Série TP n°5 2022-2023	Module	MTI Méthd. et tech d'implement.
	Filière	Master GSI 1 ère Année
Chapitre 3: Design patterns (MVC)	ملاحظة :	

- **Objectifs** أهداف: architecture MVC
- **Données** بيانات: annuaire
- **Outils** أدوات : Python

1 Partie TP

1.1 Partie 1

Exécuter le programme suivant :

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#  mvc-ann.py
# Un annuaire en format liste [] des dictionaires {}
annuaire = [
    {'prenom':'Ahmed',  'nom':'Mahdi', 'tel':'077878'},
    {'prenom':'Mohamed', 'nom':'Rabehi','tel':'06678'},
    {'prenom':'Mounir', 'nom':'Mahdi','tel':'0556644'},
    {'prenom':'Noui', 'nom':'Brahimi','tel':'067879'},
    ]
def main():
    # Ce programme permet de recherche le num dans un tableaux par son nom, et afficher
    # lire des données a partir du clavier
    print("Recherche d'un telephone")
    print("Introduire Un nom")
    nom = input()
    # nombre d'elements trouvés
    nb_found = 0
    # parcours des noms
    for personne in annuaire:
         # afficher toutes les personnes qui ont le nom donné
        if personne['nom'] == nom:
            print(nom, personne['prenom'], personne['tel'])
            nb_found += 1
    if not nb_found:
        print("ce nom %s n'existe pas "%nom)

if __name__ == '__main__':
    main()
```

Que fait ce programme ?

1.2 Partie 2

1. On veut reformuler le code précédant en séparant l'aspect affichage de l'aspect données

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Un annuaire en format liste [] des dictionaires {}
class model:
```

```
    """ classe de modele de donnée"""
    def __init__(self):
        self.annuaire = [
        {'prenom':'Ahmed',  'nom':'Mahdi', 'tel':'0778787887'},
        {'prenom':'Mohamed', 'nom':'Mahdi','tel':'0778787887'},
        {'prenom':'Mounir', 'nom':'Katibi','tel':'0778787887'},
        {'prenom':'Noui', 'nom':'Brahimi','tel':'0778787887'},
        ]
    def rechercher(self, nom):
        """ rechecher un tel par nom"""
        # La liste des personnes trouvées
        personnes  = []
        for persn in self.annuaire:
         # afficher toutes les personnes qui ont le nom donné
            if persn['nom'] == nom:
                personnes.append(persn)
        #retourner une liste de dictionnaires
        return personnes

class view:
    def __init__(self,):
        pass
    def input(self,):
        """ recuperer le nom à rechercher"""
        print("Recherche d'un telephone")
        print("Introduire Un nom")
        nom = input()
        return nom
    def output(self, personnes):
        """ afficher les informations d'une liste des personnes"""
        print("La liste des noms trouvés")
        print(" %d personnes trouvées"%len(personnes))
        for pers in personnes:
            print(pers['nom'], pers['prenom'], pers['tel'])

def controleur():
    # Ce programme permet de recherche le num dans un tableaux par son nom, et afficher
    # lire des données a partir du clavier
    data_model =  model()
    affichage = view()
    """  rechercher un nom """
    nom = affichage.input()
        personnes = data_model.rechercher(nom)
    affichage.output(personnes)

if __name__ == '__main__':
    controleur()
```

2. Quels sont les avantages et les inconvénients de cette restructuration ?

3. Modifier la classe Model pour afficher toutes les personnes dans l'annuaire.

```
    def get_all(self):
        """ rechercher un tel par nom"""
        # La liste des personnes trouvées
        personnes  = []
        for persn in self.annuaire:
         # afficher toutes les personnes
            personnes.append(persn)
        #un tableaux avec des nom des champs
        return personnes
```

4. Modifier la fonction controleur, pour qu'on puisse afficher toutes les personnes,
5. Qu'est ce qu'on doit modifier dans la classe View afin d'afficher toutes les personnes.
6. On veut maintenant créer une classe de contrôle appelée contrôleur à la place de la fonction contrôleur.

```
class controleur:
    def __init__(self,):
        """ constructeur du controleur"""
        # initialiser le model de données
        #~ self.data_model =  model()
        self.data_model =  model_fichier()
        self.affichage = view()
    def rechercher(self):
        """  rechercher un nom """
        nom = self.affichage.input()

        personnes = self.data_model.rechercher(nom)

        self.affichage.output(personnes)
if __name__ == '__main__':
    contro = controleur()
    contro.rechercher()
```

1.3 Partie 3

1. Maintenant On veut ajout une nouvelle fonctionnalité à notre système, l'ajout des nouvelles personnes:

   - ajouter une méthode d'ajout dans le modèle de données.
   - ajouter une méthode d'interface d'ajout
   - ajouter une méthode au contrôleur qui permet l'ajout.

