# PROGRAMACIÒN ORIENTADA A OBJETOS

| TALLER No. 7 | PROGRAMACIÓN ORIENTADA A OBJETOS |
|---|---|

**Profesor**: Jaime Alberto Guzmán Luna

Contenido del taller:

1. Clases abstractas
2. Métodos abstractos
3. Operador instanceOf
4. Función isinstance

## EJERCICIO 1

Dada la siguiente declaración de clase abstracta:

```
abstract class Prueba {
     abstract void m1(Object o);
}
```

¿Cuál de las siguientes declaraciones son válidas? **Argumente el motivo de su elección.**

**A.**
```
class A extends Prueba {
     void m1 (String o) { }
}
```

**B.**
```
class B extends Prueba {
     void m1 (Object o);
}
```

**C.**
```
abstract class C extends Prueba {
     void m1 (String o);
}
```

**D.**
```
abstract class D extends Prueba {
     void m1 (String o) { }
}
```

# PROGRAMACIÒN ORIENTADA A OBJETOS

## EJERCICIO 2

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 Instrumento();
            x = new Saxofon(“xxxxx”);
            x = new Guitarra();
            x = new Piano(“xxxxxxx”);
    }
}
```

d) Que 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();
            x = new Piano(“xxxxxxx”);  x.Tocar();
    }
}
```

**EJERCICIO 3**

¿Cuáles de las siguientes declaraciones representan una clase abstracta? **Argumente el motivo de su elección**.

A

```
from abc import ABC, abstractmethod
class C(ABC):
    @staticmethod
    @abstractmethod
    def my_abstract():
      print("Holy Week -->  ")
```

B

```
from abc import ABC, abstractmethod
class C(ABC):
  def my_abstract(self):
    print("programming")
```

C

```
from abc import ABC, abstractmethod
class C(ABC):
    @abstractmethod
    def my_abstract(self):
        print("Soy - no soy abstracto")
```

D

```
from abc import ABC, abstractmethod
class C(ABC):
    @classmethod
    @abstractmethod
    def my_abstract(cls):
      pass
```

¿Qué sucederá si se crea una instancia por individual de cada una de las definiciones de las clases anteriores?

**EJERCICIO 4**

¿Qué se debe quitar al código para que al ejecutarlo se obtenga la siguiente salida?

```
 vive en Granada
vive en Medellín
True
 vive en barco
 vive en Tenza
vive en Medellín
True
```

**Enlace al código:** [https://n9.cl/w1r4y](https://n9.cl/w1r4y)

## EJERCICIO 5

```
1  public class ObjTaller7 {
2      public static void main (String[] args) {
3          Galaxia g = new Galaxia();
4          Nova nova = new Nova();
5          nova.tipoCuerpo1();
6          nova.explotar();
7          ((Estrella) nova).explotar();
8          g.tipoCuerpo1();
9          System.out.println(g.toString());
10
11         ObjetoAstronomicoExtraSolar obN = (ObjetoAstronomicoExtraSolar)
   nova;
12         System.out.println(obN instanceof ObjetoAstronomicoExtraSolar);
13         System.out.println(obN instanceof Nova);
14         System.out.println(obN instanceof Estrella);
15         System.out.println(obN instanceof Object);
16         System.out.println(obN instanceof SuperNova);
17         System.out.println(obN instanceof Galaxia);
18
19         ObjetoAstronomicoExtraSolar[] oa = new
   ObjetoAstronomicoExtraSolar[3];
20         oa[0] = new Galaxia();
21         oa[1] = new Nova();
22         oa[2] = new SuperNova();
23
24         for (int i = 0; i < oa.length; i++) {
25             oa[i].descripcion();
26         }
27
28         oa[0] = oa[2];
29         oa[0].descripcion();
30     }
31 }
32
33     abstract class ObjetoAstronomicoExtraSolar {
34     private int ID;
35
36     abstract void tipoCuerpo1();
37     abstract void descripcion();
```

```
38
39      public void tipoCuerpo2() {
40          System.out.println("Extrasolar");
41      }
42
43      public int getID() {
44          return this.ID;
45      }
46  }
47
48  class Galaxia extends ObjetoAstronomicoExtraSolar {
49
50      public void tipoCuerpo1 () {
51          System.out.println("Compuesto");
52      }
53
54      public void descripcion() {
55          System.out.println("Soy una Galaxia");
56      }
57  }
58
59  abstract class Estrella extends ObjetoAstronomicoExtraSolar {
60      int a = super.getID();
61      abstract void explotar();
62
63      public void tipoCuerpo1() {
64          System.out.println("Simple " + a);
65      }
66  }
67
68  class Nova extends Estrella {
69
70      void explotar() {
71          System.out.println("Boom!");
72      }
73
74      void descripcion() {
75          System.out.println("Soy una Nova");
76      }
77  }
78
79  class SuperNova extends Estrella {
80
81      void descripcion() {
82          System.out.println("Soy una Super Nova");
83      }
84
85      void explotar () {
86          System.out.println("Cataplum!");
87      }
88  }
```

# PROGRAMACIÒN ORIENTADA A OBJETOS

**Responda:**

1. ¿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 tipoCuerpo1() {
        System.out.println("Simple " + a);
    }
}
```

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

3. 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;
   }
}
```

4. 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.

