TP N° 2

Exercice 1

Le but de cet exercice est de réaliser le jeu de Vache-Taureau. Ce jeu consiste à trouver un nombre caché en proposant quatre chiffres. Si la proposition du joueur est égale au nombre caché, le joueur va gagner et sinon il doit donner une autre proposition. Au bout de 10 propositions erronées le joueur perd.

Pour réaliser ce jeu, nous allons :

1) Créer une classe ***VacheTaureau*** qui contiendra les attributs privés qui sont décrits comme suit :
   - un tableau d'entiers ***nSolution*** pour représenter le nombre caché ;
   - un tableau d'entiers ***nProp*** pour représenter un nombre proposé par le joueur ;
   - un entier ***nTentatives*** pour compter les tentatives (c.-à.-d. les propositions de l'utilisateur).
   - un entier ***nVache*** pour compter le nombre de chiffres proposés par le joueur qui sont contenus dans le nombre caché mais pas à leurs places.
   - un entier ***nTaureau*** pour compter le nombre de chiffres proposés par le joueur qui sont contenus dans le nombre caché et à leurs places.

2) De plus cette classe aura plusieurs méthodes :
   a) un constructeur qui prend un entier formé de quatre chiffres en argument et initialise les deux tableaux et les compteurs.

Exemple : Si l'entier fourni est "1234", alors :

- le tableau ***nSolution*** contiendra

1	2	3	4

- le tableau ***nProp*** contiendra

0	0	0	0

- ***nTentatives***=0, ***nVache***=0 et ***nTaureau***=0.

   b) une méthode ***boolean gagne()*** qui renvoie true si le joueur a gagné et false sinon. Le joueur gagne si la valeur de ***nTaureau*** est égale 4 et la valeur de ***nTentatives*** est inférieure strictement à 10.

   c) une méthode ***boolean perdu()*** qui renvoie true si le joueur a perdu et false sinon. Le joueur perd si la valeur de ***nTentatives*** est égale à 10 et la valeur de ***nTaureau*** est inférieure strictement à 4.

   d) une méthode ***void propose(int val)*** qui
      i. insère la valeur ***val*** dans le tableau d'entiers ***nProp***,
      ii. met les valeurs de ***nVache*** et ***nTaureau*** à zéro,
      iii. incrémente la valeur de ***nTentatives***,

      iv. compare ***nProp*** avec ***nSolution*** et met à jour les compteurs de la manière suivante :
      - si un chiffre de ***nProp*** existe dans ***nSolution*** et dans sa place alors ***nTaureau*** sera incrémenté de 1.
      - si un chiffre de ***nProp*** existe dans ***nSolution*** et n'est pas dans sa place alors ***nVache*** sera incrémenté de 1.

Exemple : Si la proposition est "6431" alors : ***nTentatives***=1, ***nVache***=2 et ***nTaureau***=1.

   e) Une méthode ***void etat( )*** qui affiche l'état du jeu, c.-à.-d. le nombre de tentatives, le nombre proposé contenu dans ***nProp***, le nombre de vaches et celui de taureaux.

3) Ajouter une classe publique ***ProgrammeVacheTaureau*** comportant :
   a) Une méthode statique boolean verif(int n) qui retourne true si un entier n de 4 chiffres admet des chiffres sans redondance, false sinon.

   b) Une méthode ***main*** et permettant de :
   - créer un objet de la classe ***VacheTaureau*** en générant un entier au hasard comportant quatre chiffres non redondants;
   - afficher :
     « le jeu commence ! »
   - tant que le jeu n'est ni gagné, ni perdu, demander au joueur de saisir un entier comportant quatre chiffres non redondants et d'afficher l'état du jeu,
   - dès que le jeu est gagné ou perdu : si le jeu est gagné afficher "Bravo !!!!", sinon afficher "Perdu, la solution était : solution".