   - Ajout dans le modèle de données.

```
    def ajouter(self, nom, prenom, tel):
        """ ajouter une personne """
        personne ={}
        personne["nom"]    = nom
        personne["prenom"] = prenom
        personne["tel"] = tel
        self.annuaire.append(personne)
```

   - Ajout dans le l'interface View

```
    def input_ajout(self,):
        """ récupérer le nom à ajouter"""
        print("Ajout d'une personne")
        print("Introduire Un nom")
        nom = input()
        print("Introduire Un prenom")
        prenom = input()
        print("Introduire Un tel")
        tel = input()
        return (nom, prenom, tel)
```

- Ajout dans la classe Controleur

```
    def ajouter(self):
        """  rechercher un nom """
        nom, prenom, tel = self.affichage.input_ajout()

        self.data_model.ajouter(nom, prenom, tel)
        personnes = self.data_model.get_all()
        self.affichage.output(personnes)
        tel = input()
        return (nom, prenom, tel)
```

2. Modifier le programme main pour ajouter une personne.
3. Modifier le programme pour ajouter plusieurs personnes.

1.4 Partie 4: MVC + Interface Graphique

1. On veut changer le modèle de données afin de stocker les informations dans un fichier. Le fichier de données et un fichier texte qui contient les champs (nom, prénom, tel) séparés par des tabulations.

   Quelle est la partie à modifier (le modèle, la vue, ou bien le contrôleur) ?

```
class model_fichier:
    """ classe de modele de donnée"""
    def __init__(self):
        self.annuaire = []
        try:
            myfile = open("annuaire.txt")
        except:
            print("Can't open DataFile")
            sys.exit()
        lines = myfile.readlines()
        for line in lines:
            line = line.strip('\n')
            fields = line.split('\t')
            personne ={}
            personne["nom"] = fields[0]
            personne["prenom"] = fields[1]
            personne["tel"] = fields[2]
            self.annuaire.append(personne)
        myfile.close()
    def __del__(self,):
        """ Destructeur de la classe Modele,
           Il est appelé à la fin de programme
```

```
            recharger les donnes dans le fichier """
        try:
            myfile = open("annuaire.txt", "w")
        except:
            print("Can't open DataFile")
            sys.exit()
        for personne in self.annuaire:
            line = personne['nom']+'\t'+ personne['prenom']+'\t'+ personne['tel']+"\n"
            myfile.write(line)
        myfile.close()
```

Les méthodes suivantes sont identiques aux méthodes de la classe « model ».

```
def rechercher(self, nom):
def get_all(self):
def ajouter(self, nom, prenom, tel):
```

2. Modifier le programme « main » pour utiliser le modèle de stockage dans un fichier.
3. Quelle est la partie à modifier si on veut stocker nos données dans une base de données.
4. Quelle est la partie à modifier si on veut créer une interface graphique pour notre programme, ou bien une interface web ?
5. Quels sont les avantages de cette méthode de conception ?

