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 une petite application permettant de placer des paris sur des courses hippiques virtuelles. Votre programme doit être implémenté dans une seule classe nommée `ParisHippiques` qui ne doit contenir qu'une seule méthode, la méthode `main`.

Fichier fourni :

- `TP1Utils.java` : classe à importer dans votre projet pour pouvoir utiliser la méthode `executerCourse`. **NE PAS MODIFIER** cette classe.

1.1 LANCEMENT DU PROGRAMME

Le programme débute en présentant brièvement le logiciel, et signale ensuite que votre banque est vide, et qu'il faut y déposer un montant avant de pouvoir parier. Votre programme doit valider ce montant. Un montant valide doit être un nombre réel plus grand ou égal à 0. Si le montant valide entré est 0, le programme se termine. Si le montant est plus grand que 0, le programme affiche ensuite le menu principal de 3 options :

```
----
MENU
----
1. Placer un pari
2. Gerer la banque
3. Quitter

Entrez votre choix :
```

Votre programme doit valider le choix de l'utilisateur au menu. Les seuls choix valides sont `1`, `2` ou `3`. 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 3 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 (plus particulièrement les fichiers **exempleExec1.txt** et **exempleExec2.txt**).

Les sections suivantes décrivent chaque option de menu.

1.2 OPTION 1 : PLACER UN PARI

Cette option permet de placer un pari. Le programme offre 4 types de paris.

- Le *pari simple gagnant* consiste à choisir un seul cheval. On remporte ce pari si le cheval choisi est le gagnant de la course.
- Le *pari simple placé* consiste aussi à choisir un seul cheval, mais on remporte ce pari si le cheval choisi se classe dans les deux premières positions de la course.

- Le *pari couplé gagnant ordonné* consiste à choisir un premier cheval, puis un second cheval. On remporte le pari si le premier cheval choisi est le gagnant de la course, et si le second cheval choisi arrive en deuxième position dans la course.
- Le *pari couplé gagnant non ordonné* consiste à choisir deux chevaux. On remporte le pari si les deux chevaux choisis arrivent dans les deux premières positions de la course (peu importe l'ordre).

Au choix de cette option, le programme affiche d'abord un menu des différents types de paris, et demande à l'utilisateur d'en choisir un :

```
Type de pari
  1. Pari simple gagnant
  2. Pari simple place
  3. Pari couple gagnant ordonne
  4. Pari couple gagnant non ordonne
  5. Revenir au menu principal

Entrez le type de pari :
```

Votre programme doit valider le type de pari. Les seuls choix valides sont `1`, `2`, `3`, `4` ou `5`. 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 5 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.

Si le choix valide du type de pari est 5, le programme annule cette option, et réaffiche le menu principal, dans l'attente d'un autre choix de l'utilisateur.

Si le choix valide du type de pari est 1, 2, 3 ou 4 :

1) Saisir et valider le numéro du cheval ou des deux chevaux, selon le type de pari choisi :
   - Pour le choix du type de pari 1 ou 2 :
     - Le programme affiche la liste des chevaux, et demande à l'utilisateur d'entrer le numéro du cheval choisi (nombre entier). Votre programme doit valider ce numéro entre 1 et 6.
   - Pour le choix du type de pari 3 ou 4 :
     - Le programme affiche la liste des chevaux, et demande à l'utilisateur d'entrer le numéro du **premier** cheval choisi (nombre entier). Le programme doit valider ce numéro entre 1 et 6.
     - Le programme affiche de nouveau la liste des chevaux, et demande à l'utilisateur d'entrer le numéro du **deuxième** cheval choisi (nombre entier). Le programme doit valider ce numéro entre 1 et 6. Notez ici que le programme n'empêche pas de choisir un numéro identique au numéro du premier cheval choisi, mais dans ce cas, le pari sera toujours perdu.
2) Le programme demande ensuite d'entrer le montant de la mise. Le programme doit valider ce montant (nombre réel). Un montant valide doit être plus grand ou égal à 0, et plus petit ou égal au montant en banque (la mise ne peut pas dépasser le montant en banque). Si le montant de la mise entré est 0, le programme annule le pari, et demande à l'utilisateur de taper <ENTRÉE> pour revenir au menu principal. Sinon, le programme effectue la course, affiche le résultat du classement, et informe l'utilisateur s'il a gagné ou perdu. Si l'utilisateur a gagné, le programme affiche le montant gagné pour cette course. Il met aussi à jour, et affiche le montant en banque, et le gain (ou la perte) cumulé(e). Finalement, le programme demande à l'utilisateur de taper sur <ENTRÉE> pour revenir au menu principal. Lorsque l'utilisateur tape <ENTRÉE> :
   - S'il n'y a plus d'argent en banque, le programme signale à l'utilisateur que sa banque est vide, et qu'il faut y déposer un montant avant de pouvoir placer d'autres paris. Votre programme doit valider ce montant. Un montant valide doit être un nombre réel plus grand ou égal à 0. Si le montant valide entré est 0, le programme se termine. Si le montant est plus grand que 0, le programme affiche ensuite le menu principal de 3 options, et attend le choix de l'utilisateur.

