# Examen de Programación
# Junio 2019
# (Grados en Física y Matemáticas)

**Primera parte (1.25 puntos por cuestión, 50% nota del examen)**

1) Hay palabras que tienen todas las vocales del abecedario, como por ejemplo *ayuntamiento*. Escribir una función que retorne un booleano indicando si una palabra que se le pasa como parámetro tiene o no todas las vocales. No distinguir mayúsculas de minúsculas (por ejemplo, el resultado para una letra `'A'` o una `'a'` debe ser el mismo). Por simplicidad no hace falta ocuparse de las letras acentuadas. Se valorará la eficiencia y tamaño compacto del código.

2) La siguiente tabla muestra los precios de las entradas al torneo Roland Garros de tenis, en su fase final:

| Nivel | Categoría | Género | Precio (euros) |
|---|---|---|---|
| final | individual | masculina | 300 |
| final | individual | femenina | 175 |
| final | dobles | masculina | 120 |
| final | dobles | femenina | 110 |
| semifinal | individual | masculina | 150 |
| semifinal | individual | femenina | 130 |
| semifinal | dobles | masculina | 100 |
| semifinal | dobles | femenina | 90 |

Escribir una función a la que se le pasen como parámetros el nivel y la categoría de la competición así como el género, y que retorne el precio. Los parámetros son de tipo *string*. Si el nivel, categoría o género no son los indicados se retornará `math.nan`. Se valorará la eficiencia y tamaño compacto del código.

3) Escribir una función que calcule y retorne el valor del arco tangente hiperbólico de z dado por la siguiente expansión en serie:

$$arc\tanh(z) = z + \frac{z^3}{3} + \frac{z^5}{5} + \frac{z^7}{7} + \frac{z^9}{9} + \ldots$$

La función recibe como parámetros el valor de *z* y el número de términos del desarrollo en serie. Por eficiencia se debe evitar usar el operador ** o la función `pow()`. Para ello, el numerador de cada término del desarrollo en serie se debe obtener del numerador anterior multiplicándolo por `z` y por `z`.

4) En un computador con sistema operativo Linux se desea escribir un *script* para recoger ficheros de varios directorios llamados `data1`, `data2` y `data3` situados en `/usr/local/app`.

Los ficheros recogidos se meterán en el directorio `recopila` que deberá crearse en el directorio del usuario.

El *script* debe realizar los siguientes pasos. Utilizar rutas absolutas para acceder a los directorios `data1`, `data2` y `data3`, y rutas relativas para los contenidos del directorio del usuario.

- situarse en el directorio del usuario
- crear el directorio `recopila`
- mover a `recopila` los ficheros terminados en `.dat` que estén en `data1` y `data2`
- copiar en `recopila` los ficheros terminados en `.py` que estén en `data3`
- borrar de `data3` los ficheros terminados en `.txt`
- copiar en `recopila` los directorios llamados `backup1` y `backup2` (con todos sus contenidos) que están en `data1` y `data2`, respectivamente
- borrar de `data3` el directorio `backup3`

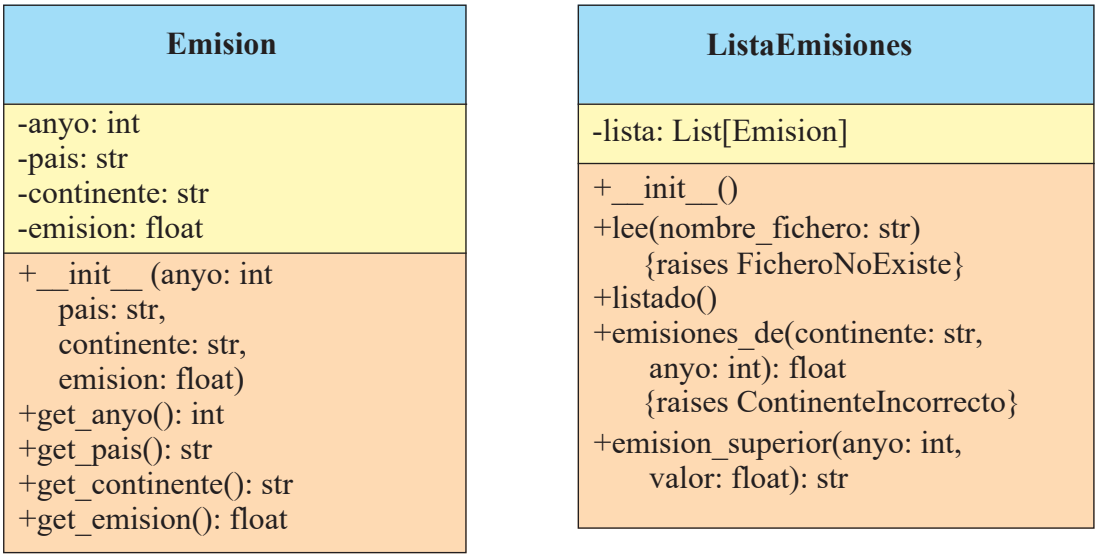
Fases a seguir:

