TP de Programmation avancée en Python : Série 4 (correction)

Exercice1 :

```python
from abc import ABC, abstractmethod

# 1. Classe abstraite Paiement
class Paiement(ABC):
    @abstractmethod
    def validerPaiement(self):
        pass

    @abstractmethod
    def effectuerPaiement(self):
        pass

    @abstractmethod
    def afficherDetails(self):
        pass

# 2. Classes concrètes
class CarteCredit(Paiement):
    def __init__(self, solde, numero_carte):
        self.solde = solde
        self.numero_carte = numero_carte
        self.montant = 0

    def validerPaiement(self, montant):
        if self.solde >= montant:
            self.montant = montant
            print(f" Carte valide pour {montant} DH")
            return True
        else:
            print(" Fonds insuffisants sur la carte")
            return False
```

```
def effectuerPaiement(self):
    self.solde -= self.montant
    print(f" Paiement Carte Credit effectué : {self.montant} DH")

def afficherDetails(self):
    print(f"Carte Credit-Numéro :{self.numero_carte},Montant payé:{self.montant} DH")
```

```
class PayPal(Paiement):
    def __init__(self, solde, email):
        self.solde = solde
        self.email = email
        self.montant = 0

    def validerPaiement(self, montant):
        if self.solde >= montant:
            self.montant = montant
            print(f" Compte PayPal valide pour {montant} DH")
            return True
        else:
            print(" Fonds insuffisants sur PayPal")
            return False

    def effectuerPaiement(self):
        self.solde -= self.montant
        print(f" Paiement PayPal effectué : {self.montant} DH")

    def afficherDetails(self):
        print(f"PayPal - Email : {self.email}, Montant payé : {self.montant} DH")
```

```
class VirementBancaire(Paiement):
    def __init__(self, solde, banque):
        self.solde = solde
        self.banque = banque
        self.montant = 0

    def validerPaiement(self, montant):
        if self.solde >= montant:
            self.montant = montant
            print(f" Solde suffisant pour virement {montant} DH")
            return True
        else:
            print(" Solde insuffisant pour virement")
            return False

    def effectuerPaiement(self):
        self.solde -= self.montant
        print(f" Virement bancaire effectué : {self.montant} DH")

    def afficherDetails(self):
        print(f"Virement Bancaire - Banque : {self.banque}, Montant payé : {self.montant} DH")
```

```python
# 3. Classe SystèmePaiement
class SystemePaiement:
    def __init__(self, methodes):
        self.methodes = methodes

    def simulerPaiements(self, montant):
        for methode in self.methodes:
            print("\n--- Nouvelle tentative de paiement ---")
            if methode.validerPaiement(montant):
                methode.effectuerPaiement()
            methode.afficherDetails()

#4. Programme principal
if __name__ == "__main__":
    methodes = [
        CarteCredit(solde=2000, numero_carte="1234-5678-9876-5432"),
        PayPal(solde=800, email="client@example.com"),
        VirementBancaire(solde=1500, banque="Banque Centrale")
    ]

    systeme = SystemePaiement(methodes)
    montant_a_payer = 600
    systeme.simulerPaiements(montant_a_payer)
```

Exercice 2 : Gestion des fichiers et répertoires avec **pathlib**

```python
from pathlib import Path

# 1-Chemin principal
base = Path("MesProjets")
# mkdir(exist_ok=True) permet d'éviter une erreur si le dossier existe déjà
base.mkdir(exist_ok=True)
(projetA := base / "ProjetA").mkdir(exist_ok=True)
(projetB := base / "ProjetB").mkdir(exist_ok=True)
(projetC := base / "ProjetC").mkdir(exist_ok=True)
'''projet = Path("MesProjets/projet_A")
projet.mkdir()'''
```

```
# ==========================================
# 2- Création de fichiers dans ProjetA
# ==========================================

# Chemins des fichiers
notes = projetA / "notes.txt"
todo = projetA / "todo.txt"
# Écriture dans notes.txt
notes.write_text("Ligne1:Notesduprojet\nLigne2:Idées\nLigne3:Informationsdiverses\n",\
                 encoding="utf-8")

# Écriture dans todo.txt
todo.write_text("Tâche 1:Faire A\nTâche 2:Faire B\nTâche 3 :Faire C\n", encoding="utf-8")

print("Fichiers créés et remplis dans ProjetA.")
```

```
# ==========================================
# 3- Lecture et analyse du fichier notes.txt
# ==========================================

# Lecture du fichier
contenu = notes.read_text(encoding="utf-8")

# Comptage des lignes
nb_lignes = len(contenu.splitlines())
print(f"Le fichier notes.txt contient {nb_lignes} lignes")
```

```
# 4- Lister les fichiers TXT de ProjetA
# ==========================================

print("\nFichiers .txt dans ProjetA :")

for fichier in projetA.iterdir():  # Parcours direct du dossier
    if fichier.suffix == ".txt":   # Vérification de l'extension
        print("→", fichier.name)
```

```python
# ============================================
# 5- Déplacement et renommage
# ============================================

# Créer un fichier dans ProjetB
ancien = projetB / "ancien.txt"
ancien.write_text("Ancien fichier", encoding="utf-8")

# Nouveau chemin dans ProjetC + nouveau nom
nouveau = projetC / "archive.txt"

# Déplacement + renommage
ancien.rename(nouveau)

print("\nancien.txt déplacé dans ProjetC et renommé en archive.txt.")

# ============================================
# 6- Suppression
# ============================================

# Suppression de todo.txt
if todo.exists():
    todo.unlink()  # unlink() supprime un fichier
    print("todo.txt supprimé.")
# Tentative de suppression du dossier ProjetC
try:
    projetC.rmdir()  # rmdir() ne supprime que si dossier vide
    print("ProjetC supprimé car il était vide.")
except OSError:
    print("Impossible de supprimer ProjetC : le dossier n’est pas vide.")
```

Exercice 3 :

```
import numpy as np
import matplotlib.pyplot as plt

Tf = 5
dt = 0.1
N = int(Tf/dt) + 1   # nombre de points

ti = np.linspace(0, Tf, N)
xi = np.zeros(N)

xi[0] = 1
x = 1

for i in range(1, N):
    x += dt * 2 * x * (1 - x / 4)
    xi[i] = x

# Représentation graphique
fig, ax = plt.subplots()
ax.plot(ti, xi, color='red', linestyle='dashed', linewidth=2)
ax.set_title("Solution approchée (Méthode d'Euler)")
ax.grid()
plt.show()
```