DESARROLLO TALLER JAVA #7

Jahnear Isaiah Faiquare Vizcaíno

1123620500

G1

Punto 1 Ejercicio de análisis

a) Explique por qué la clase Instrumento debe definirse como abstracta y qué la diferencia de una clase normal:

La clase Instrumento debe definirse como abstracta porque:

Representa un concepto general: "Instrumento" es una idea abstracta que no tiene una implementación concreta por sí misma, no se puede usar un "Instrumento" genérico directamente, se necesita un tipo específico como un "Piano" o una "Guitarra".

Contiene métodos como Tocar() y Afinar() sin implementación, lo que obliga a las subclases a proporcionar su propia lógica.

Previene la instanciación directa, una clase abstracta asegura que no puedan crearse instancias de ella, ya que no tiene un comportamiento completo por sí misma.

Diferencia respecto a una clase normal:

Una clase normal puede instanciarse directamente y debe tener todos sus métodos completamente implementados.

Una clase abstracta sirve como plantilla y no puede ser instanciada directamente, está puede contener métodos abstractos (sin implementación) que las subclases deben definir.

b) Elabore una clase denominada Piano heredada de Instrumento, añada un constructor y redefina para ella los métodos Tocar y Afinar.

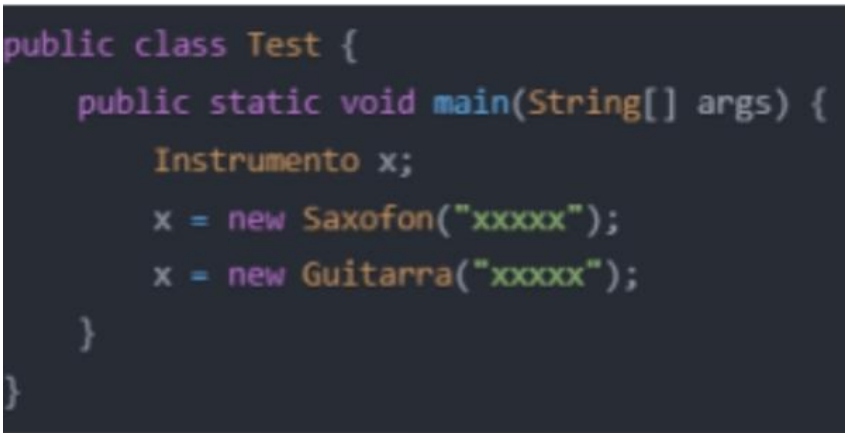
```
public class Piano extends Instrumento {

    // Constructor que llama al constructor
de la clase base
    public Piano(String tipo) {
        super(tipo); // Inicializa el
atributo 'tipo' definido en la clase
Instrumento
    }

    // Implementación del método Tocar
    @Override
    public void Tocar() {
        System.out.println("Tocando Piano");
    }

    // Implementación del método Afinar
    @Override
    public void Afinar() {
        System.out.println("Afinando Piano");
    }
}
```

Explicación:

Constructor: El constructor de Piano inicializa el atributo tipo de la clase base Instrumento mediante super(tipo).

Métodos sobrescritos: Tocar() y Afinar() proporcionan comportamientos específicos para el piano.

c) ¿Señale cuál de las siguientes líneas no se pueden ejecutar y por qué?



```
public class Test {
    public static void main(String[] args) {
        Instrumento x;
        x = new Saxofon("xxxxx");
        x = new Guitarra("xxxxx");
    }
}
```

En mi opini todas las líneas en el código son válidas porque:

Instrumento x; declara una referencia de tipo abstracto, lo cual es permitido.

x = new Saxofon("xxxxx"); y x = new Guitarra("xxxxx"); son válidas porque Saxofon y Guitarra son clases concretas que implementan todos los métodos abstractos de Instrumento.

Podrían presentarse errores en caso tal se intentara instanciar Instrumento directamente con x = new Instrumento("xxxxx");, generaría un error porque Instrumento es abstracta.

d) ¿Qué imprime el programa?

Imprime:

Tocando Saxofon

Tocando Guitarra

La referencia x de tipo Instrumento apunta primero a un objeto de tipo Saxofon. Cuando se llama a x.Tocar(), se ejecuta la implementación de Tocar() definida en la clase Saxofon. Luego, x apunta a un objeto de tipo Guitarra. Al llamar a x.Tocar(), se ejecuta la implementación de Tocar() de la clase Guitarra.

Punto 2:

a. La clase Estrella dejaría de ser abstracta al no tener métodos abstractos, permitiendo crear objetos de este tipo. Las clases hijas Nova y SuperNova seguirían heredando de Estrella sin problemas, pero ya no estarían obligadas a sobrescribir el método explotar(), ya que ahora tendría una implementación base en Estrella.

b. Una clase abstracta puede incluir métodos concretos que brinden funcionalidades comunes a sus subclases. Por ejemplo, el método tipoCuerpo2() imprime "Extrasolar" para todos los objetos, y getID() permite acceder al identificador privado ID, asegurando una funcionalidad compartida entre todas las subclases.

c. La clase Estrella no implementa el método descripcion() porque, como clase abstracta, no tiene la obligación de hacerlo. En su lugar, son las subclases concretas las responsables de definir este método para cumplir con el contrato de la superclase.

d. El arreglo oa puede referenciar una clase abstracta porque, gracias al polimorfismo, puede almacenar objetos de sus subclases concretas. Al usar el método descripcion() en la línea 25, se ejecuta la versión específica del método según el tipo real de cada objeto en el arreglo.

e. La línea 29 imprime “Soy una Super Nova” porque el objeto en oa[0] fue sobrescrito por uno de tipo SuperNova en la línea anterior.

f. La clase Estrella no implementa descripcion() porque, al ser abstracta, puede heredar métodos abstractos sin necesidad de definirlos.

g. Si tipoCuerpo1() en la clase Galaxia es privado, se genera un error de compilación porque los métodos abstractos deben implementarse con una visibilidad igual o más accesible que la original. Si es privado, las subclases no pueden heredarlo, lo que contradice su propósito.

h. La clase Nova no define tipoCuerpo1() porque lo hereda de Estrella, pero puede sobrescribirlo si desea personalizar su comportamiento, siempre que mantenga la compatibilidad en parámetros y retorno.

i. El método toString se hereda de Object, la clase base de todas las clases en Java, lo que permite llamarlo sin necesidad de redefinirlo.

j. Aunque las clases abstractas no pueden instanciarse directamente, se pueden crear referencias de tipo abstracto que apunten a objetos de subclases concretas, ya que estas referencias solo indican una dirección en memoria.

k. Las opciones A y B son válidas debido al polimorfismo. La opción C genera un error porque intenta instanciar una clase abstracta, lo cual no está permitido.

l. En A, se puede crear el objeto correctamente porque tanto el objeto como su referencia son válidos.

   En B, funciona porque una referencia de una clase base puede apuntar a un objeto de una subclase.

   En C, si el método explotar() no está en la clase base, el compilador no puede garantizar su existencia, causando un error.

   En D, no hay problema porque oa apunta a un objeto de tipo Nova, y explotar() está definido en esta clase.

m. La línea 15 imprime true porque el objeto referenciado por obN es de tipo Nova, que hereda de ObjetoAstronomicoExtraSolar, Si obN apuntara a un objeto no relacionado, como Galaxia, imprimiría false. Las líneas 16, 17, y 18 imprimen true, ya que las relaciones de herencia coinciden. Las líneas 19 y 20 imprimen false porque los objetos no son instancias de las clases esperadas.