Departament de Disseny i Programació de Sistemes Electrònics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

EXERCICI PUNTUABLE TECPRO

05/04/2019 Grau en Enginyeria de Sistemes TIC

COGNOMS:

NOM: GRUP de LAB:

Exercici 1. El diagrama UML. Una petita fàbrica de mecanitzats ens ha encarregat un programari per tal de portar a terme la gestió de la seva producció.

La fàbrica consta d'un cert nombre de premses de mecanitzat. D'aquestes premses ens interessa guardar el seu codi de referència, la força (en Tm) i la dimensió de la taula (en mm). A la fàbrica existeixen dos tipus diferents de premses: mecàniques i hidràuliques. Les premses mecàniques tenen un paràmetre que indica el nombre de cops/min que pot fer la premsa. Les hidràuliques tenen dos paràmetres que indiquen, respectivament, la velocitat de treball mínima i màxima de la premsa (en mm/seg.). Aquests paràmetres s'han de tenir en consideració en el càlcul del rendiment (nombre de peces/hora) que la premsa pot donar per a cada tipus de producte, tal com s'explica més endavant.

La fàbrica ja disposa d'una base de dades de les seves premses, així com dels seus clients i dels productes (tipus de peces) que pot fabricar. Dels clients es guarda el nom de l'empresa, el NIF i l'adreça. A la fàbrica es fan dos tipus de productes: plàstics i metàl·lics, que es diferencien pel tipus d'eina que cal fer servir a la premsa. Dels productes es guarda un codi de referència, el preu de venda (/unitat) i un paràmetre de complexitat de fabricació (valors de l'1 al 5) que intervé en el càlcul del rendiment de la premsa en la que es fabriqui aquest producte. La gestió que se'ns ha encarregat consisteix en:

- Entrar comandes: Les comandes venen definides per:
  - Producte/s a fabricar.
  - Quantitat de cada tipus de producte.
  - Client que fa la comanda.
  - Data d'entrega.
  - Estat: rebuda (=0), en procés (=1), o finalitzada (=2).
- Planificar treball: A partir de les comandes rebudes, generar les ordres de treball corresponents i assignar com estat de la comanda "en procés". Una comanda podrà donar lloc a una o més ordres de treball. En una ordre de treball s'especificaran les dates d'inici i de final, així com la premsa en la que es realitzarà el treball. Per estimar la data de final s'ha de calcular el rendiment de la premsa en la que es realitzarà el treball, que depèn del tipus de premsa i del paràmetre de complexitat del producte a fabricar.
- Preparar entregues: Consistiria en generar un llistat de totes aquelles comandes que, a la data en que s'executi aquest mètode, hagin finalitzat totes les seves ordres de treball associades. L'estat d'aquestes comandes s'assignaria com "finalitzada".

NOTA: Òbviament, s'entén que la gestió descrita en aquest enunciat correspon a una versió molt simplificada de la realitat d'una fàbrica, però volem remarcar que en l'exercici heu de considerar, exclusivament, la funcionalitat descrita a l'enunciat. Així, es demana:

Dibuixeu **el diagrama UML** que doni resposta a les especificacions anteriors. Per cadascuna de les classes creades, afegiu una breu indicació de quins atributs contindria cada classe, per permetre les relacions que heu detectat.

Exercici 2. Classes, objectes, relacions i excepcions. Donada la següent definició de classes, [Apartat a] Escriu el resultat de les expressions contingudes en el main i que estan numerades. Si no retorna res, escriu None. Si es tracta d'un error, escriu Error i una breu explicació del perquè.

```
import math
class classe1(object):
    x = -3
    y = 100
    def __init__(self, y):
        self.x = y
    def methodA1(self, y):
        self.x = min(self.x, y)*2
    def ask(self):
        try:
            return self.wait(self.x)
        except Exception as e:
            return "not found"
    def __str__(self):
        return self.__class__.__name__+"--"+str(self.x)

class classe2(classe1):
    def methodB(self):
        self.y = math.pow(self.x,2)
    def wait(self,x):
        return x*-2
class classe3(classe1):
    def methodA1(self):
        self.x = max(self.x, classe1.y)
    def wait(self,x):
        return x*2
class classe4(classe3):
    pass
```

[Apartat a] Escriu els resultats que es mostraran per pantalla després de l'execució del main que segueix.

```
if __name__=='__main__':
    classe1=classe1(5)
    classe1.methodA1(-2)
    print "1.1",classe1
    classe1.methodA1(1)
    print "1.2",classe1
    classe2=classe2(3)
    classe2.methodA1(-5)
    print "1.3",classe2
    classe2.methodB(),
    print "1.4",classe2
    print "1.5",classe2.ask()
    classe3=classe3(22)
    classe3.methodA1()
    print "1.6",classe3
    print "1.7",classe3.ask()
    classe4=classe4(5)
    print "1.8",classe4.ask()
```

[Apartat b] Ara afegim la següent definició de classe contenidora elsF, i se us demana justificar què s'escriurà per pantalla en executar el main.

```
class TheClasses(object):
    def __init__(self):
        self.dades={}
    def add(self,dada):
        if dada not in self.dades:
            self.dades[dada]=repr(dada)
        else:
            raise Exception("already")
    def __iter__(self):
        return iter(self.dades)
    def __str__(self):
        return "Classes info "+">>>".join([str(element) for element in self])

if __name__=='__main__':
    k=TheClasses()
    try:
        k.add(classe1)
        k.add(classe4)
        k.add(classe1)
    except Exception as e:
        print "1.9",e
    print "1.10",k
```

[Apartat c] Dibuixeu el diagrama UML resultant que contingui les classes anteriors.

Exercici 3. Recursivitat. [Apartat a] Dissenya òptimament la funció booleana recursiva *esPrimer*, tal que donat un nombre natural, retorni True si es tracta d'un nombre primer, i False en cas contrari. Un nombre és primer, si i només si, és divisible per 1 i per ell mateix.

```
def esPrimer(n, i = 2):
    """
    >>> esPrimer(15)
    False
    >>> esPrimer(13)
    True
    """
```

[Apartat b] Dissenya òptimament la funció recursiva *crack*, tal que, donada una llista de caràcters, s'encarrega de generar totes les contrasenyes possibles a partir d'aquests caràcters. És a dir, permet generar totes les permutacions possibles de paraules utilitzant els caràcters de la llista, amb repeticions i fins a una longitud donada. Es poden utilitzar funcions addicionals.

```
Exemple 1:
Input : llista = [a, b],
        len = 2.
Output :
a b aa ab ba bb

Exemple 2:
Input : llista = [a, b, c],
        len = 3.
Output :
a b c aa ab ac ba bb bc ca cb cc aaa aab aac aba abb
abc aca acb acc baa bab bac bba bbb bbc bca bcb bcc
caa cab cac cba cbb cbc cca ccb ccc
```