TD d'algorithmique M2202 – TD7 à 9 – Programmation objet

Rappels de cours :

Le type `String` vu précédemment est en fait une **classe** prédéfinie du langage Java qui permet de manipuler des suites de caractères (des « chaînes de caractères »).

La classe `String` fournit des méthodes qui permettent la recherche de mots, ou de caractères, bref, de sous-chaînes, dans un texte (méthode `substring`). Les mots peuvent aussi être comparés suivant l'ordre alphabétique (méthode `compareTo`, qui renvoie 0 si la chaîne de l'objet est égale à celle fournie en entrée, une valeur positive si la chaîne de l'objet est supérieure à celle fournie en entrée, une valeur négative sinon). La liste complète des méthodes de la classe String est décrite sur la page http://docs.oracle.com/javase/7/docs/api/java/lang/String.html. Voici par exemple la documentation fournie pour les méthodes `substring`, `compareTo` et `compareToIgnoreCase` :

`int`	`length()` Returns the length of this string.
`int`	`compareTo(String anotherString)` Compares two strings lexicographically.
`int`	`compareToIgnoreCase(String str)` Compares two strings lexicographically, ignoring case differences.

`String`	`substring(int beginIndex)` Returns a new string that is a substring of this string.
`String`	`substring(int beginIndex, int endIndex)` Returns a new string that is a substring of this string.

L'étude des objets de type `String` nous montre qu'une classe est une association de données appelées « **propriétés** » (information, valeur de tout type) et de fonctions appelées « **méthodes** » (outils d'accès et de transformation des données). Ces méthodes, définies dans une classe, ont directement accès aux données de cette même classe.

Le langage Java offre aussi la possibilité à la personne qui programme de développer ses propres classes. Construire une classe, c'est définir un nouveau type. Pour cela, il est nécessaire de :
• déterminer les caractéristiques communes à ce que l'on souhaite décrire. Ce sont les **propriétés**, c'est-à-dire les données, les attributs, ou encore les membres de la classe.
• définir les **méthodes**, c'est-à-dire toutes les opérations et traitements, réalisables sur ces éléments. Ces opérations sont aussi appelées comportements.

Une classe définissant un type structuré n'est pas une application directement exécutable. Elle ne contient pas de fonction `main`.

Les types structurés sont utilisés dans les applications, en déclarant des variables dont le type correspond au nom de la classe définie précédemment, comme le montre l'instruction suivante :

```
TypeDeLObjet chose = new TypeDeLObjet();
```

L'opérateur **new** détermine l'adresse où stocker les informations relatives à l'objet déclaré. Il réserve l'espace mémoire nécessaire pour stocker les données et les méthodes de la classe. Les données sont initialisées à 0 pour les entiers, 0.0 pour les réels, '\0' pour les caractères et `null` pour tous les autres types structurés.

À cette étape, la variable `chose` contient ce qu'on appelle dans le jargon informatique un **objet**. Un objet est donc un élément particulier, un représentant d'une classe définissant un type structuré. On dit aussi que c'est une **instance** de la classe. Les données (propriétés ou attributs) qui définissent cette instance sont stockées dans les variables de classe associées à cette instance particulière, appelée **variables d'instance**.

L'accès aux variables d'instance ainsi qu'aux méthodes de la classe se fait par l'intermédiaire de l'opérateur « . » (le point). On peut le voir dans l'exemple suivant, en supposant que la donnée `nomDeLaDonnee` et la méthode `nomDeLaMethode` ont été préalablement définies dans la classe `TypeDeLObjet` de l'objet `chose` :

```
// affectation d'une valeur dans la propriété de l'objet chose :
chose.nomDeLaDonnee = [valeur du bon type] ;
// appel de la méthode sur l'objet chose :
chose.nomDeLaMethode([liste des paramètres éventuels, précédés de leur type]) ;
```

Exercice : le jeu de chifoumi

On souhaite lors de ces séances construire un jeu de chifoumi (pierre/feuille/ciseaux), dont les joueurs seront soit des humains (ils choisiront leurs actions au clavier dans la ligne de commande), soit des « robots » (qui choisiront leurs actions au hasard).

