Respuestas Punto 2 -Taller 7 Java-Python- Julián Bedoya Palacio.

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");}
}
```

Debe definirse como abstracta debido a los métodos abstractos que posee y que deben ser implementados en las subclases. Es importante notar que los métodos abstractos no tienen cuerpo y terminan en punto y coma.

Lo que la diferencia de una clase normal es la palabra reservada abstract que se le coloca en la definición de la clase y aunque puede contar con constructores, no se puede instanciar un objeto de esa clase.

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");
    }
}
```

Todas las líneas se pueden ejecutar, debido a que x inicialmente no se está instanciando con la clase instrumento, y al instanciar con Saxofon y Guitarra, es válido por generalización, debido a que estas son subclases de instrumento.

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.

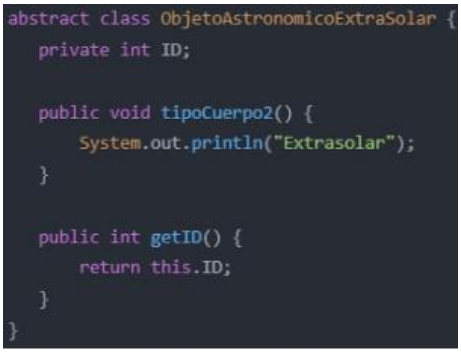
```
abstract class Estrella extends ObjetoAstronomicoExtraSolar {
   abstract void explotar() {
       System.out.println("Estrella explotar");
   }
   int a = super.getID();
   public void tipoCuerpol() {
       System.out.println("Simple " + a);
   }
}
```

Se generaría un error debido a que el método explotar al ser abstracto no debe tener cuerpo y debería terminar con un punto y coma.

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 podría considerarse un error, porque una clase abstracta puede contar con métodos abstractos y métodos normales, solo que en este caso dichos métodos no deben ser implementados en las subclases de ObjetoAstronomicoExtraSolar.

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 habría error en este caso, porque el implementar métodos abstractos es sola una posibilidad en el caso de requerir que las subclases no abstractas implementen obligatoriamente dichos métodos, por ende puede solo estar definida con métodos normales.

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 inicialmente se crea un arreglo con 3 entradas de tipo ObjetoAstronomicoExtraSolar, que no están instanciadas, posteriormente la inicialización se debe hacer con las subclases que heredan de la clase abstracta, para evitar errores porque una clase abstracta no puede instanciar; la invocación del método a cada clase derivada se hace debido a que este se sobrescribe (implementa) en cada una de ellas y realiza acciones diferentes.

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?

Debido a que en la línea anterior (28) la entrada 0 del arreglo que apuntaba a un objeto de tipo Galaxia se modificó para que quedara apuntando a lo que apuntaba la entrada 2 que en este caso es un objeto de tipo SuperNova y el método de instancia descripcion() de esta clase imprime "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 en abstracto?

Porque la subclase Estrella es una clase abstracta y este tipo de clases no deben implementar obligatoriamente los métodos abstractos de su padre.

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

Se genera error debido a que no se puede reducir la visibilidad del método abstracto de la clase padre (ObjetoAstronomicoExtraSolar) que es implementado en la subclase Galaxia.

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

Porque en su clase Padre que es abstracta correspondiente a Estralla dicho método no se define como abstracto, por lo que no es necesario que sea definido en la subclase; podría ser definida en la clase Nova y esto se interpretaría como sobre escritura de método y se resovería por ligadura dinámica

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:

Galaxia@3af49f1c

Se puede llamar debido a que el padre ObjetoAstronómicoExtraSolar hereda implícitamente de la clase Object y en particular hereda el método toString() que puede ser utilizado por sus subclases.

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

Porque no se está instanciando un objeto explícitamente de esa clase sino que se está poniendo a apuntar a un objeto nova que fue casteado para que fuera de tipo ObjetoAstronómicoExtraSolar.

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 en la línea B no es válida debido a que se está instanciando un objeto de una clase que es abstracta, lo cual no es posible. Mientras que lo que sucede en la línea C si es válido debido a que un apuntador de tipo ObjetoAstronomicoExtraSolar está apuntando a un objeto de tipo Nova que por generalización es válido, ya que en jerarquía Nova es menor.

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();
```

Lo que sucede en la línea B es válida debido a que un apuntador de tipo ObjetoAstronomicoExtraSolar está apuntando a un objeto de tipo Nova que por generalización es válido, ya que en jerarquía Nova es menor.

La instrucción en la línea C no es válida debido a que el método explotar() tiene que estar definido en la clase ObjetoAstronomicoExtraSolar o heredarlo de su padre para que pueda ser resuelto por ligadura dinámica.

Finalmente omitiendo la línea C, se imprime:

Boom!

Debido a que se hizo la respectiva especialización quedando de tipo Nova para que pueda usar 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?

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

La línea 15 imprime true, debido a que obN es un apuntador de ObjetoAstronomicoExtraSolar que es subclase de Object.

No siempre imprime lo mismo, en caso de que el objeto no sea una instancia de la clase en específico o una de sus subclases imprimirá false.

Ambas imprimen false, debido a que obn al ser null, no es instancia de ningún tipo.

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 y si es posible colocar constructores en clases abstractas en sentido que desde las subclases que heredan de esta, desde su constructor se invoque con super() el constructor del padre y se les de valor a los atributos comunes definidos este último.

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 generan errores de compilación, debido a que no se está implementando el método abstracto explotar() de la clase Estrella ni el método abstracto descripción() de la clase abstracta ObjetoAstronómicoExtraSolar que se debe agregar debido a que Estrella hereda de esta.

Para corregirlos bastaría con agregarlos dentro de la clase EnanaBlanca; puede ser así:

```
class EnanaBlanca extends Estrella {

	void agotarCombustible() {
		System.out.println("Enana blanca muere");
	}

    void explotar() {
        System.out.println("Boom!");
    }

    void descripcion() {
        System.out.println("Soy una Nova");
    }
}
```