TD N° 2 : POO

Exercice 1 : Détection de clients à risque

Une entreprise souhaite analyser ses clients afin de détecter les profils a risque.

Chaque **commande** est caractérisée par: un montant, un statut (PAID, PENDING), un nombre de jours de retard.
Chaque **client** est caractérisé par : un nom, un secteur, un tableau de commandes [].

Un client est considéré à risque si au moins une des conditions suivantes est vérifiée :

- le montant total des commandes en retard dépasse 20000
- le nombre de commandes en attente (PENDING) est supérieur ou égal a 3
- le montant paye est inferieur a 50% du montant total commande

Travail demandé :

1. Implémenter une fonction constructeur order
2. Implémenter une fonction constructeur customer
3. Implémenter les méthodes suivantes :
   - getTotalCommande()
   - getTotalPaye()
   - countEnAttente()
   - getMontantRetard()
   - isRisque()
4. Créer un jeu de données avec plusieurs clients
5. Afficher les clients a risque

========== CORRECTION ==========

```
<script type="text/javascript">

function getTotalCommande() {
    var s = 0;
    for (var i = 0; i < this.commandes.length; i++) {
         s = s  + this.commandes[i].montant;
    }
    return s;
}

function getTotalPaye() {
    var s = 0;
    for (var i = 0; i < this.commandes.length; i++) {
```

```
    if(this.commandes[i].statut ==='PAID')
         s = s  + this.commandes[i].montant;
    }
    return s;
}

function countEnAttente(){
    var c = 0;
    for (var i = 0; i < this.commandes.length; i++) {
         if(this.commandes[i].statut ==='PENDING')
             c = c + 1;
    }
    return c;
}

function getMontantRetard() {
    var s = 0;
    for (var i = 0; i < this.commandes.length; i++) {
         if(this.commandes[i].joursRetard > 0)
             s = s  + this.commandes[i].montant;
    }
    return s;
}

function isRisque(){
    var montantRetard = this.getMontantRetard();
    var nombreRetard = this.countEnAttente();
    var totalCommande = this.getTotalCommande();
    var totalPaye = this.getTotalPaye();
    if(montantRetard>20000 || nombreRetard>= 3 || totalPaye  < 0.5*totalCommande)
           return true;

    return false;
}

function afficherCustomer(){
    document.write('Nom : '+ this.nom + '<br>');
    document.write('Secteur : '+ this.secteur + '<br>');
    document.write('Total Commandes : '+ this.getTotalCommande() + '<br>');
    document.write('Risque : '+ this.isRisque() + '<br>');
}


function Order(montant, statut, joursRetard){

    // les attributs
    this.montant = montant;
    this.statut = statut;
```

```
    this.joursRetard = joursRetard;

}

function Customer(nom, secteur, commandes){

    // les attributs
    this.nom = nom;
    this.secteur = secteur;
    this.commandes = commandes;

    // les méthodes
    this.getTotalCommande = getTotalCommande;
    this.getTotalPaye = getTotalPaye;
    this.countEnAttente = countEnAttente;
    this.getMontantRetard = getMontantRetard;
    this.isRisque = isRisque;
    this.afficherCustomer = afficherCustomer;
}


// Jeu de données
var commandes1 = [
    new Order(1000, 'PAID', 0),
    new Order(2000, 'PENDING', 5)
];
var c1 = new Customer('Ahmed', 'IT', commandes1)

var commandes2 = [
    new Order(800, 'PAID', 2)
];
var c2 = new Customer('Asmae', 'Commerce', commandes2);

var clients = [c1, c2];

for (var i = 0; i < clients.length; i++) {
    document.write('<br>Client num ' + (i+1) + ' =====   <br>');
    clients[i].afficherCustomer()
}

</script>
```

Exercice 2 : Contrôle de factures fournisseurs

Une entreprise souhaite automatiser le contrôle de ses factures fournisseurs.

