DS Informatique N°1 : Corrigé

```python
# Partie 1 : Classe Polynome
class Polynome:
        PRECISION = 3
        def __init__(self, coeffs):
                if not coeffs or not all(isinstance(c, (int, float)) for c in coeffs):
                        raise ValueError("Tous les coefficients doivent être des nombres réels")
                self.coeffs = coeffs

    def setCoeffs(self, coeffs):
                if not isinstance(coeffs, list) or len(coeffs) == 0:
                        raise ValueError("La liste des coefficients doit être non vide.")
                for c in coeffs:
                        if not isinstance(c, (int, float)):
                                raise ValueError("Tous les coefficients doivent être des nombres réels.")
                self.coeffs = coeffs

        def __call__(self, t):
                return sum(c * t**i for i, c in enumerate(self.coeffs))

        def derivee(self):
                deriv_coeffs = [i*c for i, c in enumerate(self.coeffs)][1:]
                return Polynome(deriv_coeffs)

        def __repr__(self):
                expr = ""
                for i, c in enumerate(self.coeffs):
                        if abs(c) < 1e-12:
                                continue # ignorer les coefficients nuls
                        term = f"{abs(c):.{self.PRECISION}f}"
                        if i == 1:
                                term += "*t"
                        elif i > 1:
                                term += f"*t^{i}"
                        sign = "-" if c < 0 else "+"
                        expr += f" {sign} {term}" if expr else (f"-{term}" if c < 0 else term)
                return "C(t) = " + (expr if expr else "0")


        def __eq__(self, other):
                return isinstance(other, Polynome) and self.coeffs == other.coeffs

# Partie 2 : Classe Appareil
class Appareil:
        def __init__(self, nom, coeffs):
                self.nom = nom
                self.conso = Polynome(coeffs)
                self.conso_inst = self.conso.derivee()

        def evalConsommation(self, t):
                return (self.conso(t), self.conso_inst(t))

        def setCoeffs(self, coeffs):
                self.conso.setCoeffs(coeffs)
                self.conso_inst = self.conso.derivee()

        def __str__(self):
```

```
                return (f"Appareil {self.nom} :\n"
                                f"Consommation cumulée : {self.conso}\n"
                                f"Consommation instantanée : {self.conso_inst}")

        def __eq__(self, other):
                return isinstance(other, Appareil) and self.conso == other.conso

# Partie 3 : Classe GestionnaireAppareils
class GestionnaireAppareils:
        def __init__(self, Dapp):
                self.LA = [Appareil(nom, coeffs) for nom, coeffs in Dapp.items()]

        def __len__(self):
                return len(self.LA)

        def __getitem__(self, nom):
                for app in self.LA:
                        if app.nom == nom:
                                return app
                raise KeyError(f"Appareil {nom} non trouvé")

        def __contains__(self, appareil):
                return any(app == appareil or app.nom == appareil for app in self.LA)

        def __add__(self, other):
                if not isinstance(other, (GestionnaireAppareils,Appareil)):
                        raise TypeError("Objet non supporté")
                if isinstance(other, GestionnaireAppareils):
                        for app in other.LA:
                                if app not in self:
                                        self.LA.append(app)
                else: # other est un Appareil
                        if other not in self:
                                self.LA.append(other)
                return self

        def supprimerAppareil(self, nom):
                self.LA = [app for app in self.LA if app.nom != nom]

        def __sub__(self, other):
                if isinstance(other, GestionnaireAppareils):
                        for app in other.LA:
                                self.supprimerAppareil(app.nom)
                elif isinstance(other, str):
                        self.supprimerAppareil(other)
                else:
                        raise TypeError("Objet non supporté")
                return self

        def getConsoMax(self, seuil, t):
                return [app.nom for app in self.LA if app.evalConsommation(t)[1] > seuil]

        def getConsoTotale(self, t):
                return sum(app.evalConsommation(t)[0] for app in self.LA)

        def __str__(self):
                s = "Gestionnaire d'appareils :\n"
```

```
                for app in self.LA:
                        s += "- " + str(app) + "\n"
                return s

# Partie 4 : Lecture fichier et script principal
def charger_appareils(fichier):
        D = {}
        with open(fichier, "r") as f:
                for ligne in f:
                        parts = ligne.strip().split(";")
                        nom = parts[0].strip()
                        coeffs = [float(c) for c in parts[1:]]
                        D[nom] = coeffs
        return D

# Exemple d'utilisation
D = charger_appareils("appareils.txt")
G = GestionnaireAppareils(D)

nom_app = input("Nom de l'appareil : ")
t = float(input("Instant t : "))
seuil = float(input("Seuil : "))

if nom_app in G:
        cumul, inst = G[nom_app].evalConsommation(t)
        print(f"Consommation cumulée: {cumul}, instantanée: {inst}")

print("Appareils avec conso instantanée > seuil : ", G.getConsoMax(seuil, t))

# Trier par consommation cumulée décroissante
tri = sorted(G.LA, key=lambda a: a.evalConsommation(t)[0], reverse=True)
print("Appareils triés par conso cumulée décroissante :")
for app in tri:
        print(app.nom, app.evalConsommation(t)[0])
```