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

Prova final de TECPRO

16/06/2015 Grau en Enginyeria de Sistemes TIC

COGNOMS:

NOM: GRUP de LAB:

Llegiu atentament els enunciats i responeu les preguntes que segueixen. Lliureu les respostes a cada exercici en fulls separats.

Exercici 1. Comprensió de codi. Donats els següents fragments de codi, escriu els resultats que es mostraran per pantalla.

Apartat a)

```
class Forma(object):
    def __cmp__(s1, s2):
        return cmp(s1.area(), s2.area())

class Quadrat(Forma):
    def __init__(self, h):
        self.side = float(h)

    def area(self):
        return self.side**2

    def __str__(self):
        return 'Quadrat with side ' + str(self.side)

class Cercle(Forma):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14159*(self.radius**2)

    def __str__(self):
        return 'Cercle with radius ' + str(self.radius)

def f(L):

    if len(L) == 0: return None
    x = L[0]
    for s in L:
        if s >= x:
            x = s
    return x

s = Quadrat(4)
print s.area()
L = []
shapes = {0:Cercle, 1: Quadrat}
for i in range(10):
    L.append(shapes[i%2](i))
print L[4]
print f(L)
```

Apartat b)

```
def f(L):
    result = []
    for e in L:
        if type(e) != list:
            result.append(e)
        else:
            return f(e)
    return result

print f('3')
print f([1, [[2, 'a'], ['a','b']], (3, 4)])
print f(3)
```

Apartat c)

```
class F(object):
    x = 0
    y = 10
    def __init__(self, y):
        self.x = y
    def a1(self, y):
        self.x = max(self.x, y)
        return self.x
    def a2(self):
        self.x = max(self.x, self.y)
        return self.x
class G(F):
    def b(self):
        self.y = self.x * self.x
        return self.y

f=F(5)
print f
print f.a1(7)
g=G(3)
print f.a1(3)
print g.a1(5)
print g.b()
print f.a2()
print g.a2()
g=G()
f.b()
```

Apartat d)

```
from stack import Stack
s=Stack()
s.push(22)
s.push(1)
s.push(22)
s.push(1)
s.push(22)
while not s.isEmpty():
    z=s.top()
    if z==22:
        s.push(1)
    elif z==1:
        print "Trobat 1"
        s.pop()
        s.pop()
    else:
        print z
        s.pop()
```

Exercici 2. Qüestions Generals.

Apartat a) Defineix breument els conceptes que segueixen.

1. Classe abstracta.
2. Mètode constructor.
3. Redefinició de mètodes.
4. Sobrecàrrega de mètodes.

Apartat b) Respon clarament les següents qüestions.

1. Diferències entre agregació i composició.
2. Diferències entre atributs públics i atributs privats.

Apartat c). Analitza la classe següent i respon les preguntes.

```
class arbreBinari(object):
    def __init__(self,v,left=None,right=None):
        self.v=v
        self.left=left
        self.right=right

    def misteri(self):
        if self is None: return 0
        else:
            if self.left is None and self.right is None:
                return self.v
            else:
                return self.left.misteri()+self.right.misteri()+self.v

    def nose(self, t):
        if t == self.v:
            return True
        elif t < self.v:
            if self.left is not None:
                node = self.left.nose(t)
        else:
            if self.right is not None:
                node = self.right.nose(t)
        return True
```

1. L'objectiu del mètode misteri és
2. Donat un arbre binari de n elements, el cost assimptòtic en cas pitjor del mètode misteri és
3. L'objectiu del mètode nose és
4. Donat un arbre binari de n elements, el cost assimptòtic en cas pitjor del mètode nose és

Exercici 3. Implementació de classes.

L'objectiu del problema següent consisteix a la gestió d'elements químics, la taula periòdica d'elements, i finalment els elements que composen una molècula. A tal efecte, se us demana:

Apartat a) Implementar la classe *Element*, tal que permeti gestionar el nom de l'element, el número atòmic, el símbol i la massa atòmica. L'accés als atributs cal fer-lo a través de mètodes accessors. Afegiu un mètode tal que permeti obtenir la informació de l'element, en el format del joc de proves que segueix.

Donat el codi que segueix,

```
first=Element("Sodium",11,"Na",22.98977)
print first
```

Cal que es mostri per pantalla la informació de l'element tal i com es mostra a continuació.

```
Element(Sodium,11,Na,22.98)
```

Apartat b) Implementar la classe *Taula*, que ha de gestionar un contenidor d'elements químics de manera òptima, atenent als mètodes que es detallen a continuació. Suposeu que només es faran cerques pel nom de l'element químic.

