Universidad Autónoma del Estado de México

Centro Universitario UAEM Valle de Chalco

DESARROLLO

ASIGNATURA

Proyecto Integral de Comunicación de Datos

PROFESOR

Fabian Hernández Beciez

PRESENTA

Vázquez Silva Mariana

Galicia Vega Roberto Alejandro

Navarrete Arroyo Dulce Lucero

CARRERA

Ingeniería en Computación

CUVCH

Teoría

En el mundo de la programación, la comprensión de conceptos clave es esencial para el desarrollo de software efectivo y eficiente. Entre estos conceptos, la herencia, el encapsulamiento y el polimorfismo desempeñan un papel fundamental en la programación orientada a objetos. Si bien estos pueden ser difíciles de asimilar a través de la lectura de explicaciones en texto plano.

La herencia, el encapsulamiento y el polimorfismo son conceptos que adquieren vida propia cuando se los ve en acción. Requieren una comprensión práctica para dominarlos completamente, ya que su aplicación en el mundo real es esencial para el desarrollo de aplicaciones y sistemas de software robustos.
En esta página educativa de programación, hemos optado por utilizar videos en lugar de una teoría puramente textual para enseñar estos conceptos por varias razones importantes.

1. Naturaleza Práctica de los Temas: Herencia, encapsulamiento y polimorfismo son conceptos fundamentales en la programación orientada a objetos que a menudo se entienden mejor a través de ejemplos y aplicaciones prácticas. Los videos pueden mostrar la implementación de estos conceptos en código real y en contextos reales de programación, lo que facilita su comprensión.

2. Demostración en Tiempo Real: Los conceptos de programación se vuelven más tangibles cuando se demuestran en tiempo real. Los videos permiten a los estudiantes ver cómo funcionan estos conceptos mientras se desarrolla el código, lo que puede aclarar dudas y hacer que los conceptos sean más accesibles.

3. Interacción con Elementos Visuales: Al trabajar con estos conceptos, a menudo se utilizan diagramas, esquemas y ejemplos visuales para ilustrar relaciones de herencia, niveles de acceso y aplicaciones de polimorfismo. Los videos pueden incorporar elementos visuales de manera efectiva, ayudando a los estudiantes a comprender estos conceptos de manera más completa.

4. Mejor Retención de Información: Los videos pueden combinar múltiples estímulos sensoriales, como imágenes, sonido y movimiento, lo que puede aumentar la retención de información. Los estudiantes pueden recordar y aplicar más fácilmente lo que han visto y experimentado en un video en comparación con un texto teórico

Tema	Enlace
Herencia	https://youtu.be/S8ayMhvF02Q
Encapsulamiento	https://youtu.be/-w3Cy8ASBxQ
Polimorfismo	https://youtu.be/hS_APCOSk9o

Actividad de Clasificación

Objetivo de la actividad:

- Comprender el concepto de herencia.
- Identificar y clasificar objetos en categorías basadas en sus características compartidas.

Instrucciones:

Deben clasificar los siguientes objetos en categorías basadas en las características que comparten. Por ejemplo, podrían agrupar los animales en "animales domésticos" y "animales salvajes". Selecciona la posible respuesta correcta:

- Automóvil
- Avión
- Helicóptero
- Coyote
- Puma
- Jaguar
- Motocicleta
-
- Bicicleta
- Elefante
- Cebra
- Avioneta
- Koala

NOTA: Al momento de seleccionar la posible respuesta, el alumno tendrá 4 posibles respuestas o clasificaciones las cuales serían:

- Transporte Terrestre
- Transporte Aéreo
- Animales carnívoros
- Animales herbívoros

CLASIFICACIONES

Transporte terrestre

- Automóvil
- Motocicleta
- Bicicleta

Transporte aéreo

- Avión
- Helicóptero
- Avioneta

Animales carnívoros

- Coyote
- Jaguar
- Puma

Animales herbívoros

- Elefante
- Cebra
- Koala

Retroalimentación.

La lista que se otorgó al principio contaba con animales y medios de transporte al momento de clasificar se pudo realizar de diferentes formas, por ejemplo: animales terrestres o acuáticos, animales salvajes o domésticos, de la misma manera transporte ya sea para carga o personas, por su capacidad o tamaño. Esto es solo un ejemplo, recuerda que la clasificación la debes de realizar de acuerdo con las características que comparten entre ellos.

