```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# fichier: moduleJulien.py
#  auteur: Julien FORTINEAU
#    date: 2025-12-11


"""
Ce fichier regroupe toutes les fonctions et classes créés par Julien FORTINEAU pour le
projet PILS en terminale NSI
"""

import modules.moduleLouis as e


def ResetBase():
    ''' Vide le fichier de tout événement'''
    with open("base-evenements.txt", "w") as f:
        f.write('id,nom,date,localisation,prix,contact')

def Sauver(d):
    """
    Cette fonction permet de sauvegarder touts les événements un par un dans le fichier
base-evenementss.txt au format
    csv avec comme séparateur une simple virgule.
    """
    ResetBase() # Vide le fichier pour maj les modifications.

    for id, event in d.items():
        id = int(id)
        with open("base-evenements.txt", "a") as f:
            f.write(f"\n{id},{event.nom},{event.place},{str(event.date)},{event.contact},
{'*' if event.prix else ''}")


def Charger():
    Data = {}
    with open("base-evenements.txt", "r") as f:
        f.readline() # retire le header
        for line in f.readlines():
            p = line.split(',')

            d = p[3].split('-')
            date = e.Date(d[1], d[2], d[0])
            Data[p[0]] = e.Evenement(p[1], p[2], date, p[4], True if "*" in p[5] else
False)
        return Data


def ModifierEvenement(listeEvent, index, nom=None, place=None, date=None, contact=None,
prix=None):
    """Remplace un événement existant par un nouveau."""
    if 0 <= index < len(listeEvent):  # Vérifie si l'index est valide
        event = listeEvent[str(index)]
        if nom is not None: event.nom = nom
        if place is not None: event.place = place
        if date is not None: event.date = date
        if contact is not None: event.contact = contact
        if prix is not None: event.prix = prix


class Recherche:
    """
    Cette classe s'occupe de chercher et filtrer les événements.
    """

    def __init__(self, data):
        self.data = data
```

```
    def Recherche(self, s: str):
        result = {}
        s = s.lower()

        for cle in self.data.keys():
            event = self.data[cle]
            if s == str(cle):
                result[cle] = event
            if s in event.nom.lower():
                result[cle] = event
            elif s in str(event.date):
                result[cle] = event
            elif s in event.place.lower():
                result[cle] = event
            elif s in event.contact.lower():
                result[cle] = event

        if s.lower() in "gratuit":
            result.update(self.FiltrerGratuit())

        if s.lower() in "payant":
            result.update(self.FiltrerPayant())

        return result


    def FiltrerGratuit(self):
        result = {}

        for cle, val in self.data.items():
            if val.prix:
                result[cle] = val

        return result


    def FiltrerPayant(self):
        result = {}

        for cle, val in self.data.items():
            if not val.prix:
                result[cle] = val

        return result


eventTemplate = """    <div class="evenement">
          <p class="titre-event">{nom}</p>
          <p class="prix-event">{prix}</p>
          <p class="date-event">{date}</p>
          <p class="lieu-event">{lieu}</p>
          <p class="contact-event">{contact}</p>
        </div>
"""


def GenererHtml():
    """ Génère le fichier guide.html à partir de template.html"""
    data = Charger()

    with open("template.html", "r") as f:  # récupère le template
        contents = f.readlines()

    for id, e in data.items():  # insère les événements au bon endroit
        if e.date.est_passee(): continue
        temp = eventTemplate.format(nom=e.nom, prix="Gratuit" if e.prix else "Payant",
date=e.date, lieu=e.place, contact=e.contact)
        contents.insert(19, temp)
```

```
    with open("guide.html", "w") as f:  # Place le code Html généré dans le fichier
guide.html
        f.writelines(contents)


def Exporter():
    GenererHtml()
```