Chaque **ligne de facture** contient : une quantité, un prix unitaire, une catégorie (PRODUCT, SERVICE)
Chaque **facture** contient : un numéro, un tableau de lignes [], un taux de TVA et un statut

Règles de gestion :

- une facture est invalide si une ligne contient une quantité <= 0 ou un prix < 0
- une facture est suspecte si :
  - son total HT dépasse 100000
  - ou elle contient plus de 5 lignes de type SERVICE
- une facture est bloquée si son statut est DRAFT ou CANCELLED

Travail demandé :

1. Implémenter une fonction constructeur invoiceLine
2. Implémenter une fonction constructeur invoice
3. Implémenter les méthodes suivantes :
   - getTotalHT()
   - compterParCategorie(categorie)
   - isInvalide()
   - isSuspecte()
   - isBloquee()
4. Afficher les factures suspectes

========== CORRECTION ==========

```
<script type="text/javascript">

function getTotalLigne() {
    return this.quantite * this.prixUnitaire;
}

function getTotalHT() {
    var i = 0;
    var total = 0;
    for (i = 0; i < this.lignes.length; i++) {
        total = total + this.lignes[i].getTotalLigne();
    }
    return total;
}

function countByCategorie(categorie) {
    var i = 0;
    var nombre = 0;
    for (i = 0; i < this.lignes.length; i++) {
        if (this.lignes[i].categorie == categorie) {
```

```
            nombre = nombre + 1;
        }
    }
    return nombre;
}

function isInvalide() {
    var i = 0;
    for (i = 0; i < this.lignes.length; i++) {
        if (this.lignes[i].quantite <= 0 || this.lignes[i].prixUnitaire < 0) {
            return true;
        }
    }
    return false;
}

function isSuspecte() {
    if (this.getTotalHT() > 100000) {
        return true;
    }
    if (this.countByCategorie("SERVICE") > 5) {
        return true;
    }
    return false;
}

function isBloquee() {
    if (this.statut == "DRAFT" || this.statut == "CANCELLED") {
        return true;
    }
    return false;
}

function afficherFacture() {
    document.write("<h2>Exercice 2 - Facture</h2>");
    document.write("Numéro : " + this.numero + "<br>");
    document.write("Total HT : " + this.getTotalHT() + "<br>");
    document.write("Nombre de lignes SERVICE : " +
this.countByCategorie("SERVICE") + "<br>");
    document.write("Facture invalide : " + this.isInvalide() + "<br>");
    document.write("Facture suspecte : " + this.isSuspecte() + "<br>");
    document.write("Facture bloquée : " + this.isBloquee() + "<br><br>");
}

function ligneFacture(quantite, prixUnitaire, categorie) {
    // les attributs
    this.quantite = quantite;
    this.prixUnitaire = prixUnitaire;
```

```
    this.categorie = categorie;

    // les méthodes
    this.getTotalLigne = getTotalLigne;
}

function facture(numero, lignes, tauxTVA, statut) {

    // les attributs
    this.numero = numero;
    this.lignes = lignes;
    this.tauxTVA = tauxTVA;
    this.statut = statut;

    // les méthodes
    this.getTotalHT = getTotalHT;
    this.countByCategorie = countByCategorie;
    this.isInvalide = isInvalide;
    this.isSuspecte = isSuspecte;
    this.isBloquee = isBloquee;
    this.afficherFacture = afficherFacture;
}

/*************** JEU DE DONNÉES ***************/
var lignesFacture1 = [
    new ligneFacture(10, 3000, "PRODUCT"),
    new ligneFacture(5, 9000, "SERVICE")
];
var lignesFacture2 = [
    new ligneFacture(1, 10000, "SERVICE"),
    new ligneFacture(1, 12000, "SERVICE"),
    new ligneFacture(1, 13000, "SERVICE"),
    new ligneFacture(1, 14000, "SERVICE"),
    new ligneFacture(1, 15000, "SERVICE"),
    new ligneFacture(1, 16000, "SERVICE")
];
var lignesFacture3 = [
    new ligneFacture(0, 5000, "PRODUCT"),
    new ligneFacture(3, 1000, "SERVICE")
];
var facture1 = new facture("FAC-001", lignesFacture1, 20, "VALIDATED");
var facture2 = new facture("FAC-002", lignesFacture2, 20, "VALIDATED");
var facture3 = new facture("FAC-003", lignesFacture3, 20, "DRAFT");
var factures = [facture1, facture2, facture3];
    for (i = 0; i < factures.length; i++) {
    if (factures[i].isSuspecte()) {
        factures[i].afficherFacture();
    }
```

```
}

</script>
```

