UNIVERSITAT JAUME I

1.º Ingeniería Informática
1.º Matemática Computacional
EI1003/MT1003 Programación I
Práctica 5

En los ejercicios de este boletín vas a empezar a definir tus propias clases para utilizarlas en tus programas. Has de fijarte en respetar no solamente lo que cada ejercicio especifique como comportamiento de entrada/salida del correspondiente programa, sino también lo que indique sobre las clases que debas escribir y el comportamiento de cada uno de sus métodos. Además, desde tus programas debes limitarte a acceder a los objetos de tus clases a través de sus métodos, *nunca accedas directamente a sus atributos.*

Por otra parte, aunque lo normal sea reutilizar clases en diferentes programas mediante el mecanismo de la importación, *para resolver los ejercicios de este boletín, te pedimos que cada uno de ellos lo soluciones incluyendo en un único fichero todo tu código Python.* Así pues, si una misma clase definida por ti has de utilizarla en varios ejercicios, simplemente escríbela en el primero de ellos y pégala en los demás.

Ejercicio 1

Escribe una clase, `Cuenta`, para representar la situación de una cuenta bancaria. Como único atributo tendremos el `saldo`. Los métodos serán los siguientes:

- Constructor[1]: asigna a `saldo` un importe inicial recibido como parámetro.
- `ingresar`: incrementa el `saldo` en un importe recibido como parámetro.
- `reintegrar`: decrementa el `saldo` en un importe recibido como parámetro.
- `consultar_saldo`: devuelve el valor actual del `saldo`.

Supón (sin necesidad de comprobarlo) que los parámetros recibidos por el constructor, `ingresar` y `reintegrar` son siempre correctos. Asume también que todos los importes manejados vienen en la misma unidad: euros.

Y, como primera prueba de tu clase, añade y ejecuta el programa principal que aparece a continuación:

```
c1 = Cuenta(0)
c2 = Cuenta(1000)
c1.ingresar(100)
c2.reintegrar(20)
print("Saldos: {0} y {1}".format(c1.consultar_saldo(), c2.consultar_saldo()))
```

Debes obtener el siguiente resultado:

```
Saldos: 100 y 980
```

Ejercicio 2 (Obligatorio)

Utiliza la clase `Cuenta` definida en el ejercicio anterior para hacer un programa que nos permita llevar el control de una cuenta corriente. El programa creará una cuenta con un saldo inicial solicitado al usuario y después le irá mostrando repetidamente un menú con las siguientes opciones:

- *Hacer un ingreso.* Si el usuario la elige, el programa deberá solicitarle un importe y llevar a cabo el correspondiente ingreso.
- *Pedir un reintegro.* Análoga a la anterior, pero llevando a cabo un reintegro con el método `reintegrar`.
- *Consultar saldo.* El programa mostrará por pantalla el saldo actual de la cuenta.
- *Salir.* El programa terminará sin volver a mostrar el menú.

Debes respetar cuidadosamente el comportamiento de entrada/salida que ilustra este ejemplo de ejecución:

[1] Recuerda que, para definir un constructor, debes utilizar `__init__` como nombre del método.

```
Introduce cuántos euros quieres como saldo inicial de tu cuenta: 10

1. Hacer un ingreso
2. Pedir un reintegro
3. Consultar el saldo
4. Salir
Elige una opción: 1
Introduce cuántos euros quieres ingresar: 90.5

1. Hacer un ingreso
2. Pedir un reintegro
3. Consultar el saldo
4. Salir
Elige una opción: 2
Introduce cuántos euros quieres que se te reintegren: 40

1. Hacer un ingreso
2. Pedir un reintegro
3. Consultar el saldo
4. Salir
Elige una opción: 3
El saldo actual es 60.50 euros

1. Hacer un ingreso
2. Pedir un reintegro
3. Consultar el saldo
4. Salir
Elige una opción: 4
¡Adiós!
```

Y supón (sin necesidad de comprobarlo) que el usuario siempre proporciona datos correctos por teclado.