- Si la banque n'est pas vide, le programme affiche le menu principal de 3 options, et attend le choix de l'utilisateur.

Précisions :

Pour effectuer et afficher la course puis obtenir le classement : le code pour effectuer la course vous est fourni. Pour l'utiliser, vous devez d'abord importer la classe `TP1Utils.java` dans votre projet, et appeler la méthode `executerCourse`, de cette manière : `classement = TP1Utils.executerCourse();` où `classement` est une variable de type `int` qui reçoit le résultat du classement final. Cette variable va contenir un nombre de 6 chiffres, indiquant l'ordre des numéros de chevaux à l'arrivée. Par exemple, le nombre `316254` signifie que le cheval numéro `3` est arrivé en première position, le cheval numéro `1` est arrivé en deuxième position, le cheval numéro `6` est arrivé en troisième position, suivi des chevaux `2`, `5`, et `4`. Dans ce TP, vous avez besoin de connaître seulement les numéros des chevaux arrivés en première, et deuxième position.

- Pour obtenir le numéro du cheval en première position : `classement / 100000`
- Pour obtenir le numéro du cheval en deuxième position : `classement / 10000 % 10`

Attention :

- Vous **DEVEZ** utiliser la méthode `executerCourse` pour obtenir le classement de la course.
- Vous **NE DEVEZ PAS** copier le code de la classe `TP1Utils.java` dans votre classe, mais bien l'importer dans votre projet, et appeler la méthode `executerCourse` comme montré ci-dessus. C'est cette classe que nous utiliserons pour les tests.

Pour calculer le montant gagné (s'il y a lieu) : lorsque l'utilisateur perd son pari, il n'obtient aucun gain. Lorsque l'utilisateur gagne son pari, le montant gagné est calculé selon le type du pari effectué :

- Pour un *pari simple gagnant* : le montant gagné équivaut à **3 fois la mise**.
- Pour un *pari simple placé* : le montant gagné équivaut à **2 fois la mise**.
- Pour un *pari couplé gagnant ordonné* : le montant gagné équivaut à **3.5 fois la mise**.
- Pour un *pari couplé gagnant non ordonné* : le montant gagné équivaut à **2.5 fois la mise**.

Pour mettre à jour la banque : la banque est initialisée à 0 au démarrage du programme. À mesure que l'utilisateur place des paris, pour chaque pari effectué, le montant de la mise doit être soustrait de la banque. Puis, s'il gagne son pari, le montant gagné doit être ajouté à la banque.

Pour mettre à jour le gain ou la perte cumulé(e) : le gain ou la perte cumulé(e) tient le compte des montants perdus et gagnés pendant l'exécution complète du programme. Ce montant est initialisé à 0 au démarrage du programme, et est mis à jour à mesure que l'utilisateur effectue des paris. Chaque fois que l'utilisateur place un pari, le montant de la mise doit être soustrait de ce montant cumulé. Lorsque l'utilisateur gagne un pari, le montant gagné doit être ajouté au montant cumulé. Si le montant cumulé est strictement négatif, on parle de **perte** cumulée, sinon, on parle de **gain** cumulé. Notez que le montant cumulé affiché est toujours positif, c'est le mot **gain** ou **perte** qui change.

