# DOCUMENTATION TECHNIQUE

## Projet Java — Initiation à la programmation orientée objet

**Formation :** BTS SIO option SLAM — 1ère année

**Auteur :** Midoux Noé

**Environnement :** Java (Eclipse IDE) — TP2 & TP4

**Objectif :** Apprendre les bases du langage Java : variables, types, entrées/sorties, structures de contrôle

# 1. Présentation générale du projet

Ce projet est un ensemble de travaux pratiques réalisés en première année de BTS SIO option SLAM. L'objectif principal est éducatif : il s'agit de découvrir et maîtriser les fondamentaux du langage Java, un langage orienté objet très répandu dans le monde professionnel.

Le projet est structuré en deux TP progressifs :

- TP2 : introduction aux variables, types de données et opérations de base
- TP4 : introduction aux structures de contrôle (boucles et conditions)

## 1.1 Environnement de développement

| Élément | Détail |
|---|---|
| Langage | Java (JDK 8+) |
| IDE | Eclipse (Java EE / standard) |
| Système de build | Compilation Eclipse (bouton Run ou F5) |
| Structure du projet | Dossier src/ pour les sources, bin/ pour les .class compilés |
| Contrôle de version | Non utilisé (archive ZIP manuelle) |

## 1.2 Structure de l'archive

L'archive ZIP contient deux projets Eclipse distincts :

| Dossier | Contenu |
|---|---|
| TP2/src/ | 5 fichiers .java (exercices du TP2) |
| TP2/bin/ | Fichiers .class compilés correspondants |
| TP2/.classpath / .project | Fichiers de configuration Eclipse |
| TP4/src/ | 10 fichiers .java (exercices du TP4 + compléments) |
| TP4/bin/ | Fichiers .class compilés correspondants |

# 2. TP2 — Variables, calculs et affichage

Le TP2 introduit les notions de base de Java : déclaration de variables, saisie au clavier avec la classe Scanner, calculs arithmétiques et affichage des résultats. Chaque fichier est un programme indépendant avec sa propre classe principale.

## 2.1 Concepts Java utilisés

- Déclaration et initialisation de variables (int, float, String)
- Import et instanciation de la classe Scanner pour la saisie clavier
- Opérateurs arithmétiques : +, -, *, /
- Méthodes d'affichage : System.out.println() et System.out.print()
- Casting de types : (float), (int)
- Libération de ressource : sc.close()

## 2.2 Fichiers du TP2

### ProgBonjour.java — Programme de départ

Fichier généré automatiquement par Eclipse (template). Non modifié par l'élève. Affiche simplement deux messages de bienvenue.

```
System.out.println("Bonjour");
System.out.println("Bienvenue dans l'EDI Eclipse");
```

### Perimetre.java — Calcul du périmètre d'un cercle

Saisit le rayon d'un cercle et calcule son périmètre avec la formule P = 2 × π × R. La valeur de π est définie manuellement en tant que constante float.

```
float PI, Rayon, ResultPeri;
Scanner sc = new Scanner(System.in);
Rayon = sc.nextFloat();
PI = (float) 3.141592;
ResultPeri = 2 * Rayon * PI;
```

### Population.java — Répartition de la population française

Programme sans saisie utilisateur. Les données sont codées en dur (55% d'hommes, 45% de femmes, 65 millions d'habitants). Calcule et affiche le nombre d'hommes et de femmes.

```
H = (float) 0.55;  F = (float) 0.45;  Total = (int) 65000000;
PopuTotalH = H * Total;
PopuTotalF = F * Total;
```

### SalaireBrut.java — Calcul du salaire brut

Saisit le nombre d'heures mensuelles et le taux horaire, puis calcule le salaire brut par multiplication.

```
nbH = sc.nextFloat();
tauxH = sc.nextFloat();
sBrut = nbH * tauxH;
```

### Devis.java — Devis de pose de moquette

Programme le plus complet du TP2. Saisit la largeur, la longueur et le prix TTC de la moquette. Calcule la surface, le montant de la moquette, le coût de pose (fixé à 7€/m²) et le montant total à payer.

```
SurfTotal = Larg * Long;
MontMoq = SurfTotal * PrixMoq;
PrixPose = (int) 7;
MontPose = SurfTotal * PrixPose;
MontTotal = MontMoq + MontPose;
```

