiento?

a. Facilitar la reutilización de código.

b. Proteger los datos de una clase de modificaciones no autorizadas.

c. Mejorar el rendimiento del programa.

d. Ocultar la complejidad interna de una clase.

Respuesta correcta: c. Mejorar el rendimiento del programa.

Justificación: El encapsulamiento se utiliza principalmente para mejorar la seguridad y la organización del código, no para mejorar el rendimiento.

Pregunta 3. ¿Cuál de los siguientes modificadores de acceso permite el acceso desde cualquier parte del programa?

a. public

b. private

c. protected

d. internal

Respuesta correcta: a. public

Justificación: El modificador "public" permite el acceso desde cualquier parte del programa.

Pregunta 4. ¿Qué modificador de acceso restringe el acceso a los miembros de una clase solo a esa clase en particular?

a. public

b. private

c. protected

d. internal

Respuesta correcta: b. private

Justificación: El modificador "private" limita el acceso a los miembros solo dentro de la misma clase.

Pregunta 5. ¿Cuál es el propósito principal de un getter en una clase?

a. Establecer valores en propiedades privadas.

b. Obtener valores de propiedades privadas.

c. Crear nuevas instancias de la clase. d. Eliminar métodos de una clase.

Respuesta correcta: b. Obtener valores de propiedades privadas.

Justificación: Los getters se utilizan para obtener valores de propiedades privadas.