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

Exercici 3 (nivell bàsic)

Utilitzant herència volem crear una jerarquia de classes que ens permeti guardar i gestionar dades dels alumnes i professors de la UAB.

1. Per començar, haurem de declarar una classe `Assignatura` que contingui la informació d'una assignatura i que ens servirà per poder guardar les dades de les assignatures que fan tant professors com estudiants.
   1. La classe `Assignatura` ha de tenir atributs per guardar el nom, el nº de crèdits que li suposa a l'estudiant i el nº de crèdits que li suposa al professor.
   2. El constructor haurà de permetre inicialitzar els tres atributs en aquest ordre: n om, nº crèdits de l'estudiant, nº de crèdits del professor.
   3. Haurà de tenir propietats `nom`, `creditsEstudiant` i `creditsProfessor` per recuperar el valor dels atributs.

2. Com a classe base de la jerarquia d'herència haurem de declarar una classe `Persona` que contingui la informació comuna d'alumnes i professors. D'aquesta classe base, derivarem després classes específiques per alumnes i professors.
   - La classe `Persona` haurà de guardar el niu i el nom de la persona (ja sigui un estudiant o un professor) i la llista d'assignatures a les que està vinculat (en el cas d'estudiants seran les assignatures de les que està matriculat i en el cas de professors, les assignatures en les que imparteix docència).
   - Haurà de tenir propietats `nom` i `niu` per recuperar el nom i el niu de la persona.
   - Tindrà un mètode `cercaAssignatura` que rebrà com a paràmetre el nom d'una assignatura i retornarà un booleà indicant si l'assignatura amb aquest nom està a la llista d'assignatures de la persona.
   - Tindrà un mètode `afegeixAssignatura` per afegir les dades d'una assignatura a la llista d'assignatures. Rebrà com a paràmetre un objecte de la classe `Assignatura`. Si l'assignatura ja està a la llista no l'ha de tornar a afegir. Retornarà un booleà indicant si l'assignatura s'ha afegit a la llista o no.

3. Derivar una classe `Estudiant` a partir de la classe base `Persona` de l'apartat anterior. A part de tota la informació derivada de `Persona`, haurà de tenir un nou atribut per guardar la titulació a la que està matriculat l'estudiant.
   - Afegiu una propietat `titulació` per recuperar el valor de la titulació de l'estudiant
   - Afegiu un mètode `creditsMatriculats` per retornar la suma de crèdits-estudiant de totes les assignatures de les que està matriculat.

4. Derivar una classe `Professor` a partir de la classe base `Persona` de l'apartat anterior. A part de tota la informació derivada de `Persona`, haurà de tenir atributs per guardar el departament en el que està adscrit i el nº màxim de crèdits de docència que hauria de fer.
   - Afegiu una propietat `departament` i `maximCredits` per recuperar els valors del departament i del nº màxim de crèdits del professor.
   - Afegiu un mètode `creditsImpartits` per retornar la suma de crèdits-professor de totes les assignatures en les que imparteix docència.

Exercici 4 (nivell mig)

Volem aprofitar la jerarquia de classes `Producte`, `Llibre` i `Electrodomestic` que hem anat desenvolupant durant les sessions de classe dins d'un programa que gestioni totes les vendes d'una cadena comercial.

D'entrada us donem ja implementat el codi que hem fet a les sessions de classe. Al codi de les classes `Llibre` i `Electrodomestic` hi haureu d'afegir un mètode descompte que rebi com a paràmetre el nº d'unitats de producte que es volen vendre i calculi el percentatge de descompte que s'haurà d'aplicar en funció del tipus de producte i del nº d'unitats de la venda. Pels llibres, s'aplicarà un descompte del 10% si el nº d'unitats és més gran de 100 i del 5% si és més gran que 10. Si és inferior no s'aplicarà cap descompte. Pels electrodomèstics el descompte serà del 10% si el nº d'unitats és més gran que 1.

Pel que fa a les vendes dels productes es poden fer presencialment en una botiga física de la cadena o bé de forma online, per internet. El tipus d'informació que necessitem guardar és diferent segons el tipus de venda.

Per les vendes que es fan presencialment a una botiga necessitem guardar el codi del producte, el nº d'unitats que s'han comprat, la data de la compra i un identificador de la botiga on s'ha fet la compra.