## 2.3 Résumé des programmes TP2

| Fichier | Entrées | Traitement | Sortie |
|---|---|---|---|
| ProgBonjour.java | Aucune | Aucun | Affichage message |
| Perimetre.java | Rayon (float) | 2 × π × R | Périmètre du cercle |
| Population.java | Aucune (constantes) | % × total | Nb hommes / femmes |
| SalaireBrut.java | Heures, taux (float) | nbH × tauxH | Salaire brut |
| Devis.java | Larg, Long, Prix (float) | Surface, montants | Devis complet |

# 3. TP4 — Structures de contrôle : boucles et conditions

Le TP4 introduit les structures de contrôle indispensables à tout programme Java : les boucles (for, while, do...while) et les conditions (if/else). Les exercices reprennent et enrichissent les programmes du TP2 en les rendant plus interactifs et robustes.

## 3.1 Nouveaux concepts Java utilisés

- Boucle for : répéter un nombre défini de fois
- Boucle while : répéter tant qu'une condition est vraie
- Boucle do...while : exécuter au moins une fois, puis répéter si condition vraie
- Conditions if / else if / else : tester des valeurs
- Opérateurs de comparaison : <, >, <=, >=, ==, !=
- Opérateurs logiques : && (ET), || (OU)
- Accumulateur : variable mise à jour à chaque tour de boucle (+=)
- Compteurs : incrémenter/décrémenter une variable
- Constante : mot-clé final

## 3.2 Exercices principaux (Exo1 à Exo6)

### Exo1Tp4.java — Moyenne de N notes

L'utilisateur saisit d'abord le nombre de notes, puis chaque note une par une. Une boucle for accumule les notes dans une variable som, puis la moyenne est calculée et affichée avec 2 décimales via printf.

```
Nb_note = sc.nextInt();
for (int i=0; i<=Nb_note; i++) {
    note = sc.nextFloat();
    som += note;
}
System.out.printf("La moyenne des notes est : " + "%.2f", som/Nb_note);
```

### Exo2Tp4.java — Trouver le maximum parmi 5 nombres

Boucle for sur 5 itérations. À chaque tour, si le nombre saisi est supérieur à max, il devient le nouveau max. La variable max est initialisée à 0.

```
for(int i=1; i <= 5; i++) {
    n = sc.nextInt();
    if (n > max) { max = n; }
}
```

### Exo3Tp4.java — Comptage positifs / négatifs / nuls

Saisie de 4 nombres (constante NB = 4). Pour chaque nombre, un if/else if/else incrémente le compteur correspondant (posi, nega, nul). Un contrôle de saisie vérifie que le nombre est entre 1 et 20 ; sinon, la boucle recommence.

```
if (n > 0) { posi++; }
else if (n < 0) { nega++; }
else { nul++; }
```

### Exo4TP4.java — Moyenne avec contrôle de saisie

Reprise de l'Exo1 avec une vérification : la note doit être comprise entre 0 et 20. Si la saisie est invalide, un message d'erreur est affiché et la note n'est pas comptabilisée.

### Exo5TP4.java — Devis moquette en boucle

Reprise du programme Devis du TP2, encapsulé dans une boucle while. Après chaque devis, l'utilisateur est invité à continuer (taper 1) ou quitter (taper 2). La variable sure contrôle la boucle.

```
int sure = 1;
while (sure == 1) {
    // ... calculs du devis ...
    System.out.println("Recommencer ? 1=Oui / 2=Non");
    sure = sc.nextInt();
}
```

### Exo6TP4.java — Total TTC multi-articles

Boucle do...while permettant de saisir plusieurs articles (prix HT + quantité). Le totalHT est accumulé à chaque tour. En sortie de boucle, une remise est appliquée selon le montant (0% si < 500€, 5% si ≤ 1000€, 10% au-delà), puis la TVA (19,6%) est calculée pour obtenir le total TTC.

```
do {
    prixHT = sc.nextFloat();  qte = sc.nextInt();
    totalHT += qte * prixHT;
    choix = sc.nextInt(); // 1=continuer
} while (choix == 1);
if (totalHT < 500) tauxR = 0;
else if (totalHT <= 1000) tauxR = 0.05f;
else tauxR = 0.1f;
```

## 3.3 Exercices complémentaires

### Devis.java (TP4) — Version améliorée du devis

Version corrigée et propre du Devis du TP2, avec des noms de variables plus clairs (largeur, longueur, ttc, surface, montantMoq, mainOeuvre, prixTotal) et l'utilisation d'une constante PRIX_POSE.

```
final int PRIX_POSE = 7;
```

```
surface = largeur * longueur;
montantMoq = ttc * surface;
mainOeuvre = surface * PRIX_POSE;
prixTotal = montantMoq + mainOeuvre;
```

### TotalTTC.java — Facture simple avec remise

Programme de base (sans boucle) pour calculer un total TTC sur un seul article. Sert de base à Exo6TP4. Implémente la même logique de remise et de TVA.