Spécifications complémentaires :

Voir les exemples d'exécution fournis avec l'énoncé du TP (plus particulièrement les fichiers **exempleExec3A_option1.txt** et **exempleExec3B_option1.txt**).

1.3 OPTION 2 : GÉRER LA BANQUE

Cette option permet d'ajouter un montant à la banque ou bien de la vider (et l'on suppose ici que l'utilisateur récupère le montant qui était dans la banque). Au choix de cette option, le programme affiche le montant en banque, et demande à l'utilisateur s'il veut ajouter de l'argent, vider la banque, ou revenir au menu principal. Le programme doit valider le choix de l'utilisateur. Un choix valide est `'a'` ou `'A'` pour **a**jouter un montant à la banque, `'v'` ou `'V'` pour **v**ider la banque ou bien `'r'` ou `'R'` pour **r**evenir au menu principal.

Lorsque l'utilisateur choisit d'ajouter un montant à la banque, le programme demande le montant à ajouter. Le programme doit valider ce montant. Un montant valide doit être plus grand ou égal à 0. Si le montant entré est 0, le programme affiche de nouveau le montant dans la banque puis redemande à l'utilisateur s'il veut **a**jouter un montant, **v**ider la banque, ou **r**evenir au menu principal.

Si le montant valide entré est plus grand que 0, celui-ci est alors ajouté au montant déjà présent dans la banque. Le programme affiche ensuite le nouveau montant dans la banque puis redemande à l'utilisateur s'il veut **a**jouter un montant, **v**ider la banque, ou **r**evenir au menu principal.

Lorsque l'utilisateur choisit de vider la banque, le programme se termine.

Lorsque l'utilisateur choisit de revenir au menu principal, le programme affiche le menu principal, en attente du prochain choix de l'utilisateur.

Spécifications complémentaires :

Voir les exemples d'exécution fournis avec l'énoncé du TP (plus particulièrement le fichier **exempleExec4_option2.txt**).

1.4 OPTION 3 : QUITTER LE PROGRAMME

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 exemples d'exécution fournis avec l'énoncé du TP (plus particulièrement le fichier **exempleExec2.txt**).

2. Précisions supplémentaires

- Vous devez écrire votre programme dans une classe nommée `ParisHippiques` qui contiendra UNE SEULE méthode, la méthode `main`, dans laquelle se trouvera TOUT le code de votre programme (sauf les constantes).
- La classe `ParisHippiques` doit se trouver dans le paquetage par défaut.
- Les montants d'argent doivent tous être affichés avec exactement 2 décimales (arrondis à 2 décimales si le nombre a plus de 2 décimales). Utilisez la méthode `System.out.printf` pour ce faire. Les nombres affichés avec cette méthode peuvent contenir une virgule ou bien un point pour afficher la partie décimale (ex : 23.50 ou 23,50).
- La méthode `executerCourse` de la classe `TP1Utils`, que vous devez utiliser, retourne des classements aléatoires. Il est donc normal que vous n'obteniez pas les mêmes résultats que ceux montrés dans les exemples d'exécution. Les exemples d'exécution ne sont pas des tests, mais servent plutôt à montrer le comportement du programme dans tous les cas, les messages affichés, etc. 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.
- 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 / réel, il est normal, à ce stade, que votre programme plante si l'utilisateur saisit autre chose qu'un nombre entier / réel.
- 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 vue encore).
- Vous NE DEVEZ PAS utiliser les tableaux (qu'on n'a pas vus encore).

- Vous NE DEVEZ PAS utiliser les expressions régulières (qu'on ne verra pas).
- 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.
- 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`.
- 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.**
- 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 modalité de remise

4.1 REMISE

Date de remise : Au plus tard le 23 octobre 2022 à 23h59
Le fichier à remettre : `ParisHippiques.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 !