

**VERSIÓ 1**

Volem crear les classes necessàries per poder gestionar un portal d'aula interactiva, similar al que podem trobar a Caronte o al Campus Virtual, amb activitats i recursos que els professors d'una assignatura publiquen perquè els estudiants puguin descarregar-los, visualitzar-los o fer lliuraments.

**Exercici 1 (2 punts)**

De moment, us donem ja feta una classe `Usuari` que permet guardar el tipus d'usuari ('E': estudiant, 'P': professor), el niu, el nom i l'adreça de mail de tots els usuaris registrats al portal. Aquesta classe té definits els getters i setters per recuperar i modificar els valors dels atributs.

A partir d'aquesta classe `Usuari` us demanem que deriveu classes més específiques per estudiants i professors.

- La classe `Estudiant` ha de poder guardar una llista de totes les notes (només el valor numèric) que l'estudiant obté en les diferents activitats avaluable del curs. D'aquesta classe només heu d'implementar el constructor.
- La classe `Professor` ha de poder guardar la llista amb el nom de totes les activitats que ha creat (vegeu el següent exercici per l'explicació de les activitats). Aquesta classe ha de tenir els mètodes següents:
  - `registra_activitat`: serveix per registrar que el professor ha creat una activitat determinada, a partir del nom de l'activitat.
  - `consulta_activitat`: per consultar si el professor ha creat una activitat amb un nom determinat.

**Exercici 2 (2 punts)**

Us donem ja feta una classe `Document` que permet guardar la informació d'un document que es publica al portal perquè es pugui descarregar (un fitxer .pdf, una presentació, un fitxer amb codi, etc.). Aquesta classe té atributs per guardar el nom que es visualitza al portal, una descripció del seu contingut, i el nom del fitxer que es pot descarregar. La classe té tot els getters i setters necessaris.

Els documents són només una de les moltes activitats que hem de poder publicar al portal. Per exemple, hem de poder publicar també lliuraments d'exercicis, inscripcions a grups, etc.

Per tant, volem crear una classe base genèrica `Activitat` que permeti agrupar tot el que és comú a tots els tipus d'activitats. Totes les activitats tindran, igual que els documents, un nom i una descripció. En canvi el nom del fitxer és una propietat específica dels documents que no tindran altres tipus d'activitats.

A nivell de mètodes, totes les activitats hauran de tenir un mètode `visualitza` que serà el mètode que es cridi quan un usuari obri l'activitat per accedir al seu contingut. El mètode rebrà com a paràmetre un objecte de tipus `Usuari` amb les dades de l'usuari que vol visualitzar l'activitat.

Aquest mètode ha de ser específic per cada tipus d'activitat. Cada tipus d'activitat haurà de tenir la seva pròpia implementació completament diferent de les altres. No té sentit implementar un mètode `visualitza` genèric per la classe `Activitat` i per tant, tampoc té sentit poder crear objectes de la classe `Activitat`.

Us demanem que creeu la classe `Activitat`, tenint en compte tot el que acabem d'explicar i que modifiqueu tot el que calgui a la classe `Document` perquè derivi correctament d'aquesta nova classe `Activitat`.

En una aplicació real, el mètode `visualitza` de la classe `Document` hauria de permetre obrir i mostrar el document associat. Com que això s'escapa de l'abast del que podem fer a l'examen, farem que el mètode `visualitza` simplement mostri el nom del fitxer per pantalla.

### **Exercici 3 (3 punts)**

Un altre tipus d'activitat que podem tenir, apart dels documents, són les inscripcions a grups, que permeten als estudiants apuntar-se a grups de pràctiques, problemes, etc. Us demanem que creeu una nova classe `Inscripcio` que derivi de la classe `Activitat`, per gestionar tota la informació necessària per fer inscripcions a grups, segons el que expliquem a continuació.

Les inscripcions tindran com a informació específica la data límit per poder fer la inscripció, el nº de grups possibles, el número d'estudiants que es poden apuntar a cada grup (tots els grups tindran el mateix nº màxim d'estudiants) i una llista de tots els grups amb els estudiants que s'hi ha apuntat.