Per les vendes que es fan per internet, apart del codi i nº d'unitats del producte i la data de la compra (igual que per les compres presencials), necessitem guardar l'adreça d'enviament.

- Crear i implementar amb herència la jerarquia de classes necessària per guardar els diferents tipus de vendes amb la informació que s'ha descrit anteriorment i els mètodes que facin falta per poder implementar el que s'explica a continuació.
- Crear i implementar una classe `GestioVendes` que inclogui un llistat de tots els productes del catàleg i un llistat únic de totes les vendes (ja siguin presencials o online) que es realitzen. Aquesta classe haurà de tenir a la interfície pública, almenys els mètodes següents:
    - Un mètode `llegeixProductes` que permeti llegir d'un fitxer la informació de tots els productes del catàleg. Rebrà com a paràmetre el nom del fitxer. El fitxer tindrà el format següent:

      TIPUS_PROD_1 // (suposem Llibre)
      CODI_1
      PREU_ 1
      TITOL_1
      AUTOR_1
      N_PAGINES_1
      TIPUS_PROD_2 // (suposem Electrodomèstic)
      CODI__2
      PREU_ 2
      MARCA_2
      MODEL_2
      VOLUM_2
      ...

El tipus de producte serà 0 si el producte correspon a un llibre i 1 si correspon a un electrodomèstic. Per cada llibre, apart del codi i del preu tindrem el títol, l'autor i el nº de pàgines i pels electrodomèstics, apart del codi i del preu tindrem la marca, el model i el volum de l'embalatge.

- Un mètode `afegeixVendaPresencial` que permeti afegir les dades d'una nova venda presencial, rebent com a paràmetre el codi i nº d'unitats del producte, la data de la compra i l'identificador de la botiga. El mètode ha de calcular i retornar l'import total de la compra tenint en compte el preu del producte, el nº d'unitats i el descompte a aplicar segons el tipus de producte i el nº d'unitats. Si el codi de producte no existeix a la llista de productes ha de retornar -1.

- Un mètode `afegeixVendaOnLine` que permeti afegir les dades d'una nova venda per internet, rebent com a paràmetre el codi i nº d'unitats del producte, la data de la compra i l'adreça d'enviament. El mètode ha de El mètode ha de calcular i retornar l'import total de la compra tenint en compte el preu del producte, el nº d'unitats, les despeses d'enviament i el descompte a aplicar segons el tipus de producte i el nº d'unitats (tingueu en compte que les despeses d'enviament no depenen del nº d'unitats, són fixes independentment del nº d'unitats).

- Un mètode `escriuVendes` que permeti escriure en un fitxer les dades de totes les vendes que s'han fet. Rebrà com a paràmetre el nom del fitxer on es volen escriure les dades. El fitxer de sortida haurà de tenir el format següent:

  CODI_PRODUCTE_1
  N_UNITATS_1
  DATA_1
  IMPORT_TOTAL_1
  BOTIGA_1 (si venda presencial)
  CODI_PRODUCTE_2
  N_UNITATS_2
  DATA_2
  IMPORT_TOTAL_2
  DESPESES_ENVIAMENT_2 (si venda online)
  ADREÇA_ENVIAMENT_2 (si venda online)
  ...

  Com podem veure, les dades que s'han d'escriure al fitxer són diferents en funció del tipus de venda.

Exercici 5 (nivell mig) – EXERCICI AVALUABLE

Volem gestionar tots els préstecs que es fan de les publicacions que tenim guardades en una biblioteca. Les publicacions de la biblioteca poden ser de dos tipus, llibres o revistes periòdiques. Dels llibres hem de guardar el títol, l'autor, el nº de còpies que hi ha a la biblioteca del llibre i el nº de dies que es pot prestar (el nº de dies de préstec es fixa per cada llibre en funció de la demanda que té). De les revistes periòdiques hem de saber el títol i la periodicitat (mensual, trimestral o anual). Totes les publicacions (llibres o revistes) tenen un codi que les identifica. Les revistes a més a més tenen exemplars. Cada exemplar té un codi que l'identifica i hem de poder guardar per cada exemplar, si està prestat o no. Els préstecs de les revistes només es poden fer d'exemplars particulars. Suposarem que dels exemplars de les revistes en tenim només una còpia. Tant revistes com llibres s'han de representar amb classes derivades d'una classe abstracta `Publicacio`.

