Travail pratique #1

Ce travail doit être fait individuellement.

Notions mises en pratique : le respect d'un ensemble de spécifications, les instructions Java, les conventions de style Java, et les bonnes pratiques de programmation.

1. Spécifications

Il s'agit de concevoir un petit prototype servant à effectuer la facturation des ventes de billets dans un centre de ski alpin.

1.1 LANCEMENT DU PROGRAMME

Le programme débute en présentant brièvement le logiciel, et affiche ensuite un menu de 4 options, dans l'attente que l'utilisateur entre son choix au menu :

```
Ce programme permet d'effectuer la facturation pour
la vente de billets dans une station de ski alpin.

====
MENU
====
1. Vente de billets
2. Consultation des recettes
3. Reinitialisation
4. Quitter

Entrez votre choix :
```

Votre programme doit valider le choix de l'utilisateur au menu. Les seuls choix valides sont `1`, `2`, `3`, ou `4`. Votre programme ne doit pas planter à l'entrée de caractères non numériques comme `a` ou `e3!`. Pour ce faire, traitez les options de menu 1 à 4 comme des **caractères numériques** plutôt que comme des nombres entiers. De cette manière, il est normal, cependant, que seul le premier caractère de la chaine entrée soit lu lorsque l'utilisateur entre une chaine de caractères plutôt qu'un seul caractère. Ainsi, si l'utilisateur entre `1qw78` comme choix de menu, le programme lira `1`, qui sera donc considéré comme un choix de menu valide (et c'est correct pour ce TP).

Spécifications complémentaires :

Voir les exemples d'exécution fournis avec l'énoncé du TP (fichier **exempleExec_validationMenu.txt**).

Les sections suivantes décrivent chaque option de menu.

Notez qu'après l'exécution d'une option (sauf l'option 4), le programme affiche de nouveau le menu principal, puis attend le prochain choix de l'utilisateur.

1.2 OPTION 1 : VENTE DE BILLETS

Cette option permet de générer la facture suite à la vente de billets dont le prix varie selon la période choisie pour skier, et l'âge des clients. Le tableau 1 ci-dessous montre le prix unitaire d'un billet selon la période choisie (jour, jour et soir ou soirée) et l'âge du client.

Au démarrage du programme (ou lors de sa réinitialisation – voir option 3), on considère qu'aucun billet n'a encore été vendu, et donc qu'aucune facture n'a encore été produite. Les factures produites par la suite (à l'aide de cette option) doivent être numérotées à partir de 1 : la première facture aura le numéro 1, la deuxième facture aura le numéro 2, et ainsi de suite.

Le tableau ci-dessous montre les prix unitaires des billets par période et tranche d'âges.

Tableau 1

Billets journaliers (7 jours semaine)			
Période	**Jour**	**Jour et soir**	**Soirée**
0 à 6 ans	11.52 $	11.52 $	11.52 $
7 à 12 ans	33.92 $	38.92 $	22.18 $
13 à 24 ans	42.62 $	47.84 $	26.31 $
25 à 69 ans	56.53 $	61.75 $	32.62 $
70 ans et plus	37.40 $	37.40 $	26.31 $

** NOTE : Les taxes sont comprises dans les prix.*

Au choix de cette option, le programme affiche un menu des différentes périodes, et demande d'entrer la période désirée :

```
Periodes :
  A. Jour
  B. Jour et soir
  C. Soiree
  D. Terminer

Entrez votre choix :
```

Votre programme doit valider l'entrée de l'utilisateur. Les seules entrées valides sont `A`, `B`, `C` ou `D` (**peu importe la casse**). Il est normal, cependant, que seul le premier caractère de la chaine entrée soit lu lorsque l'utilisateur entre une chaine de caractères plutôt qu'un seul caractère (et c'est correct pour ce TP).

Si l'utilisateur choisit l'option `D`, soit le programme affiche la facture, soit il indique que l'opération est annulée. Plus précisément, si au moins un billet a été ajouté à cette facture, il affiche la facture. Si aucun billet n'a été vendu, le programme annule l'opération (et aucune facture n'est produite). Dans tous les cas, le programme attend, et demande de taper <ENTRÉE> pour continuer (et revenir au menu principal).

