TALLER 7 PREGUNTAS

1. EJERCICIO DE ANALISIS

a) Explique por qué la clase Instrumento debe definirse como abstracta y qué la diferencia de una clase normal, en el ejemplo siguiente: Porque al interior de la clase instrumento se están creando métodos abstractos, por lo tanto, la clase debe ser abstracta.

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 el piano");
    }
    public void Afinar() {
        System.out.println("Afinando el piano");
    }
}
```

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

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

Todas las líneas se podrán ejecutar ya que el único problema que podría haber es que no se pueden crear instancias de tipo Instrumentos ya que es una clase abstracta, pero en este código lo que se hace es que se declara es una variable x de tipo Instrumento mas no una instancia por lo que se puede ejecutar sin problemas.

d) ¿Qué imprime el siguiente programa?

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

Imprime:

Tocando Saxofon

Tocando Guitarra

2. Ejercicio de código en el repositorio

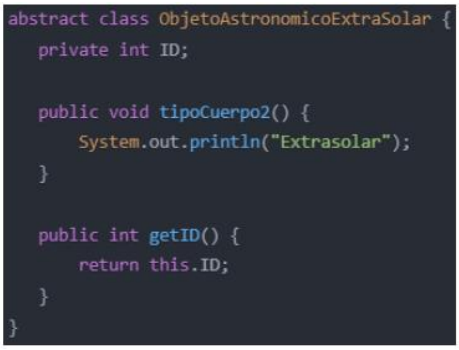
a. ¿Qué sucede si se define el método explotar() de la clase Estrella como se indica a continuación? Explique su respuesta.

Saldrá un error ya que un método abstracto no lo podemos definir porque este no debe tener implementación en la clase abstracta, solo en sus subclases.

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

No se considera un error ya que simplemente se están creando métodos que si tienen implementación en la propia clase, estos al no ser abstractos no necesariamente deben de ser usados por sus subclases, pero si son heredados.

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.



```
abstract class ObjetoAstronomicoExtraSolar {
    private int ID;

    public void tipoCuerpo2() {
        System.out.println("Extrasolar");
    }

    public int getID() {
        return this.ID;
    }
}
```

No se considera un error, porque, aunque una clase abstracta sea como una plantilla para sus subclases, no es obligatoria que tenga métodos abstractos, lo que es útil aquí es que no se podrán instanciar objetos de esta clase ya que es abstracta.

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.

Porque cada elemento del arreglo tiene un tipo mas especifico y esto permite el uso de la ligadura dinámica.

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?

Porque en la línea 28 se está cambiando el apuntador y esa posición en el arreglo ahora apunta a un objeto de tipo 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?

Porque Estrella también es una clase abstracta por lo que no debe necesariamente definir este método en su cuerpo.

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

Porque se le está dando una visibilidad menor que la que tiene en su clase padre, en la clase padre es de visibilidad package por lo que en una subclase no puede ser menos visible y privado es menos visible que package.

h. ¿Por qué la clase Nova no define el método tipoCuerpo1()? ¿Se podría definir? Si lo define, ¿qué interpreta de esta situación?

Porque su clase padre Estrella ya definió este método, por lo que no necesariamente se debe definir, en caso de que si se estaría realizando una sobreescritura de método.

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?

Porque viene heredado 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?

Porque se está creando un puntero mas no una instancia, las clases abstractas lo que impiden en la creación de instancia de ese tipo, pero no la creación de punteros de dicho tipo.

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

Explique.

```
A. Nova nova = new Nova();
B. ObjetoAstronomicoExtraSolar ob = new ObjetoAstronomicoExtraSolar();
C. ObjetoAstronomicoExtraSolar oa = nova;
```

La línea B es invalida ya que no se pueden crear objetos de una clase abstracta.

La línea C es valida porque solo se está creando un apuntador a un objeto ya creado en la línea A.

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.

```
A. Nova nova = new Nova();
B. ObjetoAstronomicoExtraSolar oa = nova;
C. oa.explotar();
D. ((Nova) oa).explotar();
```

La línea B es correcta porque solo se está creando un apuntador a un objeto ya creado en la línea A.

La línea C es incorrecta porque el tipo de apuntador no tiene definido el método explotar.

Si se omite la línea C imprime "Boom!".

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?

```
obN = null;
System.out.println(obN instanceof Object);
System.out.println("" + obN instanceof Object);
```

La línea 15 imprime true ya que obN hereda de Object, y siempre imprimirá lo mismo porque siempre se hereda de Object, las otras líneas imprimen false.

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

```
public ObjetoAstronomicoExtraSolar() {
    this.ID = 4;
    this.tipoCuerpo2();
}
```

No se genera error, el sentido de esto es que este constructor pueda ser usado por las subclases.

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?

```
class EnanaBlanca extends Estrella {
    void agotarCombustible() {
        System.out.println("Enana blanca muere");
    }
}
```

Esta clase al ser hija de una clase abstracta tendrá el error de que no está definiendo los métodos que su padre le indica, se puede corregir o poniendo esta clase como abstracta o definiendo los métodos faltantes los cuales son: descripción() y explotar()