Java Taller Nro. 7

By: Andrés Felipe Muñoz Ortiz

Documento: 1011395924

Grupo: 01

Solución:

Punto 1:

a. La clase Instrumento debe ser abstracta porque es como el molde, la base para crear otras clases que sean instrumentos específicos, no se puede crear un objeto en teoría que sea solo instrumento, un instrumento son las características comunes que tienen las subclases, que pueden ser piano, saxofón, etc. Clases normales que de existir si se podrían instanciar y tendrían sentido, serían instrumentos con todas sus características y tendrían las suyas particulares

b. class Piano extends Instrumento {
private String marca;
public Piano(String marca) {
this.marca = marca;
}
@Override
public void tocar() {
System.out.println("Tocando el piano de la marca " + marca + ": Do, Re, Mi"); }
@Override
public void afinar() {
System.out.println("Afinando el piano de la marca " + marca + ": Probando con los acordes mayores"); }
}

c.

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

Este código no genera errores, todas la líneas de pueden ejecutar, instrumento x solo le da el tipo al apuntador, no instancia la clase abstracta, y la 4 y 5 hacen que X apunte a objetos de subclases de instrumentos, además de que en las respectivas subclases hay constructores que reciben los parámetros dados, entonces por esto y todo lo anterior el código debería funcionar sin problemas

d. Simplemente imprime:
   Tocando el saxofón de tipo: xxxxx
   Tocando la guitarra de tipo: xxxxx

   Pues la sobrescritura de estos métodos es muy sencilla y solo devuelve los parámetros que se dieron

Punto 2:

a. La clase Estrella dejaría de ser abstracta porque ya no tiene métodos abstractos. Esto permitiría crear objetos de tipo Estrella, lo que no es posible cuando la clase es abstracta, las clases hijas Nova y SuperNova seguirían funcionando pues pueden ser herederas de estrella aun si no es abstracta, pero ya no estarían obligadas a redefinir el método explotar(), ya que este ya tendría una implementación en Estrella, y entonces pueden usar el método heredado, no están obligadas a redefinirlo, y lo mismo con explotar( )

b. Esta situación no es un error porque no todos los métodos en una clase abstracta necesitan ser abstractos, esta puede tener métodos concretos que proporcionen una funcionalidad común que todas las subclases puedan compartir. Por ejemplo tipoCuerpo2( ) imprime "Extrasolar" para todos los objetos y getID( ) permite obtener el identificador privado ID, proporcionando una funcionalidad que es igual para todas las subclases.

c. La clase Estrella no define descripcion() porque, al ser abstracta, no está obligada a heredar cosas, elige lo que quiere y no heredar. Esta responsabilidad recae en las clases hijas concretas, que deben implementar el método para cumplir con el contrato definido por la superclase.

d. El arreglo oa puede referenciar una clase abstracta porque almacena objetos de sus clases derivadas concretas, gracias al polimorfismo; el arreglo oa puede guardar objetos de clases que heredan de la clase abstracta, gracias al polimorfismo. Al usar descripcion() en la línea 25, se ejecuta el método propio de cada objeto según su tipo real

e. La línea 29 imprime "Soy una Super Nova" porque oa[0] fue sobrescrito con el objeto 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 el método tipoCuerpo1() en la clase Galaxia se define como privado, el programa generará un error de compilación porque el método tipoCuerpo1() está declarado como abstracto en la clase base ObjetoAstronomicoExtraSolar, y las subclases deben implementar los métodos abstractos con una visibilidad igual o más accesible que la de la

declaración original, al ser abstracto la idea es que lo hereden, pero si es privado no se podría, lo que es paradogico

h. La clase Nova no define tipoCuerpo1() porque ya lo hereda de Estrella, pero podría redefinirlo para personalizar su comportamiento siempre y cuando los parámetros usados y el retorno sea compatible

i. Porqué el método toString es heredado de Object, la clase madre de todas la clases, que va primero en jerarquía y por esto puede ser llamado aunque no haya sido definido (redefinido)

j. porqué en Java aunque no se pueden instanciar directamente las clases abstractas, sí se pueden crear punteros de tipo de una clase abstracta y esto es completamente válido porque una referencia simplemente apunta a un objeto en memoria, y las clases abstractas pueden servir como tipos base para objetos de clases concretas que las extienden

k. La A y B son validas por polimosfismo, pero la C se intenta instancia una clase abstracta, lo que no es posible y genera error

l. en A se puede crear perfectamente el objeto pues tanto este como su pauntador, son correctos, y B también pues un puntero de una clase superior lo apunta, entonces esta bien; pero en C aunque el objeto apuntado por oa es de tipo Nova, la referencia es de tipo ObjetoAstronomicoExtraSolar. Si el método explotar() no está definido en la clase ObjetoAstronomicoExtraSolar, el compilador no puede garantizar que oa tiene acceso a ese método, por lo que marca un error, y en D no hay ningún problema porque oa realmente apunta a un objeto de tipo Nova y se puede llamar al método explotar() porque está definido en la clase Nova

m. Imprime True porque el objeto apuntado por obN es de tipo Nova, que hereda de ObjetoAstronomicoExtraSolar, pero para otros objeto no necesariamente imprime lo mismo, la salida de instanceof depende de la relación entre el tipo de referencia y el tipo del objeto apuntado, si obN apuntara a un objeto de otra clase no relacionada con ObjetoAstronomicoExtraSolar, como Galaxia, la línea 15 imprimiría false; la línea 16 17 y 18 imprimen True, mientras que las 19 y 20 imprimen false por lo ya explicado anteriormente