Si l'utilisateur choisit une autre option que l'option `D`, le programme affiche un menu avec les différentes tranches d'âges, et demande d'entrer la tranche d'âges désirée.

```
Tranches d'ages :
  A. 0 - 6
  B. 7 - 12
  C. 13 - 24
  D. 25 - 69
  E. 70 et +
  F. Terminer

Entrez votre choix :
```

Votre programme doit valider l'entrée de l'utilisateur. Les seules entrées valides sont `A`, `B`, `C`, `D`, `E`, ou `F` (**peu importe la casse**). Il est normal, cependant, que seul le premier caractère de la chaine entrée soit lu lorsque l'utilisateur entre une chaine de caractères plutôt qu'un seul caractère (et c'est correct pour ce TP).

Si l'utilisateur choisit l'option `F`, le programme revient au menu pour le choix d'une période (ce qui permet de modifier la période précédemment sélectionnée). Sinon, le programme demande d'entrer le nombre de billets pour la période et la tranche d'âges choisies. Votre programme doit valider l'entrée de l'utilisateur entre 0 et 9 inclusivement. Il est normal, ici, que le programme plante à l'entrée d'une valeur autre qu'un entier.

Si l'utilisateur entre 0, aucun billet n'est ajouté à la facture en cours. Sinon, le nombre de billets vendus pour cette période, et cette tranche d'âge est ajouté à la facture (et le programme le mentionne). Dans tous les cas, le programme revient ensuite au menu des tranches d'âge (ce qui permet de sélectionner une nouvelle tranche d'âge pour la même période). Donc, l'utilisateur peut entrer, pour une même période, plusieurs billets de différentes tranches d'âge de suite. Pour ajouter d'autres billets pour une période différente, l'utilisateur doit sélectionner `F` au menu des tranches d'âges, et revenir ainsi au menu des périodes.

Pour terminer la facture et l'afficher, l'utilisateur doit choisir l'option `F` dans le menu des tranches d'âges, et ensuite `D` dans le menu des périodes.

La facture doit montrer :
- Le numéro de la facture.
- Le nombre et la description des billets ajoutés à la facture, et le sous-total (nbr billets * prix unitaire)
- Le total de tous les billets achetés sur cette facture.

Lors de l'affichage de la facture, les montants **affichés** doivent être arrondis à 2 décimales (avec toujours exactement 2 décimales). Par exemple : `32.62`, `165.00`, `37.40`, etc. Pour ce faire, vous DEVEZ utiliser la méthode `format` de la classe `String`. Par exemple, le petit bout de code suivant vous montre comment utiliser cette méthode.

```
double a = 123.4567823;
double b = 45;
double c = 5.3;

String s1 = String.format("%1.2f", a);
System.out.println(s1);  //affiche 123.46 (ou 123,46)

String s2 = String.format("%1.2f", b);
System.out.println(s2);  //affiche 45.00 (ou 45,00)

String s3 = String.format("%1.2f", c);
System.out.println(s3);  //affiche 5.30 (ou 5,30)
```

Notez que la partie décimale peut être indiquée à l'aide d'un point ou d'une virgule (ex : 245.46 ou 245,46). Cela dépend des paramètres de la machines virtuelles sur laquelle s'exécute le programme.

Exemple d'affichage d'une facture :

```
FACTURE # 4

5 billet(s) | JOUR / SOIR | 13 - 24 ans :   239.20 $
2 billet(s) | JOUR        | 7  - 12 ans :   67.84 $
7 billet(s) | SOIREE      | 25 - 69 ans :   228.34 $

TOTAL : 535.38 $
```

Spécifications complémentaires :

Voir les exemples d'exécution fournis avec l'énoncé du TP (fichiers **exempleExec_Options_1_2.txt, exempleExec_Options_1_2_3.txt**).

1.3 OPTION 2 : CONSULTATION DES RECETTES

Cette option permet de consulter les recettes (sommes des montants facturés) depuis le démarrage du programme ou depuis sa dernière réinitialisation. Au démarrage du programme ou après une réinitialisation du programme (voir option 3), les recettes sont nulles. Après l'affichage des recettes, le programme attend, et demande à l'utilisateur de taper <ENTRÉE> pour continuer (et revenir au menu principal).