Exercice 3 : Évaluation de performance des employés

Une entreprise souhaite évaluer ses employés selon leurs objectifs.
Chaque objectif contient : un code, un poids, un score et un statut
Chaque employé contient : un nom, un département, un nombre de jours d'absence et un tableau d'objectifs []

Règles :

- calcul du score pondère
- pénalité de 15% si absences > 10
- classification :
  - Excellent si score ≥ 85
  - Bon si score ≥ 70
  - Moyen si score ≥ 50
  - Faible sinon

Travail demandé :

1. Implémenter les objets objective et employée
2. Implémenter les méthodes :
   - calculerScore()
   - appliquerPenalite()
   - getMention()
3. Déterminer le meilleur employé

========== CORRECTION ==========

```
class Objectif {
    constructor(code, poids, score, statut) {
        this.code = code;
        this.poids = poids;
        this.score = score;
        this.statut = statut;
    }
}

class Employe {

    constructor(nom, departement, absences, objectifs) {
        this.nom = nom;
        this.departement = departement;
        this.absences = absences;
        this.objectifs = objectifs;
    }
```

```
    calculerScore() {
        let totalPondere = 0;
        let totalPoids = 0;
        for (let i = 0; i < this.objectifs.length; i++) {
            if (this.objectifs[i].statut === "VALID") {
                totalPondere = totalPondere + (this.objectifs[i].score *
                this.objectifs[i].poids);
                totalPoids = totalPoids + this.objectifs[i].poids;
            }
        }
        if (totalPoids === 0) {
            return 0;
        }
        return totalPondere / totalPoids;
    }

    appliquerPenalite() {
        let scoreFinal = this.calculerScore();
        if (this.absences > 10) {
            scoreFinal = scoreFinal - (scoreFinal * 15 / 100);
        }
        return scoreFinal;
    }

    getMention() {
        let scoreFinal = this.appliquerPenalite();
        if (scoreFinal >= 85) {
            return "Excellent";
        } else if (scoreFinal >= 70) {
            return "Bon";
        } else if (scoreFinal >= 50) {
            return "Moyen";
        } else {
            return "Faible";
        }
    }

    afficherEmploye() {
        document.write("<h2>Exercice 3 - Employé</h2>");
        document.write("Nom : " + this.nom + "<br>");

        document.write("Département : " + this.departement + "<br>");
        document.write("Absences : " + this.absences + "<br>");
        document.write("Score final : " + this.appliquerPenalite() + "<br>");
        document.write("Mention : " + this.getMention() + "<br><br>");
    }
}
```

```
/*************** DONNÉES EXERCICE 3 ***************/
const objectifs1 = [
new Objectif("OBJ1", 3, 90, "VALID"),
new Objectif("OBJ2", 2, 80, "VALID"),
new Objectif("OBJ3", 5, 70, "VALID")
];
const objectifs2 = [
new Objectif("OBJ1", 4, 95, "VALID"),
new Objectif("OBJ2", 3, 90, "VALID"),
new Objectif("OBJ3", 3, 85, "VALID")
];
const objectifs3 = [
new Objectif("OBJ1", 3, 60, "VALID"),
new Objectif("OBJ2", 3, 55, "VALID"),
new Objectif("OBJ3", 4, 50, "VALID")
];
const employe1 = new Employe("Sara", "Finance", 5, objectifs1);
const employe2 = new Employe("Youssef", "IT", 12, objectifs2);
const employe3 = new Employe("Amine", "RH", 3, objectifs3);

const employes = [employe1, employe2, employe3];
let meilleurEmploye = employes[0];
for (let i = 0; i < employes.length; i++) {
    employes[i].afficherEmploye();
    if (employes[i].appliquerPenalite() > meilleurEmploye.appliquerPenalite()) {
        meilleurEmploye = employes[i];
    }
}
document.write("Meilleur employé : " + meilleurEmploye.nom + " (" +
meilleurEmploye.appliquerPenalite() + ")<br><br>");
```

