#QCM:  
  
# 1-a  
# 2-b  
# 3-c  
# 4-a  
# 5-b  
# 6-a  
  
  
  
#PARTI1:  
from abc import ABC, abstractmethod  
  
class Personne(ABC):  
 def \_\_init\_\_(self, nom\_complet, CIN, age):  
 self.nom\_complet = nom\_complet  
 self.CIN = CIN  
 self.age = age  
  
 @abstractmethod  
 def Afficher\_detail(self):  
 pass  
  
 @staticmethod  
 def liste\_premiers(a, b):  
 premiers = []  
 for n in range(a, b+1):  
 if n > 1:  
 for i in range(2, n):  
 if (n % i) == 0:  
 break  
 else:  
 premiers.append(n)  
 return premiers  
  
#PARTIE2:  
class Client(Personne):  
 clients = 0  
  
 def \_\_init\_\_(self, nom\_complet, CIN, age, id\_client, budget):  
 super().\_\_init\_\_(nom\_complet, CIN, age)  
 self.id\_client = id\_client  
 self.budget = budget  
 Client.clients += 1  
  
 @classmethod  
 def afficher\_clients(cls):  
 print("Nombre total de clients :", cls.clients)  
  
 def \_\_add\_\_(self, other):  
 return self.budget + other.budget  
  
 @staticmethod  
 def print\_star(n):  
 for i in range(n, 0, -1):  
 print('\*' \* i)  
  
  
#PARTIE3:  
class Vendeur:  
 nb\_vendeurs = 0  
  
 def \_\_init\_\_(self, nom\_complet, CIN, age, id\_vendeur, salaire):  
 self.nom\_complet = nom\_complet  
 self.CIN = CIN  
 self.age = age  
 self.\_id\_vendeur = id\_vendeur  
 self.\_\_salaire = salaire  
 Vendeur.nb\_vendeurs += 1  
  
 @classmethod  
 def afficher\_nombre\_total(cls):  
 print("Nombre total de vendeurs :", cls.nb\_vendeurs)  
  
 def get\_id\_vendeur(self):  
 return self.\_id\_vendeur  
  
 def set\_id\_vendeur(self, value):  
 self.\_id\_vendeur = value  
  
  
 def get\_salaire(self):  
 return self.\_\_salaire  
  
 def set\_salaire(self, value):  
 if value >= 0:  
 self.\_\_salaire = value  
 else:  
 print("Salaire ne peut pas être négatif.")  
  
 def Afficher\_detail(self):  
 print(f"Nom complet : {self.nom\_complet}")  
 print(f"CIN : {self.CIN}")  
 print(f"Âge : {self.age}")  
 print(f"ID vendeur : {self.get\_id\_vendeur()}")  
 print(f"Salaire : {self.get\_salaire()}")  
  
#PARTIE4:  
class Produit:  
 nb\_produits = 0  
  
 def \_\_init\_\_(self, nom, prix):  
 self.nom = nom  
 self.prix = prix  
 Produit.nb\_produits += 1  
  
 def Afficher\_detail(self):  
 print(f"Nom : {self.nom}")  
 print(f"Prix : {self.prix}")  
  
 @classmethod  
 def afficher\_nombre\_total(cls):  
 print("Nombre total de produits :", cls.nb\_produits)  
  
  
 @staticmethod  
 def sort\_list(liste\_produits):  
 # Tri et suppression des doublons sans utiliser les méthodes prédéfinies  
 return sorted(set(liste\_produits), key=lambda x: x.prix)  
  
  
#PARTIE5 :  
class Vente:  
 nb\_ventes = 0  
  
 def \_\_init\_\_(self, id\_vente, date\_vente, vendeur, client, produits, prix\_total\_vente=0):  
 self.id\_vente = id\_vente  
 self.date\_vente = date\_vente  
 self.vendeur = vendeur  
 self.client = client  
 self.produits = produits  
 self.prix\_total\_vente = prix\_total\_vente  
  
 def ajouter\_produit(self, produit):  
 self.produits.append(produit)  
 self.prix\_total\_vente += produit.prix  
  
  
 def afficher\_details(self):  
 print(f"ID vente : {self.id\_vente}")  
 print(f"Date vente : {self.date\_vente}")  
 print(f"Vendeur : {self.vendeur.nom\_complet}")  
 print(f"Client : {self.client.nom\_complet}")  
 print("Produits:")  
 for produit in self.produits:  
 produit.Afficher\_detail()  
 print(f"Prix total de la vente : {self.prix\_total\_vente}")  
  
#PARTIE TEST :  
  
vendeur1 = Vendeur("Vendeur 1", "CIN-1", 30, 1234, 2000)  
client1 = Client("Client 1", "CIN-2", 25, 5678, 5000)  
produit1 = Produit("Produit 1", 100)  
produit2 = Produit("Produit 23", 200)  
produit3 = Produit("Produit 9", 100)  
print(produit1 == produit3) # True  
vente1 = Vente(1, "2022-01-01", vendeur1, client1, [produit1, produit2])  
vente1.ajouter\_produit(produit3)  
vente1.afficher\_details()  
Produit.afficher\_nombre\_total()  
Vendeur.afficher\_nombre\_total()