Ejercicio 3

Escribe una clase, `Coche`, para representar el movimiento de un coche. Como atributos tendremos el `tiempo` transcurrido (en horas), la `distancia` recorrida (en kilómetros) y la `velocidad` actual (en kilómetros por hora). Los métodos serán los siguientes:

- Constructor: inicializa a cero `tiempo`, `distancia` y `velocidad`.
- `acelerar`: modifica la `velocidad` según un incremento recibido como parámetro.
- `decelerar`: modifica la `velocidad` según un decremento recibido como parámetro.
- `actualizar`: recibe como parámetro una cantidad de tiempo y la utiliza para actualizar `tiempo` (sumándole esa cantidad) y `distancia` (suponiendo que el coche, durante esa cantidad de tiempo recibida como parámetro, ha mantenido constante su `velocidad`).
- `consultar_tiempo`: devuelve el valor actual de `tiempo`.
- `consultar_distancia`: devuelve el valor actual de `distancia`.
- `consultar_velocidad_actual`: devuelve el valor actual de `velocidad`.
- `consultar_velocidad_media`: devuelve la velocidad media del coche calculada a partir de la `distancia` recorrida y el `tiempo` transcurrido; si no puede calcularse (porque el `tiempo` es cero), el método debe devolver `None`.

Supón (sin necesidad de comprobarlo) que los parámetros recibidos por los métodos son siempre correctos. Además, considera correcto recibir valores negativos como parámetros de los métodos `acelerar` (que, en tal caso, acabará reduciendo la velocidad del coche) y `decelerar` (que, en tal caso, acabará incrementándola).

A continuación, prueba tu clase con el siguiente programa principal:

```
coche = Coche()
coche.acelerar(100)
coche.actualizar(1.5)
coche.decelerar(40.1)
coche.actualizar(0.5)
print("Tiempo: {0}".format(coche.consultar_tiempo()))
print("Distancia: {0}".format(coche.consultar_distancia()))
print("Velocidad actual: {0}".format(coche.consultar_velocidad_actual()))
print("Velocidad media: {0}".format(coche.consultar_velocidad_media()))
```

Debes obtener el siguiente resultado:

```
Tiempo: 2.0
Distancia: 179.95
Velocidad actual: 59.9
Velocidad media: 89.975
```

Ejercicio 4 (Obligatorio)

Utiliza la clase `Coche` definida en el ejercicio anterior para hacer un programa que nos permita simular el movimiento de un coche. El programa creará un objeto de la clase `Coche` y después irá mostrándole repetidamente al usuario un menú con las siguientes opciones:

- *Acelerar.* Si el usuario la elige, el programa deberá solicitarle en qué medida quiere incrementar la velocidad del coche y aplicarle la correspondiente aceleración.
- *Decelerar.* Análoga a la anterior, pero aplicando una deceleración con el método `decelerar`.
- *Actualizar.* El programa solicitará al usuario el tiempo transcurrido desde la última actualización (o desde el instante cero si es que no ha habido ninguna actualización previa) y hará uso del método `actualizar`.
- *Consultar.* El programa mostrará por pantalla el tiempo total trascurrido, la distancia total recorrida por el coche y sus velocidades actual y media (o un mensaje alusivo a la imposibilidad de calcular esta última).
- *Salir.* El programa terminará sin volver a mostrar el menú.

Debes respetar cuidadosamente el comportamiento de entrada/salida que ilustra este ejemplo de ejecución:

```
Tu coche está inicialmente parado

1. Acelerar
2. Decelerar
3. Actualizar
4. Consultar
5. Salir
Elige una opción: 1
Introduce cuántos km/h quieres acelerar: 120

1. Acelerar
2. Decelerar
3. Actualizar
4. Consultar
5. Salir
Elige una opción: 4
```

```
El tiempo trascurrido es 0.00 h
La distancia recorrida es 0.00 km
La velocidad actual es 120.00 km/h
No puedo calcular la velocidad media si no ha trascurrido tiempo

1. Acelerar
2. Decelerar
3. Actualizar
4. Consultar
5. Salir
Elige una opción: 3
Introduce cuántas horas han pasado: 0.5

1. Acelerar
2. Decelerar
3. Actualizar
4. Consultar
5. Salir
Elige una opción: 2
Introduce cuántos km/h quieres decelerar: 40

1. Acelerar
2. Decelerar
3. Actualizar
4. Consultar
5. Salir
Elige una opción: 3
Introduce cuántas horas han pasado: 1.5

1. Acelerar
2. Decelerar
3. Actualizar
4. Consultar
5. Salir
Elige una opción: 4
El tiempo trascurrido es 2.00 h
La distancia recorrida es 180.00 km
La velocidad actual es 80.00 km/h
La velocidad media es 90.00 km/h

1. Acelerar
2. Decelerar
3. Actualizar
4. Consultar
5. Salir
Elige una opción: 5
¡Adiós!
```

Y supón (sin necesidad de comprobarlo) que el usuario siempre proporciona datos correctos por teclado.

