Carlos Galvis C.C 1097492903

Ejercicio de análisis

1. Explique por qué la clase Instrumento debe definirse como abstracta y qué la distingue de una clase común en el ejemplo siguiente:

Respuesta: La razón para definir la clase Instrumento como abstracta es que actúa como una plantilla para las subclases. Si se tratara de una clase normal, no sería posible implementar una versión genérica de los métodos en Instrumento, ya que cada subclase tiene su propio comportamiento específico.

1. Elabore una clase denominada Piano que herede de Instrumento, agregue un constructor y redefina los métodos Tocar y Afinar.
2. ¿Señale cuál de las siguientes líneas no se pueden ejecutar y por qué?

Respuesta: El código no contiene errores. Aunque no se puede instanciar directamente una clase abstracta, es posible declarar variables de su tipo. Gracias al polimorfismo, una variable de tipo Instrumento puede apuntar a cualquier objeto de una de sus subclases y reasignarse a diferentes instancias de estas subclases.

1. ¿Qué imprime el siguiente programa?

Respuesta: Imprime "Tocando Saxofón" y "Tocando Guitarra" en ese orden debido a la reasignación de la variable y la ejecución del método tocar().

1. Ejercicio de código en el repositorio

* ¿Qué sucede si se define el método explotar() de la clase Estrella como se indica a continuación? Explique su respuesta.

Respuesta: Los métodos abstractos no pueden tener implementación. Si se intenta proporcionar un cuerpo al método al añadir { ... }, se genera un error.

1. ¿Qué significa que los métodos tipoCuerpo2() y getID() de la clase ObjetoAstronomicoExtraSolar no se definan como abstractos? ¿Se podría considerar un error?

Respuesta: Al no ser abstractos, estos métodos pueden tener un cuerpo, lo que implica que las subclases no están obligadas a sobrescribirlos, aunque podrían hacerlo si necesitan un comportamiento distinto.

1. Si se define como abstracta la clase ObjetoAstronomicoExtraSolar, ¿se puede considerar un error que no tenga métodos abstractos?

Respuesta: No es un error. Una clase abstracta puede usarse para representar una idea genérica que no debe instanciarse directamente. Declarar una clase como abstracta asegura que no se pueda crear una instancia de ella, incluso si no tiene métodos abstractos.

1. Explique por qué el arreglo oa (línea 16) hace referencia a una clase abstracta y, sin embargo, en la línea 25 se invoca el método correspondiente a cada clase derivada.

Respuesta: Aunque ObjetoAstronomicoExtraSolar es una clase abstracta, Java permite crear arreglos de este tipo porque el arreglo en sí no es una instancia directa de la clase abstracta. Cada elemento del arreglo contiene una referencia a un objeto de una clase concreta (como Galaxia, Nova o SuperNova), lo que permite invocar métodos específicos de cada clase gracias al polimorfismo.

1. ¿Por qué la línea 26 imprime “Soy una Super Nova” si el arreglo oa en esa posición fue inicializado con un objeto de tipo Galaxia?

Respuesta: Esto ocurre porque el elemento del arreglo fue reasignado a un objeto de tipo SuperNova antes de invocar el método descripcion().

1. ¿Por qué en la clase Estrella no se define el método descripcion() si la superclase lo está solicitando como un método abstracto?

Respuesta: La clase Estrella es declarada como abstracta, por lo que no está obligada a implementar los métodos abstractos heredados de su superclase.

1. ¿Qué sucede si el método tipoCuerpo1() de la clase Galaxia se define como privado? ¿Por qué se genera error?

Respuesta: El método tipoCuerpo1() es abstracto en la clase base ObjetoAstronomicoExtraSolar, lo que obliga a las subclases a implementarlo con al menos la misma visibilidad. Definirlo como privado genera un error porque rompe esta regla.

1. ¿Por qué la clase Nova no define el método tipoCuerpo1()? ¿Podría definirlo? Si lo define, ¿qué implica esta situación?

Respuesta: La clase Nova no necesita definir el método tipoCuerpo1() porque ya está implementado en Estrella. Sin embargo, podría hacerlo si requiere personalizar su implementación.

1. ¿Qué imprime la línea 6? ¿Por qué se puede llamar al método toString() si la clase Galaxia no lo define y su superclase ObjetoAstronomicoExtraSolar tampoco?

Respuesta: Imprime algo similar a "taller7.Galaxia@433c675d". Esto sucede porque todas las clases en Java heredan implícitamente de la clase Object, que define el método toString().

1. ¿Por qué en la línea 11 se puede crear un puntero obN de tipo ObjetoAstronomicoExtraSolar si esta es una clase abstracta?

Respuesta: Esto es posible porque no se está creando un objeto de tipo ObjetoAstronomicoExtraSolar, sino que se está utilizando una referencia de este tipo para apuntar a un objeto concreto.

1. ¿Las instrucciones en las líneas B y C son válidas? Explique.

Respuesta:

* Línea B: Incorrecta. No se pueden crear instancias directas de una clase abstracta.
* Línea C: Correcta. Las referencias de una clase base pueden apuntar a objetos de sus subclases.

1. Explique por qué la instrucción en la línea B es correcta y la de la línea C es incorrecta. Omita la línea C, ¿qué se imprime por pantalla? Explique.

Respuesta:

* Línea B: Incorrecta. Una clase abstracta no puede ser instanciada.
* Línea C: Incorrecta. La referencia nova no está definida porque fue reemplazada por la referencia oa.

Se imprime: "Boom!" debido a la ejecución del método explotar() en el objeto Nova.

1. ¿Por qué la línea 15 imprime true? ¿Para cualquier objeto creado siempre imprimirá lo mismo? ¿Qué imprimen las siguientes líneas? Explique.

Respuesta: La línea 15 imprime true porque obN es una referencia que apunta a un objeto de una clase que hereda de Object. El operador instanceof devuelve false si la referencia es null.

* System.out.println(obN instanceof Object): True si obN no es null.
* System.out.println("" + obN instanceof Object): Error de compilación.

1. Agregue el siguiente constructor. ¿Se genera error? ¿Se pueden colocar constructores en clases abstractas? ¿Qué sentido tiene?

Respuesta: No se genera error. Es válido incluir constructores en clases abstractas para inicializar valores comunes que serán utilizados por las subclases.

1. Suponga que agrega la clase EnanaBlanca como se indica a continuación. ¿Qué se puede concluir de esta situación? ¿Qué errores se presentan y cómo podría corregirlos?

Respuesta: Si no se implementan los métodos abstractos heredados, se generan errores. Para corregirlo, es necesario implementar dichos métodos en la clase EnanaBlanca y proporcionar la funcionalidad requerida.