Travail pratique 2

1) But

Ce travail consiste à mettre en pratique les concepts de la programmation Java vus en classe à travers l'implantation des règles d'affaires (les validations, les calculs, etc.) correspondant aux besoins du client. En effet, l'étudiant aura à mettre en œuvre les structures de sélection et de répétition à travers ses propres méthodes définies tout en respectant les conventions d'écriture du code Java vues en classe.

2) Description et besoins du client

Le respect de l'échéancier et la qualité du premier travail 1 ont encouragé le directeur de l'entreprise **Expert-Plancher** (E-P) à vous confier la suite du développement de son logiciel de facturation. Cette seconde phase consistera à améliorer la première version suite aux remarques des commis du service à la clientèle dont les détails sont décrits ci-dessous.

3) Les spécifications

3.1. Description du menu de choix

Votre programme doit afficher exactement les 4 options du tp1. Les descriptions de ces options sont aussi les mêmes que celles du tp1.

3.2. Saisie et validation des données

L'affichage du message de bienvenue et du résumé de ce que le programme fait, les saisies et les validations de données doivent être définis à l'intérieur des méthodes suivantes :

1) *Affichage du message de bienvenue et de ce que le programme fait* : une méthode "`public`" et "`static`" pour afficher le message de bienvenue et résumé de ce que le programme fait. Cette méthode ne reçoit aucun paramètre, et elle ne retourne rien.
2) *Saisie et validation de l'option choisie par l'utilisateur* : une méthode "`public`" et "`static`" pour saisir et valider l'option choisie par l'utilisateur. Cette méthode ne reçoit aucun paramètre, et elle doit retourner l'option choisie. L'option choisie doit être un nombre entier entre 1 et 4 inclusivement. Si l'option choisie n'est pas dans l'intervalle défini, le message d'erreur suivant doit être affiché : "`L'option choisie est invalide!`".
3) *Saisie et validation du nom du client* : une méthode "`public`" et "`static`" pour saisir et valider le nom du client. Cette méthode ne reçoit aucun paramètre, et elle doit retourner une chaîne de caractères qui correspond au nom. Le nom est une chaîne de caractères dont le nombre minimum de caractères permis est 2 et le maximum est 25. Si le nom saisi ne respecte pas le minimum et le maximum de caractères permis, le message d'erreur suivant doit être affiché : "`Le nom est invalide!`".
4) *Saisie et validation du prénom du client* : une méthode "`public`" et "`static`" pour saisir et valider le prénom du client. Cette méthode ne reçoit aucun paramètre, et elle doit retourner une chaîne de caractères qui correspond au prénom. Le prénom est une chaîne de caractères dont le nombre minimum de caractères permis est 2 et le maximum est 25. Si le prénom saisi ne respecte pas le minimum et le maximum de caractères permis, le message d'erreur suivant doit être affiché : "`Le prénom est invalide!`".
5) *Saisie et validation du numéro de téléphone du client* : une méthode "`public`" et "`static`" pour saisir et valider le numéro de téléphone du client. Cette méthode ne reçoit aucun paramètre, et elle doit retourner une chaîne de caractères qui correspond au numéro de téléphone. Le format du numéro de téléphone doit être : NNN␣NNN-NNNN (␣ = un espace, N = Numérique de 0 à 9).

Si le numéro de téléphone ne respecte pas le format défini, le message d'erreur suivant doit être affiché : "`Le numéro de téléphone est invalide!`".

6) *Saisie et validation de l'adresse du client* : une méthode "`public`" et "`static`" pour saisir et valider l'adresse du client. Cette méthode ne reçoit aucun paramètre, et elle doit retourner une chaîne de caractères qui correspond à l'adresse saisie. Le nombre minimum de caractères est 10 et le maximum est 80. Si l'adresse saisie ne respecte pas le minimum et le maximum de caractères permis, le message d'erreur suivant doit être affiché : "`L'adresse du client est invalide!`".

7) *Saisie et validation de la surface de plancher à sabler et à vernir* : une méthode "`public`" et "`static`" pour saisir et valider la surface. Cette méthode ne reçoit aucun paramètre, et elle doit retourner la surface saisie. Cette surface doit être un nombre réel strictement supérieur à 0. Si la valeur saisie n'est pas supérieure à 0, affichez le message d'erreur suivant : "`La surface est invalide!`".

