STIVEN SANTIAGO ROSERO QUEMAG

1. Ejercicio de análisis

a) Explique por qué la clase Instrumento debe definirse como abstracta y qué la diferencia de una clase normal, en el ejemplo siguiente:

La clase Instrumento es abstracta porque no tiene sentido crear un objeto "Instrumento" ya que todos los instrumentos tienen características y comportamientos específicos.

Al definirla como abstracta, estoy obligando a que cualquier clase que herede de Instrumento como Saxofón o Guitarra implemente los métodos Tocar() y Afinar(), que son comunes a todos los instrumentos pero se comportan de manera diferente para cada uno.

Así, Instrumento se convierte en una especie de plantilla o base para crear clases de instrumentos específicos, en lugar de ser una clase que se pueda instanciar directamente.

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 {

    public Piano(String tipo) { super(tipo); }
    public void Tocar() {System.out.println("Tocando Piano ");}
    public void Afinar() {System.out.println("Afinando Piano");}
}
```

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

```
x=new Guitarra("xxxxx");
```

Esto se debe a que la variable x ya había sido asignada con un objeto de tipo Saxonfon en la línea anterior, y asignarle un nuevo objeto de tipo Guitarra sobreescribe el objeto Saxonfon creado anteriormente.

d) ¿Qué imprime el siguiente programa?

Tocando Saxonfon
Tocando Guitarra

2. Ejercicio de código en el repositorio

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

El método explotar ya no se define como abstracto porque le coloqué cuerpo (System.out.println("Estrella explotar");)

b.¿Qué significa que los métodos tipoCuerpo2() y getID() de la clase ObjetoAstronomicoExtraSolar, no se definan como abstract? ¿Podría considerarse esta situación un error?

Significa que estos métodos tienen una implementación concreta y no necesitan ser sobrescritos por las subclases.

No, no es un error. Una clase abstracta puede tener métodos concretos y abstractos

c. Si se define como abstracta la clase ObjetoAstronomicoExtraSolar, como se indica a continuación, ¿puede considerarse un error definir una clase abstracta sin métodos abstractos? Explique

No, no es un error.Una clase abstracta puede tener métodos concretos y no necesariamente métodos abstractos. La clase abstracta se utiliza para agrupar comportamientos comunes y puede ser utilizada como base para otras clases.En este caso, la clase ObjetoAstronomicoExtraSolar tiene métodos concretos que pueden ser utilizados

directamente por las subclases, y también puede ser utilizada como base para otras clases que necesiten agregar comportamientos específicos

d. Explique por qué el arreglo oa (línea 19) hace referencia a una clase abstracta y sin embargo, en la línea 25 se invoca el método correspondiente a cada clase derivada.

Se debe a que el arreglo oa es de tipo ObjetoAstronomicoExtraSolar y utiliza polimorfismo, por lo que al invocar el método descripcion() se ejecuta la implementación específica de cada clase derivada Galaxia, Nova, SuperNova.

e. ¿Por qué la línea 29 imprime “Soy una Super Nova” sabiendo que el arreglo oa en esa posición fue inicializado con un objeto de tipo Galaxia?
La línea 29 imprime 'Soy una Super Nova' porque en la línea 28 se reasigna el objeto en la posición 0 del arreglo oa, reemplazando el objeto Galaxia inicial por el objeto SuperNova de la posición 2, por lo que se ejecuta la implementación de la clase SuperNova.

f. ¿Por qué en la clase Estrella no se define el método descripcion() si la superclase lo está solicitando, ya que en este método descripción en abstracto?
La clase Estrella no define el método descripción() porque es una clase abstracta y sus subclasses concretas Nova y SuperNova son las que implementan este método, cumpliendo así con la declaración abstracta en la superclase ObjetoAstronomicoExtraSolar

g. ¿Qué sucede si el método tipoCuerpo1() de la clase Galaxia se define como privado? ¿Por qué se genera error?
Si el método tipoCuerpo1() de la clase Galaxia se define como privado, se generará un error de compilación porque no se puede sobreescribir un método público declarado en la superclase ObjetoAstronomicoExtraSolar con un método privado.

h. ¿Por qué la clase Nova no define el método tipoCuerpo1()? ¿Se podría definir? Si lo define, ¿qué interpreta de esta situación?
La clase Nova no define el método tipoCuerpo1() porque su superclase Estrella ya lo implementa. Sin embargo, sí podríamos definirlo si lo deseamos, sobreescribiendo la implementación de su superclase

i. ¿Qué imprime la línea 9? ¿Por qué se puede llamar al método toString() si la clase Galaxia no lo define y su papá ObjetoAstronomicoExtraSolar tampoco?
La línea 9 imprime la descripción en cadena de la instancia de Galaxia porque, aunque Galaxia no define toString(), hereda el método toString() de la clase Object,
j. ¿Por qué en la línea 11 se puede crear un puntero obN de tipo ObjetoAstronomicoExtraSolar si esta es una clase abstracta?
Se puede crear un puntero obN de tipo ObjetoAstronomicoExtraSolar en la línea 11 porque, aunque ObjetoAstronomicoExtraSolar es abstracta, se puede utilizar como tipo de referencia para objetos de clases que heredan de ella, como Galaxia o Estrella

k. ¿Las siguientes instrucciones (instrucciones en las líneas B y C) son válidas? Explique

B.Inválida: ObjetoAstronomicoExtraSolar es una clase abstracta y no se puede instanciar directamente.

C. Válida: Nova hereda de ObjetoAstronomicoExtraSolar, por lo que se puede asignar una instancia de Nova a una variable de tipo ObjetoAstronomicoExtraSolar.

l. Explique por qué (ver código a continuación) la siguiente instrucción en la línea B es correcta y la instrucción en la línea C es incorrecta. Omitiendo la instrucción en la línea C, ¿qué se imprime por pantalla? Explique su respuesta.
La instrucción en la línea B es correcta porque Nova hereda de ObjetoAstronomicoExtraSolar.
La instrucción en la línea C es incorrecta porque ObjetoAstronomicoExtraSolar no tiene el método explotar().

m.¿Por qué la línea 15 imprime true? ¿Para cualquier objeto que se cree siempre imprimirá lo mismo? ¿Qué imprimen las siguientes líneas? ¿Por qué?
La línea 15 imprime "true" porque cualquier objeto en Java hereda de Object.
Si obN es null:
System.out.println(obN instanceof Object); imprime false
System.out.println("" + obN instanceof Object); imprime false, no "null" porque instanceof tiene prioridad sobre el operador +
n. Agregue el siguiente constructor. ¿Se genera error? ¿Se pueden colocar constructores a clases abstractas? ¿Qué sentido tiene?
No se genera error. Sí, se pueden colocar constructores en clases abstractas para:
Inicializar variables ((enlace no disponible) = 4)
Llamar métodos (this.tipoCuerpo2()

O. Suponga que agrega la clase EnanaBlanca como se indica a continuación. ¿Qué se puede concluir de esta nueva situación? ¿Qué errores se presentan y cómo podría corregirlos?
No hay errores de sintaxis en el código.
Sin embargo, hay una posible inconsistencia:
EnanaBlanca extiende Estrella, pero en el contexto del problema original, EnanaBlanca debería extender ObjetoAstronomicoExtraSolar, ya que es un tipo de objeto astronómico extra solar