Juegos de Clasificación

Objetivo principal

- Que los estudiantes comprendan las relaciones de herencia en la programación orientada a objetos y sean capaces de identificar y clasificar clases como "Clase Base" o "Clase Derivada" en un contexto práctico.

Instrucciones:

Dadas las siguientes clases, analiza y determina cuales serían clases base o también conocidas como clase padre y clase derivada o también conocida como clase hija y selecciona la opción correcta.

1. Clase Pez
    - Clase base
    - Clase derivada
2. Clase Animal
    - Clase base
    - Clase derivada
3. Clase Perro
    - Clase base
    - Clase derivada
4. Clase Vehículo
    - Clase base
    - Clase derivada
5. Clase Motocicleta
    - Clase base
    - Clase derivada
6. Clase Bicicleta
    - Clase base
    - Clase derivada
7. Clase Cuadrado
    - Clase base
    - Clase derivada
8. Clase Triangulo
    - Clase base
    - Clase derivada
9. Clase Figura Geométrica
    - Clase base
    - Clase derivada
10. Clase Vendedor
    - Clase base
    - Clase derivada

11. Clase Empleado
    - Clase base
    - Clase derivada
12. Clase Gerente
    - Clase base
    - Clase derivada

RESPUESTAS

1. Clase Pez
   Clase Derivada
2. Clase Animal
   Clase Base
3. Clase Perro
   Clase Derivada

Retroalimentación

- Clase Base (Padre): "Animal" Puede contener propiedades y métodos comunes a todos los animales, como nombre, edad y método de alimentación.
- Clase Derivada (Hija): "Perro": Un perro es un tipo específico de animal que hereda las propiedades y métodos de la clase base "Animal." Puede agregar atributos adicionales
- Clase Derivada (Hija): "Pez": Los peces son un grupo de animales acuáticos que tienen propiedades comunes, por lo que se puede crear una clase derivada "Pez" que herede propiedades generales de la clase base "Animal." Además, puede agregar atributos específicos como tipo de aleta y métodos relacionados con la natación.

4. Clase Vehículo
   Clase Base
5. Clase Motocicleta
   Clase Derivada
6. Clase Bicicleta
   Clase Derivada

Retroalimentación

- Clase Base (Padre): "Vehículo" Puede contener propiedades y métodos comunes a todos los vehículos, como velocidad máxima y capacidad de pasajeros.
- Clase Derivada (Hija): "Motocicleta" Una motocicleta es un tipo específico de vehículo que hereda las propiedades y métodos de la clase base "Vehículo."

Puede agregar atributos adicionales como el tipo de motor y métodos para cambiar de marcha.

- Clase Derivada (Hija): "Bicicleta" Una bicicleta es un tipo específico de vehículo que hereda algunas de las propiedades generales de la clase base "Vehículo," como la capacidad de transporte. Además, puede agregar atributos específicos como el número de ruedas, el tipo de pedales y métodos relacionados con la dirección y el pedaleo.

7. Clase Cuadrado
   Clase Derivada
8. Clase Triangulo
   Clase Derivada
9. Clase Figura Geométrica
   Clase Base

Retroalimentación

- Clase Base (Padre): "Figura Geométrica" La clase base "Figura Geométrica" puede contener propiedades y métodos comunes a todas las figuras geométricas, como el área y el perímetro.
- Clase Derivada (Hija): "Triángulo" Un triángulo es un tipo específico de figura geométrica que hereda las propiedades y métodos de la clase base "Figura Geométrica." Puede agregar atributos adicionales como la longitud de sus lados y métodos para calcular el tipo de triángulo (equilátero, isósceles, escaleno).
- Clase Derivada (Hija): "Cuadrado" Un cuadrado es un tipo particular de rectángulo en el que todos los lados son iguales. Hereda las propiedades y métodos generales de la clase base "Figura Geométrica," pero puede agregar atributos adicionales como la longitud de un lado y métodos específicos para calcular el área y el perímetro de un cuadrado.