Apart de les dades de totes les publicacions volem guardar també les dades de tots els préstecs que s'han fet. De cada préstec s'ha de guardar un identificador de l'usuari que fa el préstec, el codi de la publicació, la data de préstec i la data en què s'ha de fer el retorn de la publicació. La data de retorn depèn del tipus de publicació. Pels llibres, ha de ser la data de préstec més el nº de dies de préstec que es guarda per cada llibre. Pels exemplars de les revistes el nº de dies de préstec sempre és igual a 30.

Heu de crear i implementar una classe `Biblioteca` que permeti gestionar tota la informació de les publicacions i dels préstecs. Aquesta classe ha de tenir un llistat conjunt de publicacions independentment del seu tipus. Haurà de tenir, almenys els mètodes següents:

- Un mètode `llegeixPublicacions` que permeti llegir d'un fitxer la informació de totes les publicacions de la biblioteca. Rebrà com a paràmetre el nom del fitxer. El fitxer tindrà el format següent:

  TIPUS_PUBLICACIO_1 // (suposem Llibre)
  CODI_1
  TITOL_1
  AUTOR_1
  N_COPIES_1
  N_DIES_1
  TIPUS_PUBLICACIO_2 // (suposem revista)
  CODI_2
  TITOL_2
  PERIODICITAT_2
  Nº_EXEMPLAR_1 Nº_EXEMPLAR_2 Nº_EXEMPLAR_3 ....
  ...

  El tipus de publicació serà 'L' si la publicació correspon a un llibre i 'R' si correspon a un exemplar d'una revista. Fixem-nos que per totes les publicacions hem de llegir el codi i el títol. Pels llibres, a més a més, haurem de llegir l'autor, el nº de còpies que hi ha a la biblioteca d'aquest llibre i el nº de dies que es pot deixar en préstec. Per les revistes, llegirem la periodicitat i després tenim una línia amb tots els números d'exemplar que es tenen, separats per un espai en blanc.

- Un mètode `presta` que permeti afegir les dades d'un nou préstec a la llista de préstecs, rebent com a paràmetre l'identificador d'usuari, el codi de la publicació i la data del préstec i només en les revistes el nº d'exemplar que s'ha de prestar (pels llibres ha de tenir 0 com a valor per defecte). El mètode ha de retornar dos valors: un booleà que indiqui si el préstec s'ha pogut realitzar o no i la data esperada de retorn de la publicació, calculada segons si és un llibre o un exemplar d'una revista segons s'ha explicat abans.
  Si el codi de la publicació no existeix dins del catàleg de la biblioteca no es podrà fer el préstec i retornarem `False`. Igualment si existeix però no es pot prestar el llibre o la revista també retornarà `False`. Pels llibres haurem de controlar que el nº de préstecs actius no sigui superior al nº de còpies del llibre. Pels exemplars de revista que l'exemplar especificat no estigui ja prestat (només en tenim una còpia). En qualsevol altre cas retornarà `True`.

- Un mètode `retorna` que permeti registrar que es retorna un préstec i l'elimini de la llista de préstecs. Aquest mètode rep com a paràmetres tots els valors que permeten identificar un préstec: identificador d'usuari, codi de publicació i nº d'exemplar (només en el cas d'exemplar de revistes, agafa el valor per defecte 0 en el cas dels llibres). Rep també com a paràmetre la data en què es fa el retorn de la publicació.
  El mètode ha de retornar dos valors: el primer ha de ser un booleà que indiqui si es pot fer la devolució del préstec. Si el préstec (identificador d'usuari, codi de publicació i nº exemplar si cal) no existeix a la llista de préstecs o si el préstec no es pot retornar perquè la publicació no estava prestada s'ha de retornar `False`. En cas contrari `True`
  El segon valor de retorn ha de ser un booleà que indiqui si la devolució es fa dins del període determinat quan es va fer el préstec o no. És a dir, s'ha de comparar la data de retorn que es passa com a paràmetre amb la data prevista de retorn del préstec (calculada quan s'afegeix el préstec) i si hem superat la data límit s'ha de retornar `False` i si no `True`.

Per **gestiona**r totes les **dates** heu de fer servir la classe **`date`** que trobareu definida en el mòdul **`datetime`**, https://docs.python.org/3.5/library/datetime.html