EJERCICIO 1

Queremos xestionar unha academia de baile. Para elo, temos que gardar información tanto dos alumnos como dos profesores que imparten clases na academia. De ambos queremos saber o seu nome, apelidos, móbil. Dos profes además queremos almacenar o NIF para o cal chamarán ao método construtor de Persoa además de almacenar o NIF.

Temos que declarar as seguintes clases:

- A clase *Persoa* debe ter un método verInformación que devolve para a información co seguinte formato: `Uxia Loureiro Agra (699444999)`
- A clase *Alumno* ten dous métodos: `setNumClases` e `aPagar`, e debe empregar o método construtor de Persoa.
- O método aPagar devolverá o importe e pagan en función do número de actividades nas que se inscriben:

  Por unha actividade: 20 euros
  Por dúas actividades:32 euros
  Por tres ou máis: 40 euros.
  No caso de que non estea establecido o número de clases ás que asiste para ese alumno devolverá 'Debe indicar previamente o número de clases'.

- A clase *Profesor* ten un método `calcularSoldo` que calcula o que cobran os profesores dependendo do número de clases que imparten ao mes. Recibe como parámetros o número de horas e o importe de cada hora, que está establecido en 16 euros pero podería variar.
- A clase *Baile* con dous atributos: `nome` e `idadeMínima`. A idade mínima será de 8 anos salvo que se indique o contrario.

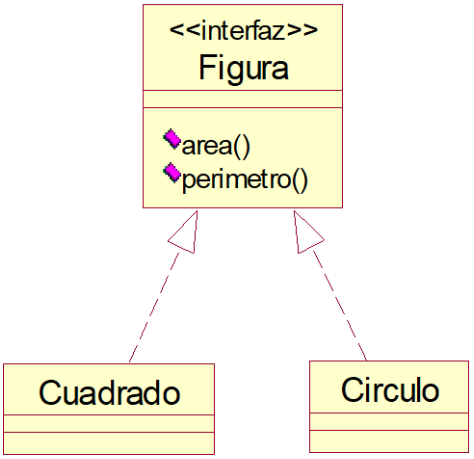
  O profesor terá 3 métodos para engadir os Bailes que imparte, eliminar un baile cando deixe de impartilo e para devolver os bailes que imparte da forma:
  `HIP HOP (idade min:8 anos)`
  Antes de engadir un baile debe comprobar se xa está dado de alta para ese profesor.

- A clase Academia: almacenará o seu nome nunha constante e debe permitir engadir Profesores e Alumnos.

Para probalo debes facer o seguinte:

1. Engade á academia un profesor que imparte 4 bailes (entre eles AFRO, e un de les duplicado) e 2 alumnos.
2. Mostra información do profesores (incluíndo o soldo e os bailes que imparte) e dos alumnos incluindo a cuota que deberá pagar.
3. O profesor deixa de dar clase de AFRO. Actualiza a información da academia e volve a mostrar a información do profesor.
4. Impide a herdanza das clases Alumno e Profesor.

EJERCICIO 2

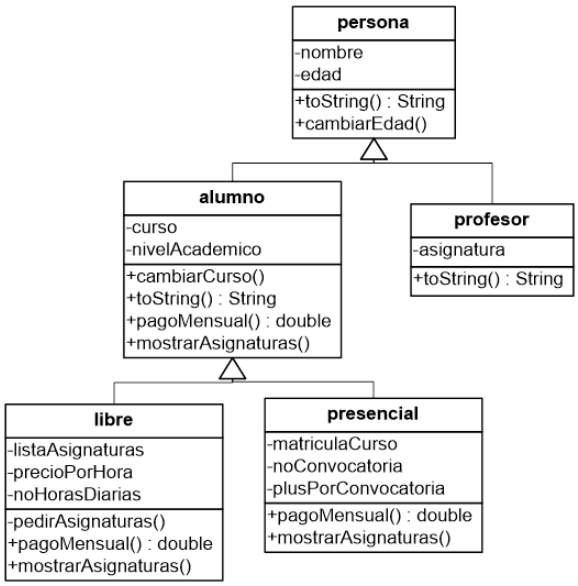


Tenemos la siguiente interfaz:

```
interface Figura {
        public function area();
        public function perimetro();
}
```

- Implementa en un proyecto PruebasFiguras la interfaz y las clases Cuadrado y Circulo para comprobar cómo se definen e implementan las interfaces. Las funciones deben calcular y retornar el valor del área y del perímetro respectivamente.
- El valor de PI existe en PHP una constante `M_PI` o puedes usar la función pi().
- Escribe el constructor de las clases Circulo y Cuadrado.
- Crea varios objetos Circulo y Cuadrado y calcula su área y perímetro.
- Crea una función que permita imprimir el área, muestraArea, sea de un Círculo o un Cuadrado haciendo uso del polimorfismo y también otra muestraPerimetro.
- Crea varios objetos Circulo y Cuadrado y calcula su área y perímetro. Utiliza las funciones muestraArea y muestraPerimetro.

EJERCICIO 3



- Crea un proyecto Curso e implementa el código del ejemplo anterior para comprobar el funcionamiento de las clases abstractas. Es necesario que crees la clase Persona y las subclases.
- El pago mensual en los alumnos de libre es: precioPorHora*noHorasDiarias*30.
- El precio hora será el mismo para todos los alumnos, será un atributo estático con un importe inicial de 10 y tendrá métodos get y set para configurarlo.
- El pago mensual para los alumnos de Presencial es: matriculaCurso+plusPorConvocatoria*noConvocatoria)/12.
- Un alumno de presencial se matricula en todas las asignaturas del curso. No tiene opción de seleccionar asignaturas.
- Un alumno de libre se inscribe un número de horas diarias y podrá seleccionar tantas asignaturas como de horas se matricule (con una máximo de 5 horas diarias).
- Crea varios alumnos presenciales y varios alumnos de libre. El método pedirAsignaturas lo vamos a nombrar como setListaAsignaturas y recibirá un array con la lista de asignaturas.
- En este ejemplo sólo implementamos la funcionalidad de las clases POO no la parte de interfaz gráfica, para realizar las pruebas…
- Comprobación de funcionamiento, el programa debe de permitir ejecutar correctamente el siguiente código de ejemplo:

```
//Comprobamos el funcionamiento de las clases creando los siguientes
    $aluLibre1 = new Libre ("Eva", 34, 1, "Básico", 2);
    $aluLibre2 = new Libre ("Pedro", 34, 2, "Intermedio", 3);
```

```
echo $aluLibre1; echo "<br/>";
echo $aluLibre2; echo "<br/>";

$aluLibre1->setAsignaturas(array("Programacion", "Lenguajes de Marcas"));
echo $aluLibre1->pagoMensual(); echo "<br/>";

$aluPresencial1 = new Presencial ("Julio", 20, 2, "Intermedio", 950.70, 10.25, 2);
$aluPresencial2 = new Presencial ("Rosa", 21, 2, "Intermedio", 950.70, 10.25, 1);

echo $aluPresencial1; echo "<br/>";
echo $aluPresencial2; echo "<br/>";

$prof1 = new Profesor("Juan", 40, "Programación");
echo $prof1; echo "<br/>";
```

- Por seguridad evita que se puedan crear atributos para los objetos de estas clases que no hayan sido declarados dentro de la clase.