Tp13

TP N° 13 - Colaboración y Herencia

1. ¿Qué es la herencia y cuál es su principal objetivo en la programación orientada a objetos?

2. ¿Cuál es la importancia de la encapsulación en el contexto de la herencia?

3. Describe los tipos de visibilidad (public, private, protected) en el contexto de herencia.

4. ¿Qué es una especialización?

5. ¿Cuáles son las ventajas y los posibles problemas que pueden surgir al utilizar la herencia múltiple?

Herencia en POO: La herencia es un principio de la programación orientada a objetos (POO) que permite crear una nueva clase basada en una clase existente. La nueva clase (subclase) hereda las propiedades y métodos de la clase base (superclase), lo que facilita la reutilización de código y la creación de una jerarquía de clases. El principal objetivo de la herencia es la reutilización de código y la creación de relaciones jerárquicas entre clases, promoviendo la extensión de comportamientos sin necesidad de reescribir el código.

Importancia de la encapsulación en la herencia: La encapsulación es el principio de ocultar los detalles internos de una clase y exponer solo lo necesario a través de interfaces públicas. En el contexto de la herencia, la encapsulación permite que una subclase no acceda directamente a los atributos y métodos privados de su superclase. Solo los miembros públicos y protegidos de la superclase pueden ser utilizados o modificados en la subclase. Esto ayuda a mantener la integridad y seguridad de los datos, asegurando que las subclases no puedan modificar el estado interno de la superclase de manera no controlada.

Tipos de visibilidad en herencia:

Public: Los miembros de la clase (atributos o métodos) marcados como public son accesibles desde cualquier parte del código, incluidas las subclases y las clases externas.

Private: Los miembros marcados como private no son accesibles ni por la propia clase ni por las subclases. Son exclusivos para la clase en la que están definidos.

Protected: Los miembros protected son accesibles en la propia clase y en sus subclases, pero no desde clases externas. Esto permite que las subclases puedan acceder a ciertos atributos o métodos sin exponerlos públicamente.

Especialización: La especialización es el proceso mediante el cual una subclase agrega o modifica características específicas de su superclase para adaptarse mejor a un contexto particular. En lugar de redefinir todo el comportamiento de una clase base, la subclase puede especializar o ampliar sus propiedades y métodos para cumplir con sus necesidades particulares. Esto se logra mediante la herencia, donde la subclase puede añadir nuevos atributos o métodos o modificar los existentes.

Ventajas y problemas de la herencia múltiple: Ventajas:

Permite la reutilización de código desde varias clases base, lo que puede reducir la cantidad de código redundante.

Facilita la creación de clases más complejas combinando características de diferentes clases base.

Problemas:

Si dos o más clases base tienen métodos o propiedades con el mismo nombre, puede ser confuso determinar cuál se debe usar en la subclase. Esto puede llevar a errores difíciles de detectar.

La herencia múltiple puede aumentar la complejidad de la jerarquía de clases, haciendo que el código sea más difícil de entender y mantener.

A medida que se agregan más clases base, los cambios en una clase base pueden afectar inesperadamente a las subclases, lo que puede generar efectos secundarios difíciles de predecir.

1. Explicar el funcionamiento del programa con sus palabras.

2. Completar los comentarios indicados con lineas punteadas: // ......

3. ¿Cuál es el nombre de la clase padre en la jerarquía?

4. ¿Qué clase hereda de la clase Persona?

5. ¿Cuál es la relación entre las clases Estudiante y Tecnico?

6. ¿Cuál es el propósito del constructor de Empleado?

7. ¿Cuál es el propósito del método mostrarPersona()?

8. ¿Qué método se utiliza para mostrar el salario del empleado?

El programa simula una jerarquía de clases relacionadas con personas, específicamente un Empleado, un Estudiante y un Técnico. La clase base es Persona, que tiene atributos comunes como nombre y edad y un método mostrarPersona() que imprime estos atributos.

Las clases Empleado y Estudiante heredan de Persona, añadiendo atributos y métodos específicos para cada tipo. Empleado tiene un atributo adicional sueldo y un método mostrarEmpleado() para mostrar la información del empleado junto con su sueldo. Estudiante tiene un atributo notaFinal y un método mostrarEstudiante() para mostrar la información del estudiante junto con su nota final.

La clase Tecnico hereda de Estudiante, añadiendo un atributo especialidad y un método mostrarTecnico() que despliega la información del estudiante, junto con su especialidad como técnico.

El main() crea instancias de Empleado, Estudiante y Tecnico, mostrando los detalles de cada uno utilizando los métodos correspondientes.

en la clase persona:

private:

string nombre;

int edad;

public:

Persona(string, int); // Constructor para inicializar nombre y edad.

void mostrarPersona(); // Método para mostrar nombre y edad.

en la clase empleado

private:

float sueldo;

public:

Empleado(string, int, float); // Constructor para inicializar nombre, edad y sueldo.

void mostrarEmpleado(); // Método para mostrar información del empleado.

En la clase estudiante

private:

float notaFinal;

public:

Estudiante(string, int, float); // Constructor para inicializar nombre, edad y nota final.

void mostrarEstudiante(); // Método para mostrar información del estudiante.

En la Clase tecnico

private:

string especialidad;

public:

Tecnico(string, int, float, string); // Constructor para inicializar nombre, edad, nota final y especialidad.

void mostrarTecnico(); // Método para mostrar información del técnico.

La clase padre en la jerarquía es Persona.

Las clases que heredan de Persona son Empleado y Estudiante.

Tecnico es una clase derivada de Estudiante, lo que significa que un Técnico es un tipo específico de Estudiante. La clase Tecnico hereda los atributos y métodos de Estudiante, pero añade un nuevo atributo que es especialidad.

El constructor de la clase Empleado tiene como propósito inicializar los atributos nombre, edad(a través del constructor de la clase base Persona) y sueldo del objeto Empleado.

El método mostrarPersona() tiene como propósito mostrar la información básica de una persona, es decir, el nombre y la edad. Este método es heredado por las clases derivadas para mostrar los detalles de la persona y puede ser utilizado para imprimir la información de un Empleado, Estudiante o Tecnico.

El método utilizado para mostrar el salario del empleado es mostrarEmpleado(), que llama al método mostrarPersona() para mostrar la información básica y luego muestra el salario con el atributo sueldo.