- En primer lugar hacer un esquema de la distribución inicial de directorios.
- En segundo lugar escribir el *script*.
- En tercer lugar hacer un esquema de la distribución final de directorios y ficheros.

# Examen de Programación
# Junio 2019
# (Grados en Física y Matemáticas)

**Segunda parte (5 puntos, 50% nota del examen)**

Se dispone de un fichero con datos de las emisiones de $CO_2$ de diferentes países y se desea hacer parte del software que sirva para su análisis. Se dispone de la clase Emision ya realizada, que guarda un dato de emisión de un país en un año concreto y responde al diagrama de clases que se muestra.




Los atributos de la clase se indican a continuación. La clase dispone también de un constructor al que se pasan los valores de los atributos y de un método observador para cada atributo:

- anyo: año en el que se mide la emisión
- pais: nombre del país
- continente: nombre del continente
- emision: valor de emisión de $CO_2$ en toneladas métricas por habitante en el año indicado

Se pide hacer la clase ListaEmisiones que responde al diagrama de clases que se muestra arriba. La clase dispondrá como atributo de una lista de objetos de la clase Emision y diversos métodos para analizar las emisiones:

- *constructor*: crea la lista vacía
- lee(): lee los datos de emisiones del fichero cuyo nombre se indica en el parámetro y los mete en la lista. El fichero contiene dos líneas de encabezamiento y luego un dato de emisión por cada línea, con sus campos (año, país, continente y emisión) separados por comas, como en el ejemplo que se muestra a continuación

```
CO2 Emissions (metric tons per capita)
Year,Country,Continent,Emission
2008,Aruba,South America,24.75013321
2009,Aruba,South America,24.87670585
2010,Aruba,South America,24.18270225
2011,Aruba,South America,23.9224121
2008,Andorra,Europe,6.296124556
...
```

Con los datos de cada línea se creará un objeto de la clase Emision y se añadirá a la lista. Se puede suponer que los datos del fichero son correctos.

Se deberá tratar la excepción que ocurre si el fichero no existe. En este tratamiento se pondrá un mensaje de error en pantalla y seguidamente se lanzará la excepción FicheroNoExiste, ya definida en el mismo módulo.

- listado(): muestra en pantalla un listado de todos los datos de la lista, en columnas, con las emisiones redondeadas a tres decimales. El listado comenzará por un encabezamiento de una o dos líneas que explique los datos que vendrán a continuación. El encabezamiento incluirá el número de elementos de la lista.
- emisiones_de(): calcula y retorna la suma de emisiones del continente y año que se pasan como parámetros. Si el continente indicado como parámetro no es válido se lanzará la excepción ContinenteIncorrecto, ya definida en el mismo módulo. Los continentes válidos son "Africa", "Asia", "Europe", "North America", "Oceania" y "South America". Si no hay datos para el año y continente indicados se retornará cero.
- emision_superior(): busca en la lista el primer dato de emisión para el año indicado, que supere el valor indicado. Si lo encuentra, retorna el país de ese dato. Si no lo encuentra, retorna None.

Finalmente, se pide hacer un programa principal en el mismo módulo, que haga lo siguiente:

a. Crea un objeto de la clase ListaEmisiones

b. Invoca al método lee() para leer los datos del fichero "emissions.csv"

c. Hace un listado invocando al método listado()

d. Muestra en pantalla las emisiones del continente Europe en 2010

e. Muestra en pantalla el primer país que supera 24 toneladas métricas per cápita en 2011

Tratamiento de errores:

- Si en el paso b) se lanzase FicheroNoExiste se abandonan los restantes pasos y se escribe el mensaje "El programa no se puede ejecutar"
- Si en el paso d) se lanzase ContinenteIncorrecto se pone un mensaje de error y luego se continúa normalmente con el paso e)

*Valoración*:

- encabezamiento de la clase y constructor: 0.5 puntos
- método lee(): 1 punto
- método listado(): 0.75 puntos
- método emisiones_de(): 1 punto
- método emision_superior(): 0.75 puntos
- programa principal: 1 punto