Ejercicio 5 (Obligatorio)

Utiliza la clase `Coche` definida anteriormente para hacer un programa que nos permita simular una carrera de coches con las siguientes características:

- Los coches tendrán como objetivo recorrer una cierta distancia. Esta distancia ha de leerla el programa por teclado.

- A todos los coches de la carrera se les asignará, en principio, una misma velocidad inicial. Esta velocidad también se leerá por teclado.
- Cada coche tendrá su propio plan de carrera y todos esos planes se leerán de una matriz de enteros. El programa tendrá que leer por teclado el nombre del fichero donde se encuentre almacenada la correspondiente matriz[2]. He aquí qué representa esa matriz:
  - Cada fila será el plan de carrera de un coche diferente. Por lo tanto, el programa deberá crear una lista con tantos objetos de la clase `Coche` como filas haya en esa matriz.
  - Cada elemento de la fila representará la aceleración que debe aplicarse al coche en el instante correspondiente: la del elemento en la posición 0 deberá aplicarse en el instante inicial (y, por tanto, modificará la velocidad inicial del coche); la del elemento en la posición 1 deberá aplicarse tras una hora de carrera; la del elemento 2, una hora después... y así sucesivamente. Si a los coches se les acaban los planes antes de que la carrera finalice, cada uno pasará a reutilizar el suyo desde su posición 0.
- Cada hora se comprobará si algún coche ya ha recorrido la distancia objetivo. Si se da el caso, la carrera finalizará declarando ganador al coche que mayor distancia lleve recorrida en ese momento. Podrá haber empate y, por lo tanto, varios coches ganadores.

Debes respetar cuidadosamente el comportamiento de entrada/salida que ilustra este ejemplo de ejecución:

```
Introduce el objetivo de la carrera en km: 600
Introduce la velocidad inicial en km/h: 100
Introduce el nombre del fichero de planes de carrera: carrera1.txt

Resultados tras 1 h:
Coche 0: alcanza el km 100 a 100 km/h
Coche 1: alcanza el km 103 a 103 km/h
Coche 2: alcanza el km 104 a 104 km/h
Coche 3: alcanza el km 102 a 102 km/h
Coche 4: alcanza el km 100 a 100 km/h

Resultados tras 2 h:
Coche 0: alcanza el km 200 a 100 km/h
Coche 1: alcanza el km 209 a 106 km/h
Coche 2: alcanza el km 208 a 104 km/h
Coche 3: alcanza el km 213 a 111 km/h
Coche 4: alcanza el km 208 a 108 km/h

Resultados tras 3 h:
Coche 0: alcanza el km 309 a 109 km/h
Coche 1: alcanza el km 316 a 107 km/h
Coche 2: alcanza el km 316 a 108 km/h
Coche 3: alcanza el km 318 a 105 km/h
Coche 4: alcanza el km 316 a 108 km/h

Resultados tras 4 h:
Coche 0: alcanza el km 418 a 109 km/h
Coche 1: alcanza el km 426 a 110 km/h
Coche 2: alcanza el km 428 a 112 km/h
Coche 3: alcanza el km 425 a 107 km/h
Coche 4: alcanza el km 424 a 108 km/h

Resultados tras 5 h:
Coche 0: alcanza el km 527 a 109 km/h
```

[2]Recuerda que en la práctica anterior ya leíste de fichero matrices de enteros haciendo uso del módulo `modulomatrices.py`.

```
Coche 1: alcanza el km 539 a 113 km/h
Coche 2: alcanza el km 540 a 112 km/h
Coche 3: alcanza el km 541 a 116 km/h
Coche 4: alcanza el km 540 a 116 km/h

Resultados tras 6 h:
Coche 0: alcanza el km 645 a 118 km/h
Coche 1: alcanza el km 653 a 114 km/h
Coche 2: alcanza el km 656 a 116 km/h
Coche 3: alcanza el km 651 a 110 km/h
Coche 4: alcanza el km 656 a 116 km/h

Coches ganadores: 2 4
```

Y tampoco en este caso has de prever que el usuario pueda introducir datos incorrectos por teclado. Además, como objetivo de la carrera y como velocidad inicial siempre introducirá enteros.