Taller 7 parte 2

Programacion Orientada a Objetos

Santiago Abelardo Salcedo Rodriguez

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:

```
public abstract class Instrumento
{
    private String tipo;
    public Instrumento(String tipo) { this.tipo = tipo; }
    public String getTipo()     { return tipo;}

    public abstract void Tocar();
    public abstract void Afinar();
}

public class Saxofon extends Instrumento
{
    public Saxofon(String tipo) { super(tipo); }
    public void Tocar() { System.out.println("Tocando Saxofon");}
    public void Afinar() {System.out.println("Afinando Saxofon");}
}

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

R= La idea tras definir a instrumento como una clase abstracta es agrupar características y elementos que los instrumentos hijos deberían implementar, es decir todos los instrumentos van a tener sus métodos afinar y tocar propios pero con la clase abstracta instrumento nos aseguramos que las clases hijas lo tengan que sobreescribir, reduciendo la repetición decodigo

b) Elabore una clase denominada Piano heredada de Instrumento, añada un constructor y redefina para ella los métodos Tocar y Afinar.

R=

```
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?

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

R= Esto si se ejecuta aunque el apuntador sea de una clase abstracta esto no tiene ningún problema, ya que es un apuntador no un objeto, las siguientes 2 lineas no tienen problema al ser objetos de clases hijas

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();
    }
}
```

R=

Tocando Saxofón

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.

R=

Si se define de esa manera hay un claro error al intentar definirlo esto no es posible al ser un método abstracto.

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.

R=

No, no hay ningún problema con esta situacion, una clase abstracta puede tener métodos de instancia sin ningún problema y ser ejecutados sin problemas, obviamente no podrían ser ejecutados por una instancia de ellos, pero si sus clases hijas

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;
    }
}
```

R=

Si, una clase abstracta debe tener minimo un método abstracto sino ocurrirá un error de compilación

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.

R=

El arreglo oa es de tipo ObjetoAstronomicoExtraSolar[], que es una clase abstracta. Sin embargo, el arreglo no almacena objetos de tipo ObjetoAstronomicoExtraSolar sino objetos de clases hijas, y se ejecuta cada método propio a esa clse, esto es un claro ejemplo del polimorfismo

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?

R= En Realidad como ambos son hijos de la clase abstracta ObjetoAstronomicoExtraSolar no habría ningún problema cuando se reasigne ahora la referencia oa[0] apuntara al 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?

R= Ya que la clase Estrella es una clase abstracta y no es necesario que implemente todos los métodos abstractos de su superclase, lo importante es que sus subclases no abstractas las implementen

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

¿Por qué se genera error?

R= Habria error de compilación debido aque el método no será accesible fuera de la clase galaxia, y este método es invocado desde la clase ObjTaller7

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

R= Nova ya hereda el método de la clase Estrella, pero si se podría definir si quisiéramos extender su comportamiento, si se sobreescribe el método esto implicaría que el método se comporta de manera diferente a 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?

R= Ya que la clase lo heredan implícitamente de la clase object y ahí hay una implementación.

j. ¿Por qué en la línea 11 se puede crear un puntero obN de tipo ObjetoAstronomicoExtraSolar si esta es una clase abstracta?

R= Se pueden usar punteros de clases abstractas, lo que no podemos es crear objetos de la clase abstracta, como este apuntador (apunta) a un objeto de tipo nova que extiende de estrella y esta extiende de ObjetoAstronomicoExtraSolar, nos permite hacer la asignación de la subclase a un puntero de la superclase

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;
```

R= La a es valida, la B no es valida ya que se esta intentando instanciar una clase abstracta, la C es valida ya que estamos haciendo upcasting de nova a 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.

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

R=La linea B es correcta ya que se esta haciendo un upcasting de Nova a ObjetoAstronomicoExtraSolar aunque esta sea abstracta, la linea C es incorrecta, ocurriria un error en tiempo de ejecucion ya que el puntero oa es de tipo ObjetoAstronomicoExtraSolar y el metodo explotar() no esta definido en dicha clase sino que esta definido en las subclases, al omitirla Se imprimiria 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);
```

R= ya que se esta verificando si el objeto es instancia de object y esto sucede siempre y cuando si sea la instancia de algo ya que todos los objetos en Java heredan de object, no siempre devolverá True, cuando obN es null entonces imprimirá false ya que null no es instancia de ninguna clase

Imprimira false y true respectivamente ya que “”+ null es un objeto de tipo String y esta hereda de Object

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();
}
```

R= No no habría un error, se pueden colocar constructores en clases abstractas a pesar de que no puedas instanciar directamente una clase abstracta, su constructor puede ser invocado por las subclases cuando se crean objetos de esas subclases, el sentido de esto es proporcionar una inicialización común para todas las subclses

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");
   }
}
```

R= Generara errores ya que la clase enanablanca no implementa el método abstracto explotar(), se podrían corregir implementando de su propia manera dicho método también tipoCuerpo1() esta definido en Estrella debería ser sobrecargado en la clase EnanaBlanca (aunque no es un error necesariamente)