Problemes Programació Avançada – Curs 2018-19
Tema 1: Programació Orientada a Objectes

Exercici 1 (nivell bàsic)

a) Un nombre racional és un nombre que pot ser expressat com a resultat de la divisió de dos nombres enters, amb el divisor sempre diferent de 0. Per exemple, el 0,5 és racional perquè pot ser descompost per la fracció 1/2. Implementeu una classe anomenada `NombreRacional`, per representar i tractar els nombres racionals, que contingui:

- Atributs per guardar el numerador i el denominador (de tipus enter).
- Un constructor per inicialitzar el numerador i denominador als valors especificats com a paràmetre (per defecte, el numerador i el denominador s'han d'inicialitzar a 0).
- Propietats per implementar els `getters` i `setters` dels atributs de la classe (numerador i denominador). S'han d'anomenar `numerador` i `denominador` respectivament.
- Un mètode per verificar que el divisor sigui diferent de 0, amb la següent declaració:

```
def esValid(self):
```

- Un mètode per obtenir la versió simplificada del nombre racional:

```
def simplifica(self):
```

  Recordeu que per simplificar un nombre racional cal dividir el numerador i el denominador pel màxim comú divisor de tots dos. Per obtenir el màxim comú divisor de dos nombres podeu fer servir l'algorisme que trobareu descrit en aquest enllaç:

  https://es.wikibooks.org/wiki/Implementaci%C3%B3n_de_algoritmos_de_teor%C3%ADa_de_n%C3%BAmeros/Algoritmo_de_Euclides

- La sobrecàrrega dels operadors aritmètics bàsics de suma (`__add__`), resta (`__sub__`), multiplicació (`__mul__`) i divisió (`__truediv__`). El resultat de cada operació s'ha de retornar simplificat.

b) Utilitzeu la classe de l'exercici anterior per crear una funció que rebi com a paràmetres una llista de nombres racionals, un caràcter que ens indica l'operació que volem fer (`'+'`, `'.'`, `'*'`, `'/'`) i un nombre racional, i retorni una nova llista que sigui el resultat d'aplicar l'operació indicada entre cadascun dels elements vàlids de la llista original i el nombre racional que es passa com a paràmetre. Pels elements de la llista original que no siguin vàlids, al resultat hi guardarem un nombre racional amb un 0 al numerador i un 0 al denominador.
La capçalera de la funció serà la següent:

```
def operar(llistaRacionals, operacio, operand):
```

Exercici 2 (nivell mig) – EXERCICI AVALUABLE

Volem implementar dues classes per gestionar les inscripcions a les diferents activitats que ofereix un club esportiu als seus socis (classes esportives, sessions d'entrenament, competicions, etc.). Cada activitat tindrà un número de participants màxim diferent i va dirigida a persones d'una determinada franja d'edat. Per simplificar, suposarem que les activitats es realitzen setmanalment un sol dia per setmana.

- Una classe **`Usuari`** per guardar les dades bàsiques d'un usuari que es registra a una de les activitats: el seu codi de soci del club, el nom i l'edat. La classe haurà de tenir els mètodes següents:
  - Un constructor que rebi com a paràmetres el codi del soci (com un `string`), el nom del soci (`string`) i l'edat del soci (`int`) i inicialitzi les dades de l'objecte.
  - Propietats per implementar els `getters` i `setters` dels atributs de la classe: `codi`, `nom`, `edat`.

  Suposarem que l'edat mínim dels usuaris del club és 18 anys. Per tant, cada cop que es modifiqui el valor de l'edat (ja sigui al constructor o al `setter`) s'ha de comprovar que l'edat és més gran o igual a 18 anys. Utilitzeu la instrucció `assert` per generar un `AssertionError` en cas que no es compleixi.

- Una classe **`Activitat`** per guardar totes les dades necessàries per gestionar les inscripcions a una activitat, com a mínim, el nom de l'activitat, el dia de la setmana i l'horari en què es fa, la franja d'edat (edat mínima i màxima) a la que va dirigida, el nº màxim de participants que s'hi poden inscriure i les dades de totes les persones inscrites. L'edat mínima haurà de ser igual o superior a 18 anys i el nº màxim de participants no pot ser superior a 40. Tant al constructor com als `setters` haurem s'ha de comprovar que es compleixi i generar un `AssertionError` en cas que no es compleixi. La classe haurà de tenir els mètodes següents:
  - Un constructor per inicialitzar les dades bàsiques de l'activitat sense cap participant inscrit. El constructor pot rebre com a paràmetres el nom de l'activitat (`string`), el nº màxim de participants, l'edat mínima i màxima per participar-hi, i el dia (`string`) i hora (`string`) en què es realitza. Si no es passa cap valor als paràmetres l'objecte ha de quedar inicialitzat amb valors per defecte.
  - Propietats per implementar els `getters` i `setters` dels atributs de la classe: `nom`, `dia`, `hora`, `edatMinima`, `edatMaxima`, `maxParticipants`.
  - Un mètode `nParticipants()` per recuperar el nº de participants que s'han inscrit a l'activitat.
  - Un `mètode buscaParticipant` que rebi com a paràmetre el nom d'una persona i retorni un objecte de tipus `Usuari` amb totes les dades (codi, nom, edat) de la persona si està inscrita a l'activitat. Si no està inscrita a l'activitat retornarà `None`.

    ```
    def buscaParticipant(self, nom):
    ```

  - Un mètode `afegeixParticipant` que permeti inscriure un nou participant a una activitat. Aquest mètode rebrà com a paràmetre un objecte de tipus `Usuari` i afegirà l'usuari a la llista d'usuaris inscrits a l'activitat si es compleixen les següents condicions:

- L’edat de la persona ha d’estar dins de la franja marcada per l’activitat.
- La persona no pot estar ja inscrita a l’activitat (considerarem que la persona ja està inscrita si hi ha un altre usuari amb el mateix nom).
- El nº de persones inscrites a l’activitat no pot superar el nº màxim de persones que poden participar a l’activitat.

Si es compleixen totes aquestes condicions s’hauran d’afegir les dades de la persona al registre de participants de l’activitat. Si alguna condició no es compleix s’haurà de generar un `AssertionError` amb la instrucció `assert`.

```
def afegeixParticipant(self, usr):
```