Exercice 2

Le but de cet exercice est de modéliser le jeu ***Motus***. Ce jeu consiste à déterminer un mot caché en proposant des lettres. Si la proposition du joueur est égale au mot caché, le joueur va gagner et sinon il a un malus (une erreur). Au bout de 10 malus le joueur perd. Pour réaliser ce jeu, nous allons :

1) Créer une classe ***Motus*** qui contiendra les attributs privés suivants :
- un vecteur de caractères ***TSolution*** pour représenter le mot caché ;
- un vecteur de caractères ***TProp*** pour représenter le mot proposé par le joueur ;
- un vecteur de caractères ***TCouleur*** pour représenter l'état du mot proposé ;
- un entier ***nMalus*** pour compter les malus.
- Une chaîne de caractères ***nomJoueur*** pour représenter le nom du joueur

De plus cette classe aura plusieurs méthodes :

a) un constructeur qui prend une chaîne de caractères en argument et initialise les trois vecteurs et le compteur.

Exemple : Si la chaîne fournie est "ABCDE", alors :
- le vecteur TSolution contiendra :

A	B	C	D	E

- les vecteurs TProp et TCouleur contiendront

-	-	-	-	-

- nMalus=0.

b) Une méthode nommée ***setNom*** qui change le nom du joueur par une valeur passée en argument.

c) Une méthode nommée ***remplir*** qui prend en paramètre une chaîne de caractères ch et qui :
- insère cette chaîne ***ch*** dans le vecteur ***TProp***
- remplit le vecteur ***TCouleur*** de façon à ce que chaque élément d'indice i contienne : 'B' si le caractère dans ***TProp*** correspondant à cet indice existe dans ***TSolution*** et est bien placé (même endroit), 'M' si le caractère correspondant dans ***TProp*** existe dans ***TSolution*** mais est mal placé, et 'N' si le caractère correspondant n'existe pas dans ***TSolution***.
- incrémente la valeur de ***nMalus***.

Exemple :

Si le vecteur ***TSolution*** *contient*

A	B	C	D	E

et ***ch***=''ACFKM''

alors les vecteurs ***TProp*** *et* ***TCouleur*** *contiendront :*

TProp

A	C	F	K	M

TCouleur

B	M	N	N	N

d) Une méthode nommée ***gagner*** qui renvoie ***true*** si le joueur a gagné et ***false*** sinon. Le joueur gagne si tous les éléments du vecteur ***TCouleur*** contiennent la lettre 'B'.

e) Une méthode nommée ***perdre*** qui renvoie ***true*** si le joueur a perdu et ***false*** sinon. Le joueur perd si la valeur de ***nMalus*** égale à 10 et si au moins un élément du vecteur ***TCouleur*** ne contient pas la lettre 'B'.

f) Une méthode statique nommée ***saisiePropo*** qui permet de saisir et de retourner une chaîne de caractères non vide de taille 5 et comportant des caractères sans redondance.

g) Une méthode nommée ***afficheEtat*** qui en se basant sur le contenu du vecteur ***TCouleur***, affiche à l'utilisateur une par une les lettres qu'il a saisi dans ***TProp*** en lui indiquant si chacune d'elles est bien placée, mal placée, ou n'existe pas dans le mot caché. Cette méthode doit afficher le nom du joueur.

Exemple : *Si le vecteur* ***TSolution*** *contient*

A	B	C	D	E

et les vecteurs ***TProp*** *et* ***TCouleur*** *contiennent respectivement*

TProp

A	C	F	K	M

TCouleur

B	M	N	N	N

alors cette méthode doit afficher :

La lettre A est bien placée
La lettre C est mal placée
La lettre F n'existe pas
La lettre K n'existe pas
La lettre M n'existe pas

2) Ajouter une classe publique ***ProgrammeMotus*** contenant la méthode main et permettant de :

a) Créer un tableau T de chaînes de caractères contenant les éléments suivants : "ABCDE", "EFRTU", "TYUIO", et "HYUOP".

b) Créer un objet de la classe ***Motus*** en tirant un mot au hasard à partir de la liste de mots contenus dans T.

c) Demander à l'utilisateur de saisir son nom.

d) Afficher :

"Le jeu Motus commence maintenant... "

e) Tant que le jeu n'est ni gagné, ni perdu, demander au joueur de saisir un mot qui comporte cinq lettres (sans redondance des caractères) et afficher les lettres de sa proposition en lui indiquant l'état de chacune des lettres versus le mot caché (bien placé, mal placé ou n'existe pas).

f) Dès que le jeu est gagné ou perdu : si le jeu est gagné, le programme affiche "Bravo !!!! Vous avez gagné", sinon il affiche "Vous avez perdu, la solution est : solution".