### Exo1TP4Comp.java — Comparaison de tarifs DVD

Calcule quel tarif de location est le plus avantageux entre avec abonnement (39€/an + 3,50€/DVD) et sans abonnement (4,75€/DVD), pour un nombre de DVD saisi par l'utilisateur.

```
avec_abo = nb_DVD * DVD_avec_abo + abo;
sans_abo = nb_DVD * DVD_sans_abo;
if (avec_abo > sans_abo) { /* sans abo moins cher */ }
```

### Exo2TP4Comp.java — Jeu du nombre mystère

Programme de jeu : le programme génère un nombre aléatoire entre 1 et 100 via la classe Random. Le joueur dispose de COUPS_MAX tentatives (5). À chaque essai, le programme indique si le nombre cherché est plus grand ou plus petit. En cas de victoire ou de défaite, le joueur peut rejouer.

```
Random r = new Random();
int n = r.nextInt(101);
for (int i=1; i<=COUPS_MAX; i++) {
    // saisie, comparaison, indication
}
```

### Exo3TP4Comp.java — Maximum d'une série

Saisit une série de nombres jusqu'à la saisie de 0, et affiche le maximum. Programme à l'état d'ébauche, contenant des bugs non corrigés.

## 3.4 Résumé des fichiers TP4

| Fichier | Structure principale | Concept clé |
|---|---|---|
| Exo1Tp4.java | for | Accumulation, moyenne |
| Exo2Tp4.java | for + if | Recherche du maximum |
| Exo3Tp4.java | for + if/else if/else | Compteurs multiples |
| Exo4TP4.java | for + if | Contrôle de saisie |
| Exo5TP4.java | while | Boucle infinie contrôlée |

| | | |
|---|---|---|
| Exo6TP4.java | do...while + if/else if | Multi-articles, remise, TTC |
| Devis.java | Séquentiel | Constante final, nommage clair |
| TotalTTC.java | Séquentiel + if/else | Remise, TVA |
| Exo1TP4Comp.java | if/else | Comparaison de tarifs |
| Exo2TP4Comp.java | do...while + for | Random, jeu interactif |
| Exo3TP4Comp.java | do...while + for (ébauche) | Recherche max avec arrêt |

# 4. Analyse des bugs identifiés

L'analyse du code source révèle plusieurs erreurs logiques dans le TP4. Ces bugs sont typiques des erreurs de débutant et constituent en eux-mêmes un apprentissage précieux.

## 4.1 Bugs dans Exo1Tp4.java

⚠ **Bug de boucle — une itération de trop**

```
for (int i=0; i<=Nb_note; i++) { ... }
```

*→ La boucle commence à i=0 et va jusqu'à i<=Nb_note, soit Nb_note+1 tours au lieu de Nb_note. Correction : démarrer à i=1 ou utiliser i<Nb_note.*

## 4.2 Bugs dans Exo4TP4.java

⚠ **Condition inversée — message affiché pour les notes valides**

```
if (note < 20 && note > 0) {
    System.out.print("La note n'est pas comprise entre 0 et 20...");
}
```

*→ La condition teste si la note est VALIDE (entre 0 et 20), mais affiche le message d'erreur ! Il fallait écrire : if (note < 0 || note > 20).*

## 4.3 Bugs dans Exo6TP4.java

⚠ **Écrasement du totalHT après la boucle**

```
// Dans la boucle (correct) :
totalHT += qte * prixHT;

// Juste après la boucle (bug) :
totalHT = prixHT * qte;  // ← réinitialise avec le dernier article uniquement
!
```

*→ La ligne après la boucle efface toute l'accumulation et réinitialise totalHT avec seulement le dernier article. Cela rend la boucle do...while inutile. Il faut supprimer cette ligne.*

