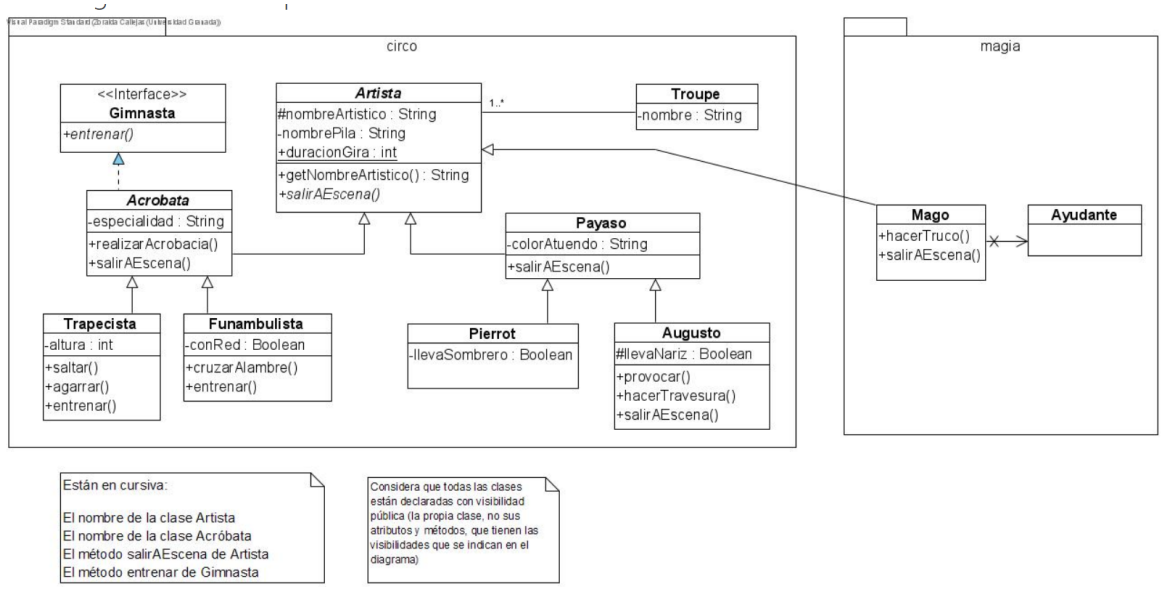
CLASE 11: REPASO DE HERENCIA Y POLIMORFISMO

EJERCICIO 1




```ruby
class Trapecista < Acrobata
    def initialize(altura, especi, nombreArt,nombreP,divGira)
        # Contructor de clase
        super(especi, nombreArt, nombreP, divGira)
        # Inicializar atributos propios
        @altura = altura

    end
end
```

EJERCICIO 2

JAVA

```java
public class Payaso{
    public void salirAEscena(){
        System.out.printl("¿Como están ustedes?");
    }
}

public class Augusto extends Payaso{
    @Override
    public void salirAEscena(){
        super.salirAEscena();
        hacerTravesura();
    }
}
```

RUBY

```ruby
class Augusto < Payaso
    def salirAEscena
        super
        hacerTravesura
    end
end
```

EJERCICIO 3

```ruby
def presentacionShow(compi) # compi es de la clase Augustu
    #nombreArtistico es protected en Artista y llevaNariz es un atributo
    puts "Mi compi es #{compi.nombreArtistico}, lleva nariz? #
{compi.llevaNariz}"
end
```

Como visto yo colorAtuendo, no puedo

Llevo sombrero? @llevaSombrero si

Lleva sombrero el compi? compi.llevaSombrero. En ruby no, en java si

EJERCICIO 4

Encuentra errores

```java
Artista x1 = new Acrobata(); // es un error porque Acrobata es abstracta
Gimnasta x2 = new Funambulista(); // si se puede declarar una interfaz y una
clase abstracta pero no puedo instanciarlas, por lo tanto es correcto
Artista p3 = new Funambulista();
// Si funciona, primero el compilador busca si esta en Artista, donde si está. El
interprete en tiempo de ejecucion mira el dinamico (funambulista), el cual el
método esta heredado de acrobata y es el que ejecuta.
p3.salirAEscena();
Payaso x7 = new Augusto(); // funciona
// El compilador busca provocar() en Payaso y como no está ni en la super clase,
pues da un error de compilador.
x7.provocar();
// Para solucionarlo:
((Augusto)x7).provocar();

Payaso pa2 = new Pierrot(); // bien
((Augusto)pa2).salirAEscena(); // bien porque coge el metodo heredado de payaso
y no hace falta el casting
```

EJERCICIO 4

Creamos una subclase de Fundambulista llamada, FunambilistaCiclista. Redefino los métodos de salirAEscena() y cruzarAlambre().

EJERCICIO 5

La clase acróbata no debería ser abstracta porque no tiene ningún método abstracto. FALSO, porque tiene el método abstracto entrenar de Gimnasta() ya que como Acrobata implementa la interfaz, se compromete a crear los métodos de Gimnasta y sus hijas tambien heredan y como no lo implementan, es abstracto.

EJERCICIO PARCIAL

EJERCICIO 2

En Java la implementacion del método disparar de Forajido de forma que dispare a matar sería:

a. FUNCIONA

```
void disparar(Persona blanco){
    dispararAMatar(blanco); // dispararAMatar lo hereda de la clase y puede usar
sin poner super
}
```

b. FUNCIONA

```
void disparar(Persona blanco){
    super.dispararAMatar(blanco); // dispararAMatar lo hereda de la clase y
puede usar sin poner super
}
```

Por lo tanto ambas opciones funcionarían.