Exercice 4 : Analyse de stock multi-produits

Une entreprise logistique gère un stock de produits dans un entrepôt.
Chaque **produit** contient : un code, une quantité, un seuil minimal et un prix unitaire

Règles :

- produit **à risque** si quantité < seuil
- produit **surstocké** si quantité ≥ seuil * 3

Travail demandé

1. Implementer les objets necessaires
2. Implementer les methodes :
    - getValeurStock()
    - compterRisque()

- compterSurstock()

3. Calculer la valeur totale du stock
4. Trier les produits selon le niveau de risque.

========== CORRECTION ==========

```
<script type="text/javascript">

class Produit {

    constructor(code, quantite, seuilMinimal, prixUnitaire) {
        this.code = code;
        this.quantite = quantite;
        this.seuilMinimal = seuilMinimal;
        this.prixUnitaire = prixUnitaire;
    }

    getValeurStock() {
        return this.quantite * this.prixUnitaire;
    }

    isRisqueStock() {
        return this.quantite < this.seuilMinimal;
    }

    isSurstock() {
        return this.quantite >= this.seuilMinimal * 3;
    }

}

class Entrepot {

    constructor(code, produits) {
        this.code = code;
        this.produits = produits;
    }

    getValeurTotale() {
        let total = 0;
        for (let i = 0; i < this.produits.length; i++) {
            total = total + this.produits[i].getValeurStock();
        }
        return total;
    }

    countRisque() {
        let nombre = 0;
```

```
        for (let i = 0; i < this.produits.length; i++) {
            if (this.produits[i].isRisqueStock()) {
                nombre = nombre + 1;
            }
        }
        return nombre;
    }

    countSurstock() {
        let nombre = 0;
        for (let i = 0; i < this.produits.length; i++) {
            if (this.produits[i].isSurstock()) {
                nombre = nombre + 1;
            }
        }
        return nombre;
    }

    afficherEntrepot() {
        document.write("<h2>Exercice 4 - Entrepôt</h2>");
        document.write("Code : " + this.code + "<br>");
        document.write("Valeur totale du stock : " + this.getValeurTotale() +
"<br>");
        document.write("Nombre de produits à risque : " + this.countRisque() +
"<br>");
        document.write("Nombre de produits en surstock : " + this.countSurstock()
+
        "<br><br>");
    }
}

function trierEntrepotsParRisque(entrepots) {
    let temp = null;
    for (let i = 0; i < entrepots.length - 1; i++) {
        for (let j = i + 1; j < entrepots.length; j++) {
            if (entrepots[j].countRisque() > entrepots[i].countRisque()) {
                temp = entrepots[i];
                entrepots[i] = entrepots[j];
                entrepots[j] = temp;
            }
        }
    }
}

/*************** DONNÉES EXERCICE 4 ***************/
const produit1 = new Produit("P001", 5, 10, 100);
const produit2 = new Produit("P002", 40, 10, 50);
const produit3 = new Produit("P003", 8, 15, 80);
```

```
const produit4 = new Produit("P004", 2, 10, 200);

const produit5 = new Produit("P005", 35, 10, 70);
const produit6 = new Produit("P006", 50, 20, 90);
const produit7 = new Produit("P007", 1, 5, 500);
const produit8 = new Produit("P008", 60, 15, 20);
const entrepot1 = new Entrepot("ENT-CASA", [produit1, produit2, produit3]);
const entrepot2 = new Entrepot("ENT-RABAT", [produit4, produit5, produit6]);
const entrepot3 = new Entrepot("ENT-TANGER", [produit7, produit8]);
const entrepots = [entrepot1, entrepot2, entrepot3];

trierEntrepotsParRisque(entrepots);

for (let i = 0; i < entrepots.length; i++) {
    entrepots[i].afficherEntrepot();
}

</script>
```

Exercice 5 : Détection de fraude assurance

Une compagnie d'assurance souhaite detecter les demandes frauduleuses.