8) *Saisie et validation de l'identifiant du type de vernis* : une méthode "`public`" et "`static`" pour saisir et valider l'identifiant du type de vernis. Cette méthode ne reçoit aucun paramètre, et elle doit retourner l'identifiant du type de vernis saisi. Les valeurs permises pour cet identifiant sont 101, 102, et 103. Si la valeur saisie n'est pas une des valeurs mentionnées, affichez le message d'erreur suivant : "`L'identifiant du type de vernis est invalide!`".

9) *Saisie et validation de l'identifiant du type de finition* : une méthode "`public`" et "`static`" pour saisir et valider l'identifiant du type de finition. Cette méthode ne reçoit aucun paramètre, et elle doit retourner l'identifiant du type de finition saisi. L'identifiant du type de finition doit être un nombre entier entre 1 et 4 inclusivement. Si la valeur saisie n'est pas dans l'intervalle défini, affichez le message d'erreur suivant : "`L'identifiant du type de finition est invalide!`".

10) *Saisie et validation de la réponse de la question si le client a des escaliers à sabler et à vernir* : une méthode "`public`" et "`static`" pour saisir et valider la réponse de la question. Cette méthode ne reçoit aucun paramètre, et elle doit retourner la réponse de la question saisie. Les valeurs permises sont o, O, n, et N. Si la valeur saisie n'est pas une des valeurs mentionnées, affichez le message d'erreur suivant : "`La réponse est invalide!`".

11) *Saisie et validation du nombre de marches* : une méthode "`public`" et "`static`" pour saisir et valider le nombre de marche. Cette méthode ne reçoit aucun paramètre, et elle doit retourner le nombre de marches saisi. Ce nombre doit être un nombre entier strictement supérieur à 0. Si la valeur saisie n'est pas supérieure à 0, affichez le message d'erreur suivant : "`Le nombre de marches est invalide!`".

12) *Saisie et validation du nombre de contremarches* : une méthode "`public`" et "`static`" pour saisir et valider le nombre de contremarches. Cette méthode ne reçoit aucun paramètre, et elle doit retourner le nombre de contremarches saisi. Ce nombre doit être un nombre entier strictement supérieur à 0. Si la valeur saisie n'est pas supérieure à 0, affichez le message d'erreur suivant : "`Le nombre de contremarches est invalide!`".

13) *Saisie et validation du mode de paiement*: une méthode "`public`" et "`static`" pour saisir et valider l'identifiant du mode de paiement. Cette méthode ne reçoit aucun paramètre et elle doit retourner l'identifiant du mode de paiement saisi. Cet identifiant doit être C ou c, D ou d, et R ou r. Si cet identifiant saisi n'est aucun des caractères précédemment définis, affichez le message d'erreur suivant : "`L'identifiant du mode de paiement est invalide!`".

3.3. Les règles de calcul de la facture

Une fois que les données sont saisies et validées, les règles d'affaires (ou de calcul) doivent être appliquées. Ces règles doivent être définies à l'intérieur des méthodes suivantes :

1) *Déterminer le prix de sablage et de vernissage par pied carré selon l'identifiant du type de vernis saisi* : une méthode "`public`" et "`static`" qui reçoit en paramètre l'identifiant du type de vernis saisi, et elle doit retourner le prix de sablage et de vernissage par pied carré. Les prix par pied carré sont les mêmes que ceux du TP1.

2) *Calculer le sous-total de sablage et de vernissage selon le prix par pied carré et la surface à sabler et à venir* : une méthode "`public`" et "`static`" qui reçoit en paramètres le prix de sablage et de vernissage du plancher par pied carré, la surface à sabler et à vernir, et elle doit retourner le sous-total pour sabler et vernir toute la surface.

3) *Calculer le sous-total des marches d'escaliers à sabler et à vernis* : une méthode "`public`" et "`static`" qui reçoit en paramètre le nombre de marches saisi, et elle doit retourner le sous-total pour sabler et vernir les marches. Le prix par marche est le même que celui du TP1.

4) *Calculer le sous-total des contremarches d'escaliers à sabler et à vernis* : une méthode "`public`" et "`static`" qui reçoit en paramètre le nombre de contremarches saisi, et elle doit retourner le sous-total pour sabler et vernir les contremarches. Le prix par contremarche est le même que celui du TP1.

5) *Calculer le total des sous-totaux* : une méthode "`public`" et "`static`" qui reçoit en paramètres le sous-total de la surface, le sous-total des marches, et le sous-total des contremarches. Elle doit retourner le total des sous-totaux en additionnant les 3 sous-totaux reçus en paramètres.

