Taller 7 Java Python Ejercicio 2

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");}
}
```

La clase Instrumento debe ser de tipo abstracta ya que posee métodos definidos de tipo abstracto, también debe ser una clase abstracta ya que es funcional no crear objetos de tipo Instrumento, o sea que esta clase sea una clase abstracta de la cual otras puedan heredar sus métodos y crear objetos de estas clases hijas.
Esta clase Instrumento se diferencia de una clase normal por que de esta pueden derivar otras clases de las cuales se pueden crear instrumentos específicos.

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?

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

No hay errores ya que el tipo de referencia puede ser de una clase abstracta pero la instanciación no puede declararse con una clase abstracta.

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.

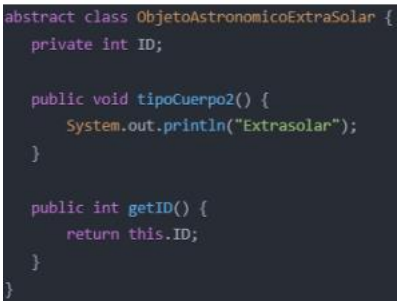
```
abstract class Estrella extends ObjetoAstronomicoExtraSolar {
   abstract void explotar() {
       System.out.println("Estrella explotar");
   }
   int a = super.getID();
   public void tipoCuerpol() {
       System.out.println("Simple " + a);
   }
}
```

El código no compilaría ya que los métodos abstractos solo deben declararse más no debe definirse nada en estos.

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 hay error, que estos métodos no sean abstract es por que realizan procesos en su cuerpo y además las clases hijas de la clase ObjetoAstronomicoExtraSolar pueden utilizar directamente estos métodos sin sobrescribirlos.

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 ya que una clase que hereda de una abstracta y no sobrescribe todos los métodos abstractos de la clase padre, esta clase debe ser abstracta. Además si se desea que una clase no pueda instanciar se debe simplemente colocar como 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.

En la línea 19 hace referencia a una clase abstracta ya que todos los elementos del arreglo serán de tipo ObjetoAstronomicoExtraSolar, pero ya en la línea 25 cada elemento del arreglo tiene un tipo más específico (el primer elemento es de tipo Galaxia, el segundo de tipo Nova y el tercero es de tipo SuperNova) por lo que en la línea 25 se toma el método que se sobrescribió en cada clase en específico de cada elemento.

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 asignó a esa posición donde antes era un elemento de tipo Galaxia, el valor del elemento en la ultima posición del arreglo el cual es de tipo SuperNova. Y en el método descripción de la clase SuperNova se imprime el mensaje: "Soy una Super Nova".

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 es abstracto?

Porque simplemente no se colocó y la clase Estrella se definió como una clase abstracta que no necesitaba del método descripción.

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

El código no compilaría porque se esta colocando un nivel de accesibilidad menor del que se puede, el método tipoCuerpo1 en la clase padre es public por lo que en la hija solo puede ser public también.

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

Porque no es necesario hacerlo ya que en la clase Estrella que es clase padre de la clase Nova ya se sobrescribió el método abstracto de la clase abstacta ObjetoAstronomicoExtraSolar. Si se puede definir este método, se puede sobrescribir normalmente y esto sería una simple sobrescritura 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?

Imprime lo predeterminado cuando se imprime un objeto, porque esto esta definido en 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?

Por que no importa si el apuntador es de tipo abstracto, además solo se está creando un puntero más no se está creando un objeto.

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 instrucción B es incorrecta ya que no se puede crear una instancia de una clase abstracta. La instrucción C es válida ya que no se está creando una instancia solo se esta creando un puntero a un objeto creado.

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 es un caso de generalización, solo se está creando un puntero a un objeto creado. La línea C es incorrecta ya que en el tipo de referencia, en esta clase no está definido el método explotar().
Omitiendo la instrucción C se imprimiría: “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);
```

Imprime true ya que al ser un objeto siempre va heredar a la clase Object, para cualquier objeto imprimiría true. Las siguientes líneas imprimen: false 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, se puede colocar normalmente un constructor a una clase abstracta y esto tiene sentido ya que las clases hijas de esta clase abstracta pueden usar este constructor para asignar valores a atributos y utilizar métodos que esten dentro de este constructor.

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");
    }
}
```

Se puede concluir que es una clase nueva que hereda de Estrella, pero esta clase tiene un error ya que no ha sobrescrito ningún método de las clases abstractas de

las cuales hereda, para corregir esto podría poner la clase de tipo abstract o sobrescribir los métodos descripcion() y explotar().