10. Clase Vendedor
    Clase Derivada
11. Clase Empleado
    Clase Base
12. Clase Gerente
    Clase Derivada

Retroalimentación

- Clase Base (Padre): "Empleado" La clase base "Empleado" puede contener propiedades y métodos comunes a todos los empleados, como nombre, número de empleado y salario base.
- Clase Derivada (Hija): "Gerente" Un gerente es un tipo específico de empleado que hereda las propiedades y métodos de la clase base "Empleado." Puede agregar atributos adicionales como el departamento que supervisa y métodos para aprobar o rechazar solicitudes de vacaciones.
- Clase Derivada (Hija): "Vendedor" Un vendedor es un tipo específico de empleado que hereda las propiedades generales de la clase base "Empleado," como nombre y salario base. Además, puede agregar atributos adicionales como las ventas realizadas y métodos relacionados con el seguimiento de las ventas y el cálculo de comisiones.

Cuestionario de conocimientos

Objetivo General

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

Instrucciones:

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

Pregunta 1: ¿Cuál es la definición de herencia en programación orientada a objetos?

a) El proceso de recibir bienes de un familiar fallecido.

b) La capacidad de una clase de adquirir las propiedades y métodos de otra clase.

c) Una forma de pagar impuestos sobre propiedades.

d) Un tipo de inversión en bienes raíces.

Retroalimentación: **La respuesta correcta es la opción "b".** La herencia en programación orientada a objetos permite que una clase adquiera las propiedades y métodos de otra clase.

Pregunta 2: ¿Qué son las clases base y derivadas en herencia?

a) Las clases base son clases hijas, y las clases derivadas son clases padre.

b) Las clases base son clases que no pueden heredar, y las clases derivadas son las que heredan.

c) Las clases base son las que heredan propiedades, y las clases derivadas son las que no heredan.

d) Las clases base son las clases padre, y las clases derivadas son las que heredan de ellas.

Retroalimentación**: La respuesta correcta es la opción "d"**. Las clases base son las clases padres de las cuales se hereda en las clases derivadas.

Pregunta 3: ¿Cómo se heredan los métodos y las propiedades en la jerarquía de clases?

a) Los métodos se heredan automáticamente, pero las propiedades deben copiarse manualmente.

b) Los métodos y las propiedades se heredan automáticamente.

c) Los métodos y las propiedades deben reescribirse en cada clase derivada.

d) Los métodos se heredan solo si se declara explícitamente, y las propiedades se heredan automáticamente.

Retroalimentación: **La respuesta correcta es la opción "b".** Tanto los métodos como las propiedades se heredan automáticamente en la jerarquía de clases.

Pregunta 4: ¿Cuál es una de las ventajas clave de la herencia en programación orientada a objetos?

a) Aumenta la complejidad del código.

b) Permite la creación de objetos sin clases.

c) Promueve la reutilización de código.

d) Requiere menos memoria.

Retroalimentación**: La respuesta correcta es la opción "c"**. La herencia promueve la reutilización de código al permitir que las clases hereden propiedades y métodos de otras clases.

Pregunta 5: ¿Cuál de las siguientes es un ejemplo de relación de herencia?

a) Un coche y una bicicleta.

b) Un gato y un perro.

c) Una clase "Animal" y una clase "Mamífero".

d) Un teléfono y un ordenador.

Retroalimentación**: La respuesta correcta es la opción "c".** La relación entre "Animal" y "Mamífero" es un ejemplo de herencia en programación orientada a objetos.

Pregunta 6: ¿Cuál es el propósito principal de las clases base en la herencia?

a) Heredar propiedades de las clases derivadas.

b) Proporcionar una implementación completa de una clase.

c) Proporcionar una base común para las clases derivadas.

d) Evitar la creación de objetos.

Retroalimentación: **La respuesta correcta es la opción "c".** Las clases base proporcionan una base común para las clases derivadas.

Pregunta 7 ¿Cuál es la principal diferencia entre una clase base y una clase derivada en herencia?

a) La clase base es más importante que la clase derivada

b) La clase base contiene más atributos

c) La clase derivada se crea a partir de la clase base

d) No hay diferencia entre ellas

Retroalimentación: **La respuesta correcta es la opción c)** La clase derivada se crea a partir de la clase base. La clase derivada se define utilizando la clase base como punto de partida, agregando o modificando atributos y métodos según sea necesario.

Pregunta 8 ¿De qué otra forma se le conoce a una clase base?

a) Clase primaria
b) Clase secundaria
c) Super Clase
d) Mega Clase

Retroalimentación: **La respuesta correcta es la opción "c".** Las clases base también son conocidas como Clase Padre, Super clase.