## 4.4 Bugs dans Exo3TP4Comp.java

⚠ **Variable stop non initialisée — erreur de compilation**

```
int nb, fois, max, stop;  // stop jamais assigné !
} while(stop == 0);       // valeur indéterminée → erreur
```

*→ La variable stop n'est jamais initialisée ni assignée. En Java, une variable locale non initialisée génère une erreur à la compilation. Il fallait écrire : int stop = 1; et l'assigner à 0 quand nb == 0.*

⚠ **Point-virgule parasite après if — instruction vide**

```
if(nb==0);{  // le ; termine le if → le bloc {} est toujours exécuté !
}
```

*→ Le point-virgule après if(nb==0) crée une instruction vide comme corps du if. Le bloc {} qui suit est exécuté inconditionnellement. Il faut supprimer ce point-virgule.*

## 4.5 Tableau récapitulatif des bugs

| Fichier | Type de bug | Gravité |
|---|---|---|
| Exo1Tp4.java | Boucle exécutée Nb_note+1 fois | Mineur |
| Exo4TP4.java | Condition inversée (logique contraire) | Majeur |
| Exo6TP4.java | totalHT réinitialisé après la boucle | Majeur |
| Exo3TP4Comp.java | Variable stop non initialisée | Bloquant (compile pas) |
| Exo3TP4Comp.java | Point-virgule parasite après if | Majeur |

# 5. Récapitulatif des concepts Java abordés

## 5.1 Syntaxe de base

| Concept | Exemple | TP |
|---|---|---|
| Déclaration variable | float rayon; | TP2 |
| Initialisation | float PI = 3.141592f; | TP2 |
| Constante | final int PRIX_POSE = 7; | TP4 |
| Cast de type | PrixPose = (int) 7; | TP2 |
| Saisie clavier | Scanner sc = new Scanner(System.in); | TP2 |
| Lecture float | valeur = sc.nextFloat(); | TP2 |
| Lecture int | nb = sc.nextInt(); | TP4 |
| Affichage simple | System.out.println("Résultat : " + valeur); | TP2 |
| Affichage formaté | System.out.printf("%.2f", valeur); | TP4 |

## 5.2 Structures de contrôle

| Structure | Usage dans les TPs | Fichier exemple |
|---|---|---|
| for | Boucle pour N notes, 5 nombres | Exo1Tp4, Exo2Tp4 |
| while | Répéter le devis tant que l'utilisateur veut | Exo5TP4 |
| do...while | Saisir au moins un article, puis proposer de continuer | Exo6TP4 |
| if / else | Déterminer le taux de remise selon le total | TotalTTC, Exo6TP4 |
| if / else if / else | Compter positifs, négatifs, nuls | Exo3Tp4 |

## 5.3 Bonnes pratiques observées

- Commentaires explicites dans le code (// calcul de la moyenne, // saisie)
- Libération des ressources avec sc.close() en fin de programme
- Utilisation de printf et %.2f pour un affichage propre des décimales
- Utilisation de constantes final pour les valeurs fixes (PRIX_POSE, TAUX_TVA)
- Nommage descriptif des variables dans les versions améliorées (largeur, montantMoq...)

## 5.4 Points d'amélioration possibles

- Utiliser Math.PI au lieu de définir PI manuellement
- Décomposer les programmes complexes en méthodes (static float calculerSurface(...))
- Initialiser toutes les variables locales pour éviter les erreurs de compilation
- Vérifier systématiquement les conditions de boucle (i=0 vs i=1, < vs <=)
- Tester les cas limites (0 articles, note = 0, note = 20)

# 6. Conclusion

Ces deux TPs constituent une introduction solide et progressive à Java. Le TP2 pose les fondations (variables, saisie, calcul, affichage) tandis que le TP4 introduit les outils indispensables que sont les boucles et les conditions.

Les programmes réalisés correspondent à des problèmes concrets et réalistes (devis, salaire, jeu, facturation), ce qui favorise la compréhension des concepts. Les bugs identifiés dans le TP4 sont typiques des erreurs de débutant et montrent une progression normale dans l'apprentissage de la logique de programmation.

Ce projet constitue une base solide pour la suite du BTS SIO SLAM, notamment pour l'apprentissage de la programmation orientée objet (classes, objets, héritage) qui sera abordé dans les TPs suivants.

**Note :** Documentation générée à partir de l'analyse complète du code source des fichiers TP2 et TP4.