Per poder guardar aquesta llista de grups us donem ja feta una classe `Grup` que permet guardar el codi del grup i una llista amb els nius de tots els estudiants inscrits. Té els getters i setters necessaris i també un mètode `afegeix_estudiant` per afegir el niu d'un estudiant al grup.

La classe `Inscripcio` ha de tenir un mètode `crea_grups` que permeti inicialitzar tots els grups disponibles. Aquest mètode ha d'afegir a la llista de grups tants grups com indiqui el valor de l'atribut que guarda el nº de grups possibles de la inscripció. Els codis dels grups s'han de generar de forma consecutiva des de 0 fins al nº de grups menys 1. Tots els grups han de quedar inicialitzats buits, sense estudiants inscrits.

Per la classe `Inscripcio` el mètode `visualitza` ha de permetre que un estudiant realitzi la inscripció a un dels grups. S'haurà de demanar per teclat el codi del grup al que es vol inscriure i comprovar que la data actual és inferior a la data límit d'inscripció, que queden places lliures en el grup i que l'estudiant no està ja inscrit a cap dels grups. Si no, no es podrà fer la inscripció i s'haurà de generar un missatge d'error.

Per poder guardar i gestionar les dates, recordeu que igual que hem fet en algun dels exercicis avaluable podem utilitzar el mòdul `datetime`. En aquest mòdul teniu una classe `date` que us permet guardar una data, inicialitzant un objecte `date` d'aquesta forma:

```
date(any, mes, dia)
```

Dos objectes de tipus `date` es poden comparar amb l'operador de comparació `<` habitual. També podeu fer servir la funció `today()` per recuperar el dia actual.

**Exercici 4 (3 punts)**

Utilitzant les classes que hem fet fins ara volem crear una classe `Assignatura` que agrupi totes les activitats d'una assignatura. Aquesta classe haurà de guardar un nom, la llista de totes les activitats (de qualsevol tipus) que s'han creat i la llista de tots els usuaris (estudiants i professors) registrats a l'assignatura. Haurà de tenir els mètodes següents:

- `llegeix_usuaris`: permet llegir i guardar les dades de tots els usuaris d'un fitxer que tingui aquest format:

```
E NIU_1 NOM_1 MAIL_1
P NIU_2 NOM_2 MAIL_2
E NIU_3 NOM_3 MAIL_3
E NIU_4 NOM_4 MAIL_4
P NIU_5 NOM_5 MAIL_5
```

És a dir, una línia per cada usuari amb els valors separats per espais en blanc, on el primer valor és el tipus d'usuari ('E': Estudiant, 'P': Professor).

- `afegeix_activitat`: permet afegir una activitat de qualsevol tipus a l'assignatura. S'haurà de llegir per teclat la informació comuna a totes les activitats i, després, en funció del tipus d'activitat la resta d'informació específica. El mètode rep com a paràmetre un objecte amb les dades de l'usuari que vol afegir l'activitat. S'ha de comprovar que l'usuari sigui un professor. Si no s'ha de generar un missatge d'error. L'activitat ha de quedar afegida a la llista d'activitats de l'assignatura i al registre d'activitats creades pel professor.
- `get_activitat`: a partir del nom d'una activitat, s'ha de retornar un objecte amb totes les dades de l'assignatura corresponent. S'ha de comprovar que existeixi una activitat amb aquest nom.
- `visualitza_activitat`: permet visualitzar una activitat cridant al mètode corresponent de l'activitat, segons el seu tipus. Rep com a paràmetres el nom de l'activitat i el niu de l'usuari que la vol visualitzar. S'ha de comprovar que tant l'activitat com l'usuari existeixin a l'assignatura.

**Notes:**

- Tingueu en compte que el criteri principal d'avaluació no serà que el codi funcioni correctament, sinó que sigui conceptualment correcte i s'ajusti als que hem explicat a classe.
- Podeu afegir tot el que creieu que sigui necessari (atributs, mètodes) a les classes que s'indiquen, si penseu que us ajuda en la implementació del que es demana.
- Quan parlem de llistes no estem predeterminant que s'ha d'utilitzar un objecte de tipus `list`. Podeu utilitzar llistes o diccionaris en cada cas, segons considereu més convenient.