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

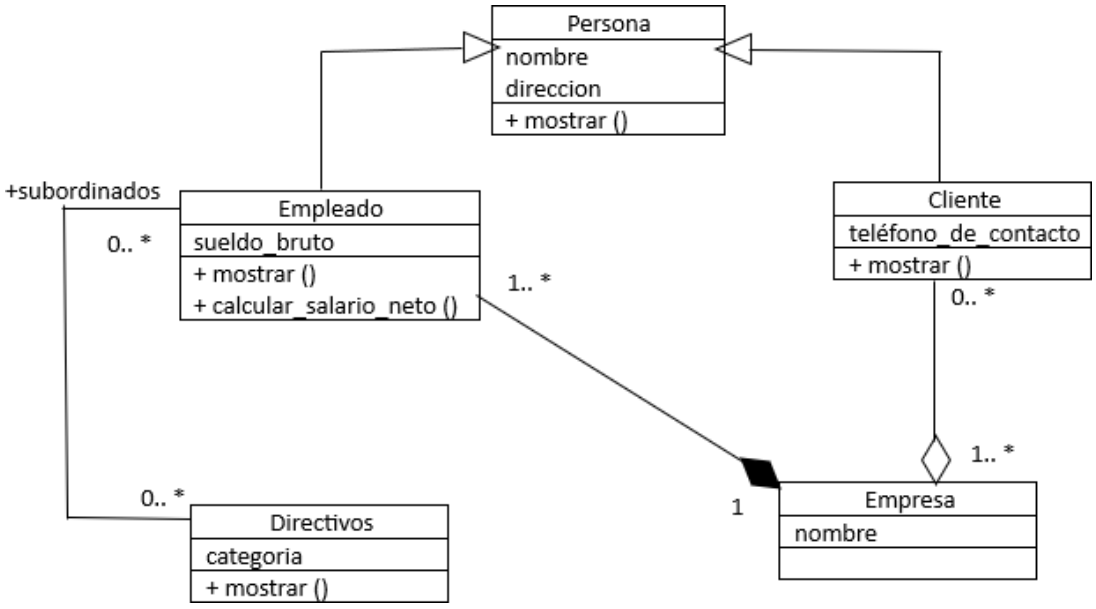
Objetivo

- 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.