Les dades dels elements químics es proporcionen a través d'un fitxer de nom *nameFile*, que és un fitxer *.csv* amb el següent format de contingut:

```
Sodium,11,Na,22.98
Hydrogen,1,H,1.01
Carbon,6,C,12.01
Nitrogen,7,N,14.01
Oxygen,8,O,16
Phosphorus,15,P,30.98
Sulfur,16,S,32.06
Potassium,19,K,39.1
```

En la següent definició de mètodes, cal que gestioneu totes les excepcions que es puguin produir. Utilitzeu, si us cal, altres mètodes addicionals propis de les definicions de classes.

1. Cal implementar el mètode constructor de la classe *Taula*. Aquest mètode ha de cridar al mètode *readFile* per tal d'inicialitzar els *Elements* que ha de contenir.

   ```
   def __init__(self,nameFile):
           """
           Crea una taula amb els elements continguts en el fitxer .csv de nom nameFile. Usa el metode readFile.
           """
   ```

2. Cal implementar el mètode *readFile*, tal que, reb com a paràmetre el *nameFile.csv* i ha de llegir el fitxer línia per línia. Aquest mètode ha de ser invocat pel constructor de la classe *Taula*.

   ```
    def readFile(self,nameFile):
           """
           llegeix el fitxer .csv nameFile i gestiona el contenidor d'elements
           """
   ```

3. Cal implementar el mètode *searchSymbol(name)*, tal que, donat el nom de l'element, en mostri la informació associada. Únicament es faran cerques pel nom de l'element químic.

   ```
   def searchSymbol(self,name):
          """
          Mostra per pantalla la informació associada a l'element de nom name
          """
   ```

4. Cal implementar el mètode *showAll*, tal que proporcioni tota la informació dels elements emmagatzemats a la taula.

```
def showAll(self):
        """
        Mostra per pantalla la informació associada de tots els elements de la taula
        """
```

A continuació segueix un exemple de funcionament, i dels resultats esperats del mètode *showAll* i del mètode *searchSymbol*.

```
t=Taula("nameFile.csv")
t.showAll()
t.searchSymbol("Sodium")
```

Resultats esperats:

```
List of elements in the table ..........................
Element(Oxygen,8,O,16.0)
Element(Sodium,11,Na,22.98)
Element(Potassium,19,K,39.1)
Element(Nitrogen,7,N,14.01)
Element(Sulfur,16,S,32.06)
Element(Carbon,6,C,12.01)
Element(Hydrogen,1,H,1.01)
Element(Phosphorus,15,P,30.98)

Info for element Sodium ............................
Element(Sodium,11,Na,22.98)
```

Apartat c) Considereu per exemple la molècula aigua, amb fórmula química H_2O. A continuació se us demana gestionar la classe *Molècula*. Per tant, us caldrà gestionar un contenidor òptim que permeti emmagatzemar els àtoms dels elements químics dels quals està composada. En aquest cas, cal gestionar l'accés òptim per símbol.

1. Cal implementar el mètode constructor de la classe *Molècula*

   ```
   def __init__(self,name):
         """
         Crea una molècula de nom name i inicialitza el contenidor d'elements
         """
   ```

2. Cal implementar el mètode *add*, que afegeix un element a la *Molècula*

   ```
   def add(self,Element):
       """
       afegeix l'Element al contenidor de la molècula
       """
   ```

3. Cal implementar el mètode *addAtom*, que reb 2 paràmetres: *n* i *symbol*, i ha d'afegir n àtoms de l'element símbol a la molècula

   ```
   def addAtom(self,symbol,n):
       """
       afegeix n àtoms a l'elements amb símbol symbol de la molècula
       """
   ```

4. Cal implementar el mètode *atoms*, tal que ha de retornar la informació dels elements que composen la molècula, juntament amb el seu nombre d'àtoms

   ```
   def atoms(self):
       """
       mostra per pantalla la informació dels elements que composen la molècula i el seu nombre d'àtoms
       """
   ```

5. Cal implementar el mètode *weight*, tal que ha de retornar la massa atòmica de la molècula. És a dir, la suma dels pesos de cada àtom en la molècula. Per exemple, donats els pesos dels Elements Hidrogen i Oxigen són 1.01 i 16.0 respectivament, i donat que en la molècula aigua hi ha 2 àtoms d'hidrogen i un àtom d'oxigen, la massa de la molècula ha de ser 1.01*2+16.0*1=18.02

```
    def weight(self):
        """
        retorna la massa atòmica de la molècula
        """
```

A continuació segueix un exemple de crides i el resultat esperat.

Exemple d'execució:

```
if __name__==’__main__’:
    Water=Molecula("Water")
    hidrogen=Element("Hidrogen",1,"H",1.01)
    oxigen=Element("Oxigen",8,"O",16.0)
    Water.add(hidrogen)
    Water.add(oxigen)
    Water.addAtom("H",2)
    Water.addAtom("O",1)
    Water.atoms()
    print Water.weight()
```

Resultat esperat:

```
Molecule information Water .....................................
Water H2 O1
Molecule weight Water ..........................................
18.02
```