Parte 2 taller 7, Java
Samuel Botero Rivera

a)
La clase Instrumento debe definirse como abstracta porque representa un concepto genérico que no puede instanciarse directamente, ya que no tiene una implementación concreta de los métodos Tocar y Afinar. Estos métodos son abstractos porque cada instrumento (como Saxofón o Guitarra) tiene una forma específica de ejecutarlos. A diferencia de una clase normal, una clase abstracta actúa como plantilla, forzando a las subclases a implementar los métodos abstractos para proporcionar un comportamiento específico, mientras que una clase normal puede ser instanciada directamente y no necesita contener métodos abstractos.

b)
```
public class Piano extends Instrumento {
    public Piano(String tipo) {
        super(tipo);
    }

    @Override
    public void Tocar() {
        System.out.println("metodo redefinido");
    }

    @Override
    public void Afinar() {
        System.out.println("metodo redefinido.");
    }
}
```

c)
todas las lineas son validas

d)
Tocando Saxofon
Tocando Guitarra

2.

a) ocurre un error al compilar el código. Un método abstracto está diseñado específicamente para ser una plantilla que las subclases deben implementar, por lo que no puede tener un cuerpo o implementación.

b) Los métodos tipoCuerpo2() y getID() en la clase ObjetoAstronomicoExtraSolar no están definidos como abstractos porque ya tienen una implementación proporcionada. En otras palabras, estos métodos no requieren que las clases hijas los implementen de manera obligatoria, ya que ya están completos en la clase base.
c) si una clase abstracta no tiene métodos abstractos, simplemente está cumpliendo el rol de ser una clase base para otras clases. Las clases hijas pueden heredar la implementación de los métodos concretos (como tipoCuerpo2() y getID(), pero no están obligadas a sobrescribirlos, ya que esos métodos ya tienen comportamiento definido.
d) aunque el tipo de la referencia en el arreglo oa es ObjetoAstronomicoExtraSolar, el polimorfismo permite que se invoque el método adecuado según el tipo concreto de cada objeto almacenado en el arreglo. Esto es posible porque las clases derivadas sobrescriben el método descripcion(), y el método que se ejecuta corresponde al tipo real del objeto (no al tipo de la referencia).
e) Aunque originalmente oa[0] contenía una instancia de Galaxia, el reemplazo de oa[0] por una instancia de SuperNova en la línea anterior hace que, en la siguiente invocación de descripcion(), se ejecute el método correspondiente a SuperNova
f) esto porque una clase abstracta no está obligada a implementar los métodos abstractos de su superclase. Las clases hijas (no abstractas) de la clase estrella sí deben implementar el método.
g) Cuando una subclase sobrescribe un método de la superclase, la visibilidad del método en la subclase no puede ser más restrictiva que la visibilidad del método en la superclase (la cual implicitamente es de visibilidad pública).
h) no lo define porque ya su superclase lo definió, así que hereda esta implementación a menos que quiera definir otra, con ayuda del @Override
i) imprimirá una cadena que describe la referencia del objeto g en el formato por defecto proporcionado por la implementación del método toString(). Este método es heredado de la clase Object por lo que es posible llamarlo sin que se haya definido en las clases presentes.
j) esto porque en Java, puedes utilizar una referencia de tipo de clase base (incluso si es abstracta) para apuntar a objetos de cualquier subclase de esa clase base. lo que se conoce como polimorfismo. cabe destacar que al hacer esto no se crea una instancia de la clase.
k) B no es válida, ya que no es posible crear una instancia de una clase abstracta. C es válida ya que nova es subclase de la clase abstracta ObjetoAstronomicoExtraSolar
l) La línea C es incorrecta porque el método explotar() no está definido en la clase ObjetoAstronomicoExtraSolar, clase del apuntador oa, por lo que no es accesible desde una referencia de ese tipo, incluso si el objeto al que apunta

tiene ese método definido. Si se omite esta línea. se realiza un casting explícito y lo que se imprime es “boom!”

m) obN es una referencia a un objeto de tipo ObjetoAstronomicoExtraSolar. En Java, todas las clases heredan implícitamente de la clase Object, ya sea directa o indirectamente. por lo tanto imprime True. cuando obN no apunta a nigun objerto (null) se imprime False

n) No, no se genera error. En Java, es válido definir constructores en una clase abstracta. Aunque no se puede instanciar directamente una clase abstracta. Aunque no se puede instanciar una clase abstracta directamente, las subclases que la extienden llamarán implícitamente al constructor de la clase abstracta mediante super() al momento de su creación. Esto asegura que cualquier lógica de inicialización definida en la clase abstracta se ejecute antes de que la subclase complete su propio constructor.

o) La clase EnanaBlanca extiende la clase abstracta Estrella, pero no implementa todos los métodos abstractos heredados de Estrella y de ObjetoAstronomicoExtraSolar. Esto genera errores de compilación porque una clase concreta debe implementar todos los métodos abstractos de sus superclases. Para corregir estos errores, se deben implementar los métodos abstractos descripcion() y explotar() en la clase EnanaBlanca