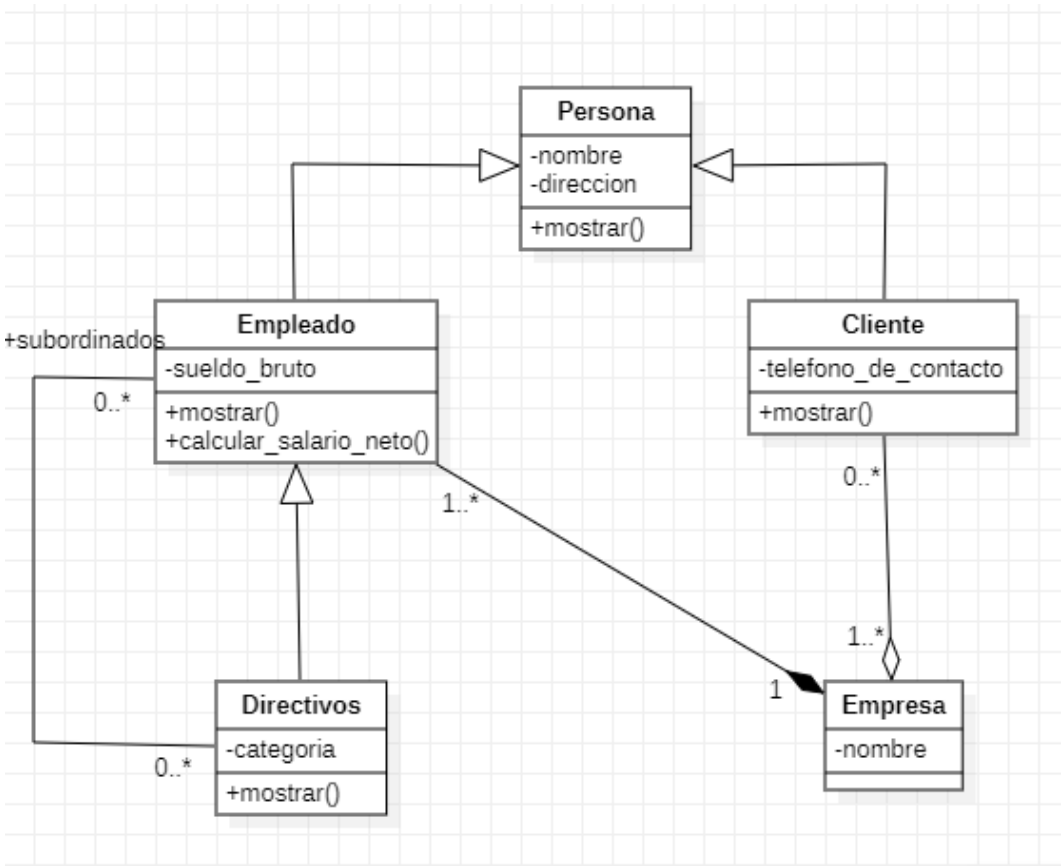


1. Nombre
    - Private
    - Public
    - Protected
2. Dirección
    - Private
    - Public
    - Protected

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.