Chaque **justificatif** contient: un type, un indicateur de validation (true/false) et un score de confiance
Chaque **demande** contient: un identifiant, un assure, un montant, un type d'acte, un niveau d'urgence, un tableau de justificatifs []

Règles de fraude :

- montant > 15000 sans justificatif valide
- plus de 50% des justificatifs ont un score < 40
- acte = SURGERY, urgence = LOW et montant > 10000

Travail demandé

1. Implementer les objets proof et claim
2. Implementer les methodes :
   - compterJustificatifsValides()
   - compterScoresFaibles()
   - estFraude()
3. Identifier la demande la plus suspecte

========== CORRECTION ==========

```
<script type="text/javascript">

class Justificatif {
    constructor(type, valide, scoreConfiance) {
```

```
        this.type = type;
        this.valide = valide;
        this.scoreConfiance = scoreConfiance;
    }
}

class Demande {
    constructor(identifiant, assure, montant, typeActe, niveauUrgence,
justificatifs) {
        this.identifiant = identifiant;
        this.assure = assure;
        this.montant = montant;
        this.typeActe = typeActe;
        this.niveauUrgence = niveauUrgence;
        this.justificatifs = justificatifs;
    }

    countJustificatifsValides() {
        let nombre = 0;
        for (let i = 0; i < this.justificatifs.length; i++) {
            if (this.justificatifs[i].valide === true) {
                nombre = nombre + 1;
            }
        }
        return nombre;
    }

    countScoresFaibles() {
        let nombre = 0;
        for (let i = 0; i < this.justificatifs.length; i++) {
            if (this.justificatifs[i].scoreConfiance < 40) {
                nombre = nombre + 1;
            }
        }
        return nombre;
    }

    isFraude() {
        const nombreValides = this.countJustificatifsValides();
        const nombreFaibles = this.countScoresFaibles();
        if (this.montant > 15000 && nombreValides === 0) {
            return true;
        }
        if (this.justificatifs.length > 0 && nombreFaibles >
this.justificatifs.length / 2) {
            return true;
        }
```

```
        if (this.typeActe === "SURGERY" && this.niveauUrgence === "LOW" &&
this.montant >
        10000) {
            return true;
        }
        return false;
    }

    getScoreSuspicion() {
        let score = 0;
        if (this.montant > 15000 && this.countJustificatifsValides() === 0) {
            score = score + 50;
        }
        if (this.justificatifs.length > 0 && this.countScoresFaibles() >
        this.justificatifs.length / 2) {
            score = score + 30;
        }
        if (this.typeActe === "SURGERY" && this.niveauUrgence === "LOW" &&
this.montant >
        10000) {
            score = score + 20;
        }
        return score;
    }

    afficherDemande() {
        document.write("<h2>Exercice 5 - Demande</h2>");
        document.write("Identifiant : " + this.identifiant + "<br>");
        document.write("Assuré : " + this.assure + "<br>");
        document.write("Montant : " + this.montant + "<br>");
        document.write("Fraude détectée : " + this.isFraude() + "<br>");
        document.write("Score de suspicion : " + this.getScoreSuspicion() +
"<br><br>");
    }
}
/*************** DONNÉES EXERCICE 5 ***************/
const justificatifs1 = [
    new Justificatif("ORDONNANCE", false, 20),
    new Justificatif("FACTURE", false, 30)
];
const justificatifs2 = [
    new Justificatif("RAPPORT", true, 80),
    new Justificatif("FACTURE", true, 90)
];
const justificatifs3 = [
    new Justificatif("SCANNER", false, 35),
    new Justificatif("FACTURE", false, 25),
    new Justificatif("ORDONNANCE", false, 10)
```

```
];
const demande1 = new Demande("DEM-001", "Ali", 17000, "CHECKUP", "HIGH",
justificatifs1);
const demande2 = new Demande("DEM-002", "Sara", 8000, "CONSULTATION", "LOW",
justificatifs2);
const demande3 = new Demande("DEM-003", "Youssef", 12000, "SURGERY", "LOW",
justificatifs3);
const demandes = [demande1, demande2, demande3];
let demandePlusSuspecte = demandes[0];

for (let i = 0; i < demandes.length; i++) {
    demandes[i].afficherDemande();
    if (demandes[i].getScoreSuspicion() > demandePlusSuspecte.getScoreSuspicion())
{
        demandePlusSuspecte = demandes[i];
    }
}
document.write("Demande la plus suspecte : " + demandePlusSuspecte.identifiant +
"<br>");

</script>
```