Programmation orientée objet

DOCUMENTS NON AUTORISES sauf supports distribués en cours

(le barème est donné à titre indicatif)

1. Exercice (4pts)

```
import java.util.ArrayList;
import java.util.Random;
public class Main {
  protected static String [] temps = {"past", "now", "future"};
  public static void swap(ArrayList<String> stk, int i1, int i2) {
    String tmpt = stk.get(i1);
    stk.set(i1, stk.get(i2));
    stk.set(i2, tmpt);
  }
  public static void queFaitIl(ArrayList<String> stockage) {
    int indA = 0;
    int indB = stockage.size()-1;
    while (indA <= indB) {
      if ((stockage.get(indA)).equals("past")) {
        String obj = stockage.get(indA);
        stockage.remove(indA);
        stockage.add(0, obj);
        indA++;
      } else
        if ((stockage.get(indA)).equals("now"))
          indA++;
        else {
          swap(stockage, indB, indA);
          indB--;
        }
      System.out.println ("indA : "+indA+" indB : "+indB+ " liste : "+stockage);
    }
  }
  public static void main(String[] args) {
    Random rd = new Random();
    ArrayList<String> list = new ArrayList<String> ();
    for (int i = 0; i < 5; i++) {
      int index = rd.nextInt(3);
      String t = temps[index];
      list.add(t);
    }
    System.out.println (""+list);
    queFaitIl(list);
    System.out.println (""+list);
  }
}
```

Faire dérouler le code précédent et donner le résultat de l'exécution en précisant à chaque tour de boucle le résultat de l'affichage à l'écran.

2. Problème (16 pts)

On souhaite écrire un programme Java qui permet à un joueur de jouer à Shifumi contre le programme. Le joueur choisit un label parmi la liste : CAILLOU, PAPIER ou CISEAUX, et le programme choisit au hasard un des trois labels. S'il y a égalité, un point est accordé au joueur et au programme. Sinon, deux points sont accordés au gagnant, selon les règles habituelles du Shifumi :

- CAILLOU gagne contre CISEAUX, perd contre PAPIER
- PAPIER gagne contre CAILLOU, perd contre CISEAUX
- CISEAUX gagne contre PAPIER perd contre CAILLOU

Le gagnant à la fin de la partie est celui qui a obtenu le plus de points.

On gère l'obtention d'un label par le programme grâce à un tirage aléatoire.

On considère que l'on dispose de tous les accesseurs nécessaires …

On dispose du code de la classe **Joueur** qui contient :
- le nom,
- le score,
- un label (qu'il ait été choisi par le joueur ou bien tiré au sort pour le programme),
- un constructeur qui initialise les attributs,
- **void gagner(int gain)** méthode qui ajoute le gain au score du joueur.

[Q1] Ecrire le code de l'énumération **Shifu** qui contient :

1) les trois labels correspondant aux trois possibilités offerte au joueur : CAILLOU, PAPIER, CISEAUX

2) une méthode statique **public static Shifu get(int index)** qui prend en entrée un index (dans l'exemple compris entre 0 et 2) et qui retourne le label correspondant.

[Q2] Ecrire le code de la classe **Shifumi** qui contient :
- un attribut : ArrayList de joueurs (ici 2 joueurs, une personne et la machine),
- un attribut : instance de **Random**, pour pouvoir faire des tirages au sort de labels,
- un attribut : instance de **Scanner** pour pouvoir faire la saisie au clavier,
- une méthode : **String toString()** qui renvoie une chaîne de caractères représentative de l'état du jeu.

Ecrire le code du constructeur qui crée le joueur dont le nom est reçu en paramètre, et le joueur "machine".

Ecrire le code de la méthode : **boolean choisir(Joueur j)** qui fait choisir le joueur j en lui demandant de saisir son choix au clavier. La méthode modifie le label courant du joueur et renvoie **true** si le joueur saisit 0 pour CAILLOU, 1 pour PAPIER, ou 2 pour CISEAUX, et renvoie **false** si le joueur saisit -1. Rendre la méthode **choisir** robuste en utilisant s'il le faut un bloc try-catch quand la saisie est erronée.

Ecrire le code de la méthode **Joueur gagnant()** méthode qui retourne le joueur qui a le plus grand score,

Ecrire le code de la méthode : **boolean jouerUnTour()** qui se déroule de la façon suivante :

Elle fait choisir les deux joueurs.

Le premier joueur joue en choisissant un label grâce à la méthode **choisir** demandée ci-dessus, et le deuxième joueur qui représente la machine choisit au hasard.

Si la partie continue (choix différent de -1), elle appelle la méthode **void arbitrer(joueur j1, joueur j2)** qui modifie les scores des joueurs et dont le code est demandé à la question suivante.

Elle renvoie un booléen indiquant si la partie continue ou se termine.

Ecrire le code de la méthode : **void jouerUnePartie()** qui appelle **jouerUnTour** tant que l'utilisateur souhaite continuer et affiche l'état du jeu à chaque tour.

[Q3] Proposer une solution permettant une gestion simple des règles d'arbitrage entre les labels pris 2-à-2. Indice, cette solution utilise un tableau à deux dimensions. Ceci afin de permettre d'ajouter des labels dans **Shifu** sans avoir à réécrire le code de la méthode **void arbitrer(Joueur j1, Joueur j2)**. