



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:

```
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");}
}
```

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



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();  
    }  
}
```

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.

```
abstract class Estrella extends ObjetoAstronomicoExtraSolar {  
    abstract void explotar() {  
        System.out.println("Estrella explotar");  
    }  
    int a = super.getID();  
    public void tipoCuerpol() {  
        System.out.println("Simple " + a);  
    }  
}
```

- 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.
- 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;  
    }  
}
```



- 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.
- 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*?
- 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?
- g. ¿Qué sucede si el método *tipoCuerpo1()* de la clase *Galaxia* se define como privado? ¿Por qué se genera error?
- h. ¿Por qué la clase *Nova* no define el método *tipoCuerpo1()*? ¿Se podría definir? Si lo define, ¿qué interpreta de esta situación?
- 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?
- j. ¿Por qué en la línea 11 se puede crear un puntero *obN* de tipo *ObjetoAstronomicoExtraSolar* si esta es una clase abstracta?
- 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;
```

- 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();
```

- 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);
```



- 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();  
}
```

- 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");  
    }  
}
```