Spécifications complémentaires :

Voir les exemples d'exécution fournis avec l'énoncé du TP (fichiers **exempleExec_Options_1_2.txt, exempleExec_Options_1_2_3.txt**).

1.4 OPTION 3 : RÉINITIALISATION

Cette option permet de réinitialiser le programme dans l'état où il se trouve lorsqu'on le démarre.

1) Les numéros de facture sont réinitialisés : après une réinitialisation, le numéro de la première facture créée sera 1, le numéro de la deuxième facture créée sera 2, et ainsi de suite.
2) Les recettes sont réinitialisées à zéro.

Spécifications complémentaires :

Voir les exemples d'exécution fournis avec l'énoncé du TP (fichier **exempleExec_Options_1_2_3.txt**).

1.5 OPTION 4 : QUITTER

Cette option permet de terminer le programme. Lorsque cette option est choisie, le programme affiche un message de fin et se termine.

Spécifications complémentaires :

Voir les divers exemples d'exécution fournis avec l'énoncé du TP.

2. Précisions sur les spécifications

- Vous devez écrire votre programme dans une classe nommée `FacturationStationSki` qui contiendra **UNE SEULE méthode**, la méthode `main`, dans laquelle se trouvera TOUT le code de votre programme (sauf les constantes).
- La classe `FacturationStationSki` doit se trouver dans le paquetage par défaut.
- Lorsqu'on vous demande de valider une valeur saisie par l'utilisateur, cela sous-entend une **BOUCLE** de validation : solliciter la valeur, saisir la valeur, et lorsque la valeur saisie est invalide, afficher un message d'erreur, solliciter et saisir de nouveau la valeur, et ce, tant que la valeur saisie n'est pas valide.
- Lorsque votre programme lit un nombre entier, il est normal, à ce stade, que votre programme plante si l'utilisateur saisit autre chose qu'un nombre entier.
- Vous DEVEZ utiliser la classe `Clavier.java` pour effectuer toutes les saisies.
- Commentez bien les parties principales de votre programme, et respectez les bonnes pratiques de programmation vues en classe.
- Les seules CLASSES PERMISES sont `Clavier` (pour lire), et `System` (pour afficher). De plus, vous pouvez / devez utiliser des variables de type `String`, mais NE DEVEZ PAS utiliser aucune des méthodes de la classe `String` (qu'on n'a pas vues encore), à l'exception de la méthode `format` que vous devez utiliser pour obtenir une chaine de caractères contenant un nombre arrondi à deux décimales. De plus, vous pouvez (mais n'êtes pas obligés) utiliser la méthode `toUpperCase` de la classe `Character` pour transformer un caractère en majuscule. Voir la javadoc pour plus de détails.
- Vous DEVEZ respecter à la lettre les **informations, les messages**, et le **format** de l'affichage qui sont montrés dans les exemples d'exécution fournis avec l'énoncé de ce TP.
- **Vous NE DEVEZ PAS utiliser les tableaux** (qu'on n'a pas encore vus).
- Vous NE DEVEZ PAS utiliser les expressions régulières (qu'on ne voit pas dans ce cours).
- Toute **VARIABLE GLOBALE** est **INTERDITE : toutes les variables doivent être déclarées à l'intérieur (et au début) de la méthode `main`** (sauf pour les boucles `for` où vous pouvez déclarer la variable de contrôle dans la boucle).
- Utilisez des constantes (`final`) autant que possible : les messages affichés, les bornes de validation, etc.
- Les constantes DOIVENT être déclarées au niveau de la classe (`public static final`), au début de la classe.
- Vous DEVEZ respecter le style Java.
- Vous DEVEZ utiliser les notions vues au cours pour faire votre TP.
- Votre code doit pouvoir s'exécuter avec le JDK7. Si vous utilisez les notions vues au cours, il n'y aura pas de problème.