6) *Calculer le montant de la taxe TPS (5%)* : une méthode "`public`" et "`static`" qui reçoit en paramètre le total des sous-totaux, et elle doit calculer le montant de la taxe TPS dont le taux est 5%. Ensuite elle doit retourner le montant de la taxe TPS calculée.

7) *Calculer le montant de la taxe TVQ (*9.975%*)* : une méthode "`public`" et "`static`" qui reçoit en paramètre le total des sous-totaux, et elle doit calculer le montant de la taxe TVQ dont le taux est 9.975%. Ensuite elle doit retourner le montant de la taxe TVQ calculée.

8) *Calculer le montant total de la facture* : une méthode "`public`" et "`static`" qui reçoit en paramètres le total des sous-totaux, les montants des taxes TPS et TVQ. Ensuite elle doit calculer et retourner le montant total de la facture.

9) *Calculer le montant total de toutes les factures* : une méthode "`public`" et "`static`" qui reçoit en paramètres le montant total de la facture courante, et le montant total de toutes les factures précédentes. Elle doit faire la somme des 2 montants, et retourner la nouvelle valeur.

10) *Incrémenter le numéro de la facture* : une méthode "`public`" et "`static`" qui reçoit en paramètre le numéro de la facture, elle doit ajouter 1 sur ce numéro et retourner la nouvelle valeur.

11) *Obtenir le type de vernis à partir de l'identifiant du vernis saisi* : une méthode "`public`" et "`static`" qui reçoit en paramètre l'identifiant du type de vernis saisi, et elle doit retourner le type de vernis correspondant. Par exemple, si l'identifiant du type de vernis saisi est `101`, le type de vernis retourné doit être "`Le vernis à base d'eau`".

12) *Obtenir le type de finition à partir de l'identifiant du type de finition saisi* : une méthode "`public`" et "`static`" qui reçoit en paramètre l'identifiant du type de finition saisi, et elle doit retourner le type de finition correspondant. Par exemple, si l'identifiant du type de finition saisi est `1`, le type de finition retourné doit être "`Mat`".

13) *Obtenir le mode de paiement à partir de l'identifiant du mode de paiement saisi* : une méthode "`public`" et "`static`" qui reçoit en paramètre l'identifiant du mode de paiement saisi, et elle doit retourner le mode de paiement correspondant. Par exemple, si l'identifiant du mode de paiement saisi est `'C'`, le mode de paiement retourné doit être "`Comptant`".

14) *Incrémenter le nombre de clients par type de vernis* : une méthode "`public`" et "`static`" qui reçoit en paramètres l'identifiant du type de vernis ciblé, l'identifiant du type de vernis saisi, et le nombre de clients. Si l'identifiant du type de vernis saisi est égal à l'identifiant du type de vernis ciblé, le nombre de clients doit être incrémenté et retourné, sinon le nombre de clients doit être retourné sans aucun changement.

3.4. Affichage

Comme dans le travail pratique 1, l'entreprise E-P désire afficher son nom, son adresse (*2021 boulevard Java, Informatique, QC*) et son numéro de téléphone *((438) 182-1100)* dans l'entête de la facture, lors de l'affichage du montant total de toutes les factures ou du nombre de clients par type de vernis. Ces données doivent être définies comme des constantes et utilisées lors des différents affichages.

Les affichages de la facture, du montant total de toutes les factures, et du nombre de clients par type de vernis doivent être définis à l'intérieur des méthodes suivantes :

1) *Afficher la facture* : une méthode "`public`" et "`static`" qui prend en paramètres le numéro de la facture, le nom du client, le prénom du client, le numéro de téléphone du client, l'adresse du client, la surface, l'identifiant du type de vernis, la réponse de la question, l'identifiant du type de finition, l'identifiant du mode de paiement, le prix par pied carré, le sous-total de la surface, le nombre de marches, le sous-total des marches, le nombre de contremarches, le sous-total des contremarches, le total des sous-totaux, le montant de la taxe TPS, le montant de la taxe TVQ, et le montant total. Cette méthode doit afficher les données exactement comme dans la trace d'exécution se trouvant dans le document "TraceExecutionTp2.pdf".

   Les méthodes "*Obtenir le type de vernis à partir de l'identifiant du vernis saisi*", "*Obtenir le type de finition à partir de l'identifiant du type de finition saisi*", et "*Obtenir le mode de paiement à partir de l'identifiant du mode de paiement saisi*" doivent être utilisées dans la méthode d'affichage de la facture. Ces méthodes permettent d'obtenir le type de vernis, le type de finition, et le mode de paiement utilisés dans l'affichage de la facture.