Actividades Polimorfismo

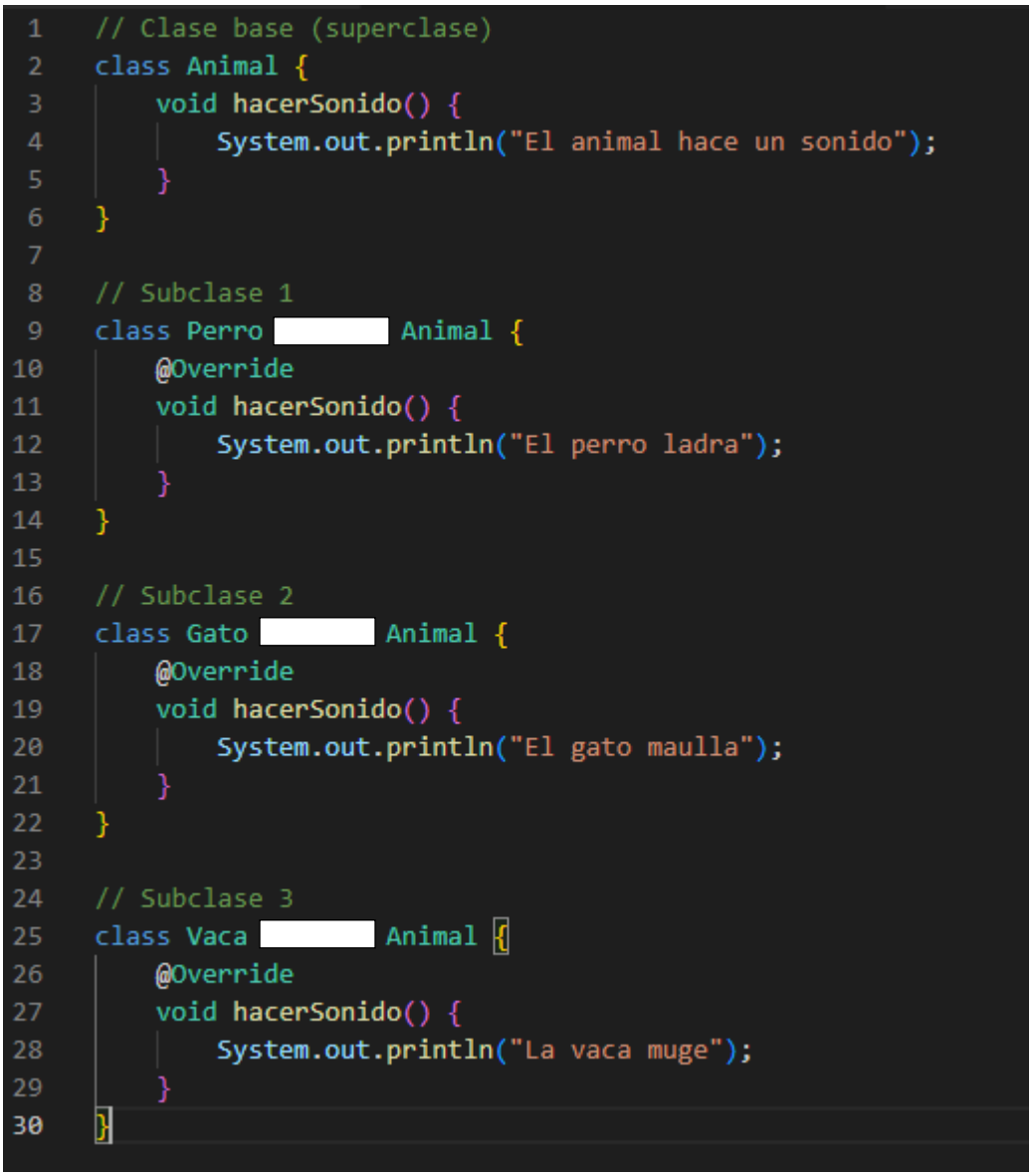
Completar el código

Objetivo:

- Que los alumnos apliquen y demuestren su comprensión del concepto de polimorfismo en programación orientada a objetos, desarrollando la habilidad de implementar este principio de manera efectiva al completar y adaptar código proporcionado para que funcione correctamente.

Instrucciones:

- Coloca en los espacios vacíos la palabra clave para que el código funcione correctamente:



```
1   // Clase base (superclase)
2   class Animal {
3       void hacerSonido() {
4           System.out.println("El animal hace un sonido");
5       }
6   }
7
8   // Subclase 1
9   class Perro          Animal {
10      @Override
11      void hacerSonido() {
12          System.out.println("El perro ladra");
13      }
14  }
15
16  // Subclase 2
17  class Gato           Animal {
18      @Override
19      void hacerSonido() {
20          System.out.println("El gato maulla");
21      }
22  }
23
24  // Subclase 3
25  class Vaca           Animal {
26      @Override
27      void hacerSonido() {
28          System.out.println("La vaca muge");
29      }
30  }
```

La respuesta correcta sería: “Extends”, ya que gracias a esa palabra sabemos que está heredando a la superclase “Animal”, y la función “hacerSonido” podrá utilizarse en las subclases. Esto dando a entender que el Perro es un Animal y emite un sonido, así como el gato, pero aunque son sonidos, son distintos, uno ladra mientras el otro maulla.

El código se vería de la siguiente manera:

```java
// Clase base (superclase)
class Animal {
    void hacerSonido() {
        System.out.println("El animal hace un sonido");
    }
}

// Subclase 1
class Perro extends Animal {
    @Override
    void hacerSonido() {
        System.out.println("El perro ladra");
    }
}

// Subclase 2
class Gato extends Animal {
    @Override
    void hacerSonido() {
        System.out.println("El gato maulla");
    }
}

// Subclase 3
class Vaca extends Animal {
    @Override
    void hacerSonido() {
        System.out.println("La vaca muge");
    }
}
```

Identificar código.

Objetivo

- Que los alumnos apliquen y demuestren su comprensión del concepto de polimorfismo en programación orientada a objetos.

Instrucciones:

Siguiendo el código anterior la variable ‘miAnimal’ se declara un objeto de tipo Animal, pero en tiempo de ejecución, se le asignan instancias de las subclases Perro, Gato y Vaca. A través del polimorfismo, la llamada al método hacerSonido() se resuelve dinámicamente en función del tipo de objeto real al que se refiere la variable en ese momento, lo que permite que cada objeto haga su propio sonido. Esto es un ejemplo de polimorfismo.

```
public class Main {
    public static void main(String[] args) {
        Animal miAnimal;

        miAnimal = new Perro();
        miAnimal.hacerSonido();

        miAnimal = new Gato();
        miAnimal.hacerSonido();

        miAnimal = new Vaca();
        miAnimal.hacerSonido();
    }
}
```

Teniendo esto en cuenta, ¿Cuál sería la salida de este código?

Respuestas:

- El Perro Ladra.
- El Gato Maulla.
- La Vaca Muge.

Cuestionario

1. ¿Qué es el polimorfismo en programación orientada a objetos?

   a) Una técnica para ocultar datos.

   b) Una técnica para organizar el código.

   c) Una técnica para tratar objetos de diferentes clases de manera uniforme

2. ¿Cuál de las siguientes palabras clave se usa en Java para permitir el polimorfismo?

   a) polymorph

   b) override

   c) extends

3. ¿Cuál es el beneficio principal del polimorfismo en la programación?

   a) Aumenta la complejidad del código.

   b) Facilita la reutilización del código.

   c) Hace que los objetos sean menos flexibles.