1.5 Partie 5: MVC + Interface Graphique

```
from tkinter import Tk, Label, Button, Entry, IntVar, END, W, E, Text, Listbox, OptionMenu
from tkinter import StringVar
from model_fichier import model_fichier
class view_tk:
    def __init__(self, model):
        master = Tk()
        self.master = master
        self.model = model
        master.title("Annuaire")
        self.entered_number = 0
        #~ master.withdraw()
        master.clipboard_clear()
        master.clipboard_append('i can has clipboardz?')

        self.output_label = Label(master, text="Output")

        self.label = Label(master, text="Input:")
        self.entry = Text(master, height=1, width=50)
        self.label_prenom = Label(master, text="Prenom:")
        self.entry_prenom = Text(master, height=1, width=50)
        self.label_tel = Label(master, text="Tel:")
        self.entry_tel = Text(master, height=1, width=50)
        self.output = Text(master, height=10, width=50)
        self.search_button = Button(master, text="Search", command=lambda: self.update("search"))
        self.add_button = Button(master, text="Add", command=lambda: self.update("add"))
        self.all_button = Button(master, text="Get All", command=lambda: self.update("get_all"))

        # LAYOUT
```

```
        self.label.grid(row=0, column=0, sticky=W)
        self.label_prenom.grid(row=0, column=1, sticky=W)
        self.label_tel.grid(row=0, column=2, sticky=W)
        #input nom
        self.entry.grid(row=1, column=0, columnspan=10, sticky=W+E)
        # prenom et tel
        self.entry_prenom.grid(row=1, column=1, columnspan=10, sticky=W+E)
        self.entry_tel.grid(row=1, column=2, columnspan=2, sticky=W+E)


        # command

        self.search_button.grid(row=2, column=0)
        self.add_button.grid(row=2, column=1)
        self.all_button.grid(row=2, column=2)

        # Output
        self.output_label.grid(row=3, column=0, sticky=E)

        # output
        self.output.grid(row=4, column=0, columnspan=5, sticky=W+E)

    def update(self, method):
        nom = self.entry.get("1.0",END).strip()
        prenom = self.entry_tel.get("1.0",END)
        tel = self.entry_tel.get("1.0",END)
        # clean ends of words
        nom = nom.strip()
        prenom = prenom.strip()
        tel = tel.strip()
        if method == "search":
            result = self.search(nom)
        elif method == "add":
            result = self.add(nom, prenom, tel)
        elif method == "get_all":
            result = self.get_all()
        else: # reset
            print("method", method)
            result = nom
        self.output.insert(END, str(result))

    def add(self, nom, prenom, tel):
        return self.model.ajouter(nom, prenom, tel)
    def search(self, nom):
        result = self.model.rechercher(nom)
        if result:
            return result
        else:
            return "'%s' not found"%nom
    def get_all(self):
        return self.model.get_all()

    def run(self,):
        self.master.mainloop()


def controleur3():
```

```
# Ce programme permet de recherche le num dans un tableaux par son nom, et afficher
# lire des données a partir du clavier
data_model =  model_fichier()
affichage = view_tk(data_model)
affichage.run() # lancer l'interface graphique
```

2 Travail à domicile واجب منزلي

Refaire le projet du suivi des étudiants avec l'architecture MVC.

- **Langage** : au choix : Python/Java/PHP
- **Modèle de données** : fichier XML avec Accès en utilisant l'API DOM.
- **Vue** : aux choix : Console/ Interface graphique/ web
- **Contrôleur** : réaliser les opérations de suivi des étudiants.

2.1 Questions supplémentaires (facultatif) أسئلة إضافية (اختيارية)

Utiliser le framework web Flask pour manilpuler et afficher le mushaf basé sur le fichier du Quran-juza-amma.xml جزء عمّ

استعمل إطار العمل Flask لعرض المصحف مبنيا على مستند القرآن الكريم بصيغة XML

3 Références مراجع

- Applications orientées données (Cours CNAM NSY 135) `http://orm.bdpedia.fr/mvc.html`
- Sciences Informatique et Numérique: Architecture MVC Cours `http://projet.eu.org/pedago/sin/term/5-architecture_MVC.pdf`
- Tutorial points, MVC Framework - Introduction `https://www.tutorialspoint.com/mvc_framework/`
- Model-View-Controller وجدي عصام، تصميم البرمجيات باستخدام نمط `https://informatic-ar.com/mvc_java/`
- MVC المبرمج العربي، نمط التصميم `https://arabicprogrammer.com/article/6434275856/`

4 Exercices supplémentaires تمارين للتعمق

- Découvrir les design patterns similaires au MVC: MVP, MVVM.
- Citer les meilleurs frameworks qui utilisent l'architecture MVC.
- Utiliser Flask ou bien Django pour réaliser une interface web pour le projet suivi des étudiants.

- اكتشف النماذج المشابهة لMVC : MVP, MVVM
- اذكر أفضل أطر العمل التي تستخدم معمارية MVC.
- استعمل Flask أو Django لإنجاز واجهة وب لمشروع متابعة الطلبة.