2) *Afficher le montant total de toutes les factures* : une méthode "`public`" et "`static`" qui prend en paramètre le montant total de toutes les factures, et doit afficher les données exactement comme dans la trace d'exécution se trouvant dans le document "TraceExecutionTp2.pdf"

3) *Afficher le nombre de clients par type de vernis* : une méthode "`public`" et "`static`" qui prend en paramètres le nombre de clients pour le vernis à base d'eau, le nombre de clients pour le vernis à base d'huile, et le nombre de clients pour le vernis à base d'alcool. Elle doit afficher les données exactement comme dans la trace d'exécution se trouvant dans le document "TraceExecutionTp2.pdf"

4. Travail demandé

4.1. Organisation de votre programme

Vous devez modifier votre travail pratique 1 en définissant les méthodes mentionnées ci-haut. Le fichier de votre travail doit être nommé `Tp2Inf1120Grpe21A2021.java`. Votre code Java doit tenir compte de toutes les spécifications décrites ci-haut tout en respectant les conventions d'écriture vues en classe. Votre programme doit être organisé de la façon suivante :

1) *Section 1* : Un entête de commentaires de la classe composé de votre nom, de votre prénom, de votre code permanent, du sigle / titre du cours, d'une brève description de ce que la classe doit faire, et de la date d'écriture de votre code. Vous pouvez également ajouter toute autre information que vous jugez pertinente.

2) *Section 2* : Après le nom de la classe, déclarez et initialisez toutes vos constantes à l'extérieur de la méthode `main` à l'aide des mots clés `public`, `static`, et `final`. Aucune constante ne doit être déclarée à l'intérieur de la méthode `main` ou de toute autre méthode.

3) *Section 3* : Définissez toutes les méthodes mentionnées précédemment à l'extérieur de la méthode `main`. Assurez-vous de mettre les commentaires d'entête de chaque méthode (ce que la méthode doit faire, les paramètres et le type de retour). Les noms des méthodes doivent être significatifs.

4) *Section 4* : Les méthodes définies doivent être appelées directement à l'intérieur de la méthode `main` excepté les 3 méthodes mentionnées dans la méthode d'affichage de la facture. Ces appels de méthodes doivent se faire à l'intérieur de votre boucle principale, excepté la méthode d'affichage du message de bienvenue et du résumé de ce que le programme fait.

5) Notez que vous pouvez ajouter toute autre méthode que vous jugez nécessaire pour faciliter votre travail.

4.2. Ce que vous devez remettre

La date de remise du travail pratique 2 est le **mercredi 2 novembre 2021** avant **23H55 via Moodle**. Ce travail doit être remis sous format électronique compressée (.zip ou .rar) contenant le fichier `Tp2Inf1120Grpe21A2021.java` et une page de présentation (document Microsoft Word ou PDF). Vous devez mentionner les informations suivantes sur cette page de présentation : votre nom, votre prénom, le sigle du cours, le groupe du cours, et le nom de l'enseignant.

4.3. Pénalités

Pour tout travail remis en retard, les pénalités seront appliquées selon la formule suivante : `Nombre de points de pénalité = m / 144`, où m est le nombre de minutes de retard par rapport à l'heure de remise. Aucun travail ne sera accepté après cinq (5) jours de retard.

4.4. Plagiat

Le travail pratique 2 est strictement individuel. Le règlement sur le plagiat sera appliqué sans exception. Vous devez ainsi vous assurer de ne pas échanger du code avec des collègues.

5. Pondération.

5.1. Le code source (49 points)

Les noms significatifs pour les constantes, les variables et les méthodes, l'indentation des blocs d'instruction et leur aération, des messages d'invite ou de sollicitation clairs et précis, les bonnes structures de contrôle non redondantes, des commentaires pertinents, le respect du nombre de paramètres, l'ordre des paramètres, les types de paramètres, les types de retour, les commentaires d'entête de la classe et des méthodes, etc. sont des éléments qui seront évalués. Les noms des méthodes doivent commencer par des verbes à l'impératif (Exemple : afficher, saisir, calculer).

5.2. L'exécution du projet Java (50 points)

La validation des saisies, l'application des règles d'affaires (respect des spécifications) et l'obtention des bons résultats lors de génération des factures, etc. sont des éléments qui seront évalués. La note zéro (0) sera attribuée à ce niveau pour tout programme Java qui ne compile pas.

5.3. La page de présentation (1 point)

Une page de présentation bien identifiée avec les informations suivantes : votre nom, votre prénom, le sigle du cours, le groupe du cours, et le nom de l'enseignant.