Respuestas

Ejercicio de análisis

a) La clase Instrumento se define como abstracta porque:

Tiene métodos abstractos:
Los métodos Tocar() y Afinar() no tienen implementación en la clase Instrumento. Una clase con al menos un método abstracto debe ser declarada como abstracta.

Diferencia con una clase normal:

- **Clase abstracta**: No puede ser instanciada directamente y puede tener métodos abstractos que deben ser implementados por sus subclases.
- **Clase normal**: Puede ser instanciada directamente y no contiene métodos abstractos.

b)

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

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

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

c) El código no compila porque los métodos Tocar() y Afinar() en las clases Saxofon y Guitarra están marcados como abstract. No se puede instanciar una clase abstracta. Por ende no compila por las líneas dónde se crean nuevas instancias.

d) Tocando Saxofon
Tocando Guitarra

Ejercicio de análisis

1. El código genera error de compilación ya que se intenta definir un método abstracto.

2. Los métodos tipoCuerpo2() y getID() en ObjetoAstronomicoExtraSolar no se definen como abstractos porque tienen una implementación concreta. Esto no es un error. De hecho, permite a las subclases heredar funcionalidad sin necesidad de redefinirla, manteniendo flexibilidad.

3. Es válido definir una clase abstracta sin métodos abstractos. Esto puede hacerse para evitar la creación directa de instancias de la clase y obligar a que las subclases la implementen.

4. Aunque oa es un arreglo de tipo ObjetoAstronomicoExtraSolar, se pueden almacenar instancias de sus subclases. En la línea 25, la invocación del método utiliza polimorfismo, ejecutándose la versión específica del método implementada en cada subclase.

5. En la línea 29, se asigna oa[2] (un objeto SuperNova) a oa[0]. Por polimorfismo, al invocar descripcion(), se ejecuta el método de SuperNova.

6. La clase Estrella no necesita implementar descripcion() porque es abstracta, lo que permite delegar esa responsabilidad a sus subclases (Nova y SuperNova).

7. Definir tipoCuerpo1() como privado genera un error, ya que un método sobrescrito debe tener igual o mayor visibilidad que en la clase padre. tipoCuerpo1() es abstracto en ObjetoAstronomicoExtraSolar y tiene visibilidad protected o public.

8. Nova hereda tipoCuerpo1() de Estrella. Podría sobrescribirlo, pero no es obligatorio si no necesita un comportamiento específico.

9. La llamada a toString() en la línea 9 imprime algo como Galaxia@hashcode. Esto se debe a que todas las clases en Java heredan de Object, que define el método toString().

10. Se puede crear un puntero de tipo ObjetoAstronomicoExtraSolar porque se refiere a una subclase concreta, en este caso Nova. Sin embargo, no se puede instanciar directamente una clase abstracta.

11. **B**: Inválido. No se pueden instanciar clases abstractas.
    C: Válido. Una referencia de tipo abstracto puede apuntar a un objeto de una subclase concreta.

12. **B: Correcta.** Se permite asignar un objeto de tipo Nova a una referencia de tipo ObjetoAstronomicoExtraSolar por herencia.
    C: Incorrecta. El método explotar() no está definido en ObjetoAstronomicoExtraSolar, lo que genera un error de compilación.

Impresión sin C: Al ejecutar **D**, se realiza un cast a Nova, permitiendo llamar a explotar(). Esto imprime: Boom!.

13. La línea 15 imprime true porque todas las instancias en Java son objetos. Sin embargo, si obN es null, obN instanceof Object imprimirá false.

14. Es válido definir constructores en clases abstractas. Estos se invocan mediante super() en las subclases para inicializar atributos comunes.

15. Genera un error porque no implementa el método abstracto explotar(). Para corregirlo, se debe definir.