5. ¿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*?

6. ¿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?

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

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

9. ¿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?

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

11. ¿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;
```

12. 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();
```

13. ¿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);
```

14. Agregue el siguiente constructor, ¿Se genera error? ¿Se puede colocar constructores a clases abstractas? ¿Qué sentido tiene?

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

15. 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");
 }
}
```

16. ¿Por qué las líneas 16 y 17 imprimen *false* y *false* respectivamente?

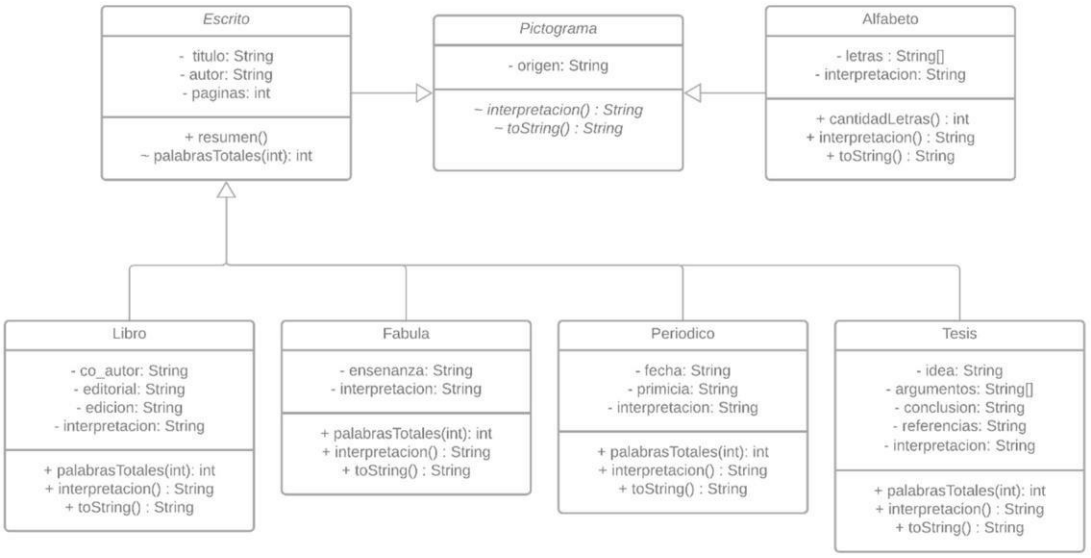
# PROGRAMACIÒN ORIENTADA A OBJETOS

**EJERCICIO 4 - GitHub**

**Link entrega:** https://classroom.github.com/a/tcohKpeG

## PROBLEMA Los inicios de la escritura

La escritura como la conocemos hoy evoluciono y se transformó desde algo conocido como los Pictogramas (pequeños dibujos que tenían interpretación), y un proceso evolutivo de ciertos pasos intermedios para llegar a lo que conocemos hoy en día. Suponga que se le pidió crear esta línea jerárquica y ciertos procedimientos usando POO.




El ejercicio consiste en representar el diagrama de clases anterior y ciertos procedimientos especiales aquí descritos, también tenga en cuenta que las clases Escrito y Pictograma ya se les entrega, no las modifique.

Siga las siguientes indicaciones:

- Cree los métodos get y set de los atributos de las clases
- Cree los constructores con los atributos que tiene la clase en el orden como se enuncia en el diagrama de clase.
- Para los atributos de listas, haga uso de listas estáticas
- El método interpretacion() de las clases, retornará el valor del atributo interpretacion.
- El método toString() de la clase Alfabeto, retornara cada letra del alfabeto separada por “, ”no olvide el espacio entre la coma y la siguiente letra
- El método toString() de las demás clases, retornara el valor de los atributos (inclusive los de su padre) sin incluir el atributo interpretación de las clases, este retornara línea por línea cada valor por ejemplo si se crea un objeto de la clase Fabula de la siguiente manera:
  new Fabula("pensamiento", "La tortuga y la liebre", "Esopo", 2, "no se debe uno burlar de los demás, ni presumir o ser vanidoso", "Cuento corto");
  Al llamar el método interpretación retornara lo siguiente:
  "pensamiento
  La tortuga y la liebre
  Esopo
  2
  no se debe uno burlar de los demás, ni presumir o ser vanidoso”
  Para la clase Tesis, en el atributo argumentos retorne el valor de la cantidad de

argumentos que tiene el objeto de la Clase Tesis.

- Para la clase Alfabeto el método cantidadLetras() retornara el numero de elementos que tiene el atributo letras.
- Para el método palabrasTotales() retornara la multiplicación de las paginas por el parámetro por un factor. Este factor depende de la clase, para la clase Tesis el factor es 5, para la clase Fabula el factor es 1, para la clase Libro el factor es 2, y para la clase Periódico el factor es 10.

**NOTA:** Tenga en cuenta que, en un diagrama de clases, una clase abstracta se representa con elnombre de la clase en *cursiva.*