- L'affichage des résultats doit se faire à la console.
- Utilisez des variables réelles uniquement lorsque requis. Par exemple, un compteur ne doit pas être un `float` ou un `double`.
- Votre fichier de code source doit être encodé en UTF-8.
- **Le non-respect de toute spécification ou consigne se trouvant dans l'énoncé du TP (et les exemples d'exécution donnés avec l'énoncé du TP) est susceptible d'engendrer une perte de points.**

Note : *Si quelque chose est ambigu, obscure, s'il manque de l'information, si vous ne comprenez pas les spécifications, si vous avez des doutes… vous avez la responsabilité de vous informer auprès de votre enseignante.*

3. Détails sur la correction

3.1 LA QUALITÉ DU CODE (40 POINTS)

Concernant les critères de correction du code, **lisez attentivement** le document "CriteresGeneraux DeCorrection.pdf". De plus, votre code sera noté sur le respect des conventions de style Java vues en classe (dont un résumé se trouve dans le document "ConventionsStyleJavaPourINF1120.pdf". Ces deux documents peuvent être téléchargés dans la section TRAVAUX PRATIQUES (ET BOITES DE REMISE) de la page Moodle du cours.

Note : Votre code sera corrigé sur la totalité ou une partie des critères de correction indiqués ci-dessus, ainsi que sur la qualité de votre code, et le respect des spécifications/consignes mentionnées dans ce document.

3.2 L'EXÉCUTION (60 POINTS)

Un travail qui ne compile pas se verra attribuer la note 0 pour l'exécution.

Note : Votre code sera testé en tout ou en partie. Les points seront calculés sur les tests effectués. Ceci signifie que si votre code fonctionne dans un cas x et que ce cas x n'est pas testé, vous n'obtiendrez pas de points pour ce cas. Il est donc dans votre intérêt de vous assurer du bon fonctionnement de votre programme dans tous les cas possible. Notez que les exemples d'exécution fournis ne couvrent pas nécessairement tous les cas à tester.

4. Date et modalités de remise

4.1 REMISE

Date de remise : Au plus tard le 21 février 2023 à 23h59
Le fichier à remettre : `FacturationStationSki.java` (PAS dans une archive zip, rar, etc.)

VÉRIFIEZ BIEN QUE VOUS AVEZ REMIS LE BON FICHIER SUR MOODLE (le .java et non le .class, par exemple).

Remise via Moodle uniquement.

Vous devez remettre (téléverser) votre fichier sur le site du cours (Moodle). Vous trouverez la boite de remise dans la section **TRAVAUX PRATIQUES (ET BOITES DE REMISE) - REMISE DU TP1**.

4.2 POLITIQUE CONCERNANT LES RETARDS

Une pénalité de 10% de la note finale, par jour de retard, sera appliquée aux travaux remis après la date limite. La formule suivante sera utilisée pour calculer la pénalité pour les retards : **`Nbr points de pénalité = m / 144`**, où **`m`** est le nombre de minutes de retard par rapport à l'heure de remise. Ceci donne 10 points de pénalité pour 24 heures de retard, 1.25 point de pénalité pour 3 heures, etc.

Aucun travail ne sera accepté après 1 jour (24 h) de retard et la note attribuée sera 0.

4.3 REMARQUES GÉNÉRALES

- **Aucun programme reçu par courriel ne sera accepté.** Plus précisément, un travail reçu par courriel sera considéré comme un travail non remis.
- Vous avez la responsabilité de conserver des copies de sauvegarde de votre travail (sur disque externe, Dropbox, Google Drive, etc.). La perte d'un travail due à un vol, un accident, un bris… n'est pas une raison valable pour obtenir une extension pour la remise de votre travail.
- **N'oubliez pas d'écrire (entre autres) votre nom complet et votre code permanent dans l'entête de la classe à remettre.**

N'attendez pas à la dernière minute pour commencer le travail, vous allez fort probablement rencontrer des problèmes inattendus!

Le règlement sur le plagiat sera appliqué sans exception. Vous devez ainsi vous assurer de ne pas échanger du code avec des collègues ni de laisser sans surveillance votre travail au laboratoire. Vous devez également récupérer rapidement toutes vos impressions de programme au laboratoire.

BON TRAVAIL !