Partie 1 – Construire la classe JoueurChifoumi

Cette partie 1 consiste à créer la classe `JoueurChifoumi` : vous ajouterez progressivement le contenu en fonction des questions. Des indications sont fournies au verso si vous n'arrivez pas à avancer.

Q1. Créez une classe `JoueurChifoumi` en précisant quel sera le nom du fichier (laissez 2 lignes vides avant le début).

Q2. Ajoutez des propriétés à cette classe : le pseudo du joueur, son nombre de points, et son type (0 pour robot, 1 pour humain).

Q3. Ajoutez deux constructeurs à cette classe : un qui prend en entrée seulement le pseudo, un autre qui prend en entrée le pseudo ainsi que le code de type de joueur (0 pour un robot, 1 pour un humain).

Q4. Ajoutez une méthode `toString` à la classe `JoueurChifoumi`. Cette méthode, qui ne prend rien en entrée, renvoie le nom du joueur correspondant à l'objet, puis entre parenthèses « robot » ou « humain » selon le type de joueur, puis le nombre de points du joueur (le mot « point » doit être correctement accordé au singulier ou au pluriel en fonction du nombre de points).

Q5. Ajoutez une méthode `joueUnCoup` à la classe `JoueurChifoumi`. Cette méthode, qui n'a aucune entrée, adopte le comportement suivant. Si l'objet est un robot, elle choisit aléatoirement un nombre entre 0 inclus et 1 exclu, puis si ce nombre est inférieur strictement à 0.3333, elle renvoie « pierre » ; s'il est compris entre 0.3333 et 0.6666 inclus, elle renvoie « feuille » ; et sinon elle renvoie « ciseaux ». Si l'objet est humain, la méthode lui demande « Pourriez-vous choisir entre pierre, feuille et ciseaux ? », récupère sa réponse, et la renvoie.

Pour choisir un nombre aléatoire, vous pouvez utiliser la fonction `Math.random` :
• importez tout d'abord la bibliothèque `Math` en insérant `import java.lang.Math;` avant le début de la classe (d'où la ligne vide à la question Q1...)
• appelez ensuite la fonction `Math.random()` au moment voulu : elle renvoie un nombre flottant (de type `double`) aléatoirement choisi entre 0 inclus et 1 exclu.

Q5 bis (bonus). Si jamais deux joueurs humains jouent à la suite, il faut cacher la réponse de chaque joueur humain après qu'il l'a entrée au clavier dans la ligne de commande. Pour cela, ajoutez 30 retours à la ligne après que tout humain a entré sa réponse.

Partie 2 – Construire la classe JeuChifoumi

On va désormais construire la classe principale `JeuChifoumi` qui contient la fonction `main`, et qui va utiliser la classe `JoueurChifoumi`.

Q6. Commencez le code de cette classe en laissant 2 lignes libres pour les imports de bibliothèques, et en ajoutant une fonction `main`, qui sera remplie dans les questions suivantes.

Q7. Dans la fonction main, créez un objet correspondant à un robot joueur de chifoumi, enregistrez-le dans une variable nommée `joueur1`.

Q8. Dans la fonction main, enregistrez dans une variable nommée `joueur2` un objet de la classe `JoueurChifoumi` qui vous correspond (donc joueur humain avec le pseudo que vous choisirez).

Q9. En appelant la méthode appropriée de la classe `JoueurChifoumi`, affichez dans la console les informations sur les deux joueurs créés.

Q10. Créez une fonction `meilleurCoup` qui prend en entrée deux chaînes de caractères (chacune contiendra soit « pierre », soit « feuille » soit « ciseaux »), et renvoie 0 si les deux chaînes sont égales ; renvoie 1 si la première « bat » la seconde (c'est-à-dire que la première est « pierre » et la seconde « ciseaux », ou bien la première « ciseaux » et la seconde « feuille », ou bien la première « feuille » et la seconde « pierre ») ; ou bien renvoie -1 sinon.

Q11. Créez une fonction `lancePartie` qui prend en entrée deux objets de la classe `JoueurChifoumi`, puis fait choisir leur coup aux deux joueurs correspondants (peu importe s'ils sont humains ou robots), appelle la fonction `meilleurCoup` pour savoir qui a gagné, et attribue 3 points au vainqueur, 0 point au vaincu, et 2 points à chacun en cas d'ex-aequo.

Indications

Ne lisez ces indications que si vous n'arrivez pas à avancer sur les questions concernées.

Q1. Il n'y a qu'une seule ligne à écrire à cette question pour le début de la classe
Q2. Les propriétés sont des variables de classe : il suffit de les déclare, c'est-à-dire écrire leur nom précédé de leur type.
Q3. Chaque constructeur devra initialiser les variables de classe en fonction des valeurs des variables qui lui seront fournies en entrée. On rappelle que pour distinguer les variables d'entrée des constructeurs des variables de classe, il faut insérer « this. » devant ces dernières, pour montrer qu'elles se rapportent bien à l'objet qu'on est en train de créer avec le constructeur.
Q4. Il faut tout d'abord correctement déclarer la méthode `toString`, avec le bon type de sortie et aucune entrée. Puis il s'agit de créer progressivement la chaîne de caractères qui sera renvoyée (attention : « renvoyée » et pas « affichée »), en concaténant toutes les informations demandées, qui seront trouvées dans les variables de classe. Puis on terminera en renvoyant la chaîne.
Q5. Commencez par déclarer la méthode avec le bon type de sortie et aucune entrée. Ensuite, utilisez des tests pour distinguer les différents cas. Pour le cas où le joueur est un robot, il faudra bien penser à stocker la valeur choisie aléatoirement, pour qu'il n'y ait pas un nouveau choix de nombre aléatoire à chaque test effectué. Enfin, il faudra renvoyer la chaîne de caractères correspondant à la valeur choisie.
Q5 bis. Pour ajouter trente retours à la ligne, il faut utiliser la fonction qui permet d'afficher une chaîne de caractères avant de faire un retour à la ligne, et utiliser une boucle pour répéter 30 fois l'appel de cette fonction.
Q6. Il suffit ici de se rappeler ce que la fonction `main` renvoie en sortie et ce qu'elle prend en entrée, pour la déclarer correctement (cours 4 du premier semestre).
Q7. Il faut bien penser à déclarer la variable, puis l'initialiser en utilisant le constructeur de la classe `JoueurChifoumi`, avec les bonnes informations en entrée pour que l'objet soit un robot et non un humain.
Q8. Même chose qu'à la question précédente sauf que là il s'agit d'un humain et non d'un robot.
Q9. Il faut appeler la méthode `JoueurChifoumi` sur chacune des deux variables `joueur1` et `joueur2`.
Q10. Il faut utiliser des tests pour vérifier les différents cas possibles en comparant les valeurs des chaînes de caractères fournies en entrée avec les chaînes « pierre », « feuille » et « ciseaux » : pour cela il faut utiliser la méthode `compareTo` sur les chaînes de caractères fournies en entrée (voir début de la page précédente). N'oubliez pas que vous pouvez utiliser `&&` (signifie « et »).
Q11. Déclarez correctement `lancePartie` avec deux objets de la classe `JoueurChifoumi` en entrée, et un entier renvoyé en sortie. Pour faire choisir le coup joué par chaque joueur, il suffit d'appeler la méthode `joueUnCoup` de chaque joueur donné en entrée de `lancePartie`, transmettre les résultats en entrée de `meilleurCoup`, puis renvoyer le bon nombre de points.

Partie 3 – Améliorer la classe JoueurChifoumi

On va désormais améliorer la classe `JoueurChifoumi`.

Q12-a. Dans le code actuel, le joueur doit écrire exactement « pierre », « feuille » ou « ciseaux ». Il serait utile que sa réponse soit prise en compte même s'il fait des erreurs. En utilisant la méthode `substring` de la classe `String`, modifiez le code de la méthode `joueUnCoup` de la classe `JoueurChifoumi` pour que la réponse entrée par l'utilisateur soit transformée en une des trois réponses attendues, si elle lui correspond :
• si une des trois réponses attendues est un préfixe de la réponse entrée par l'utilisateur (par exemple : si l'utilisateur a entré la chaîne de caractères "pierre !", "pierre" est un préfixe de sa réponse, donc la méthode `joueUnCoup` renverra "pierre")
• si la réponse entrée par l'utilisateur est un préfixe d'une des trois réponses attendues (par exemple : si l'utilisateur a entré la chaîne de caractères "pi", c'est un préfixe de "pierre", donc la méthode `joueUnCoup` renverra "pierre")

Q12-b. Modifiez le code de la question précédente pour que les réponses de l'utilisateur soient prises en compte quelle que soit la casse (majuscules et minuscules) utilisée.

Q13. Ajoutez dans la classe `JoueurChifoumi` une méthode `compareTo` qui prend en entrée un autre objet `autreJoueur` de la classe `JoueurChifoumi`, et renvoie la différence de points entre le joueur correspondant à l'objet sur lequel s'applique cette méthode `compareTo` et le joueur correspondant à `autreJoueur` (valeur positive si `autreJoueur` a moins de points, nulle s'il a autant de points, négative s'il a plus de points).

Q14. Ajoutez une propriété `dernierCoup` à la classe `JoueurChifoumi` pour stocker le dernier coup joué par le joueur. Faites les modifications nécessaires dans le code de la classe pour que `dernierCoup` stocke effectivement le dernier coup joué par le joueur.

Q15. Ajoutez une méthode `aBattu` à la classe JoueurChifoumi, qui prend en entrée un autre objet `autreJoueur` de la classe `JoueurChifoumi`, renvoie 1 si cet autre joueur a moins de points, 0 s'il a autant de points, et -1 s'il a plus de points que le joueur sur lequel s'applique cette méthode `aBattu`. Dans la fonction `main`, enregistrez dans une variable nommée `joueur2` un objet de la classe `JoueurChifoumi` qui vous correspond (donc joueur humain avec le pseudo que vous choisirez).

Partie 4 (bonus) – Ajouter un peu de stratégie

On va désormais améliorer la classe `JoueurChifoumi` pour permettre d'ajouter différentes stratégies de jeu : une pour le robot et une pour l'humain.

Q16. Ajoutez 3 propriétés `probaPierre`, `probaFeuille`, `probaCiseaux` (des entiers) telles que si le joueur est un robot, il a `probaPierre` % de chances de tirer « pierre », `probaFeuille` % de chances de tirer « feuille », `probaCiseaux` % de chances de tirer « ciseaux », si les trois propriétés somment à 100. Sinon, une règle de trois est appliquée pour respecter les poids respectifs attribués à ces trois propriétés (par exemple : les poids 1, 2 et 2 sont transformés en 20%, 40% et 40%).

Remarque : attention aux arrondis, arrangez-vous, en cas d'utilisation de la règle de 3, pour que la somme des trois poids transformés fasse bien 100% !

Q17. Ajoutez un constructeur pour définir ces trois poids, et ajoutez dans la classe `JeuChifoumi` un appel de ce constructeur qui choisit les 3 valeurs au hasard (sans vérifier que ça somme à 100).

Q18. Dans le cas où un humain joue contre un robot, ajoutez du code pour enregistrer le nombre de réponses de chaque type envoyées par l'adversaire, et afficher à un utilisateur humain, avant de lui demander son choix, la réponse la moins choisie par son adversaire (robot), ainsi que le pourcentage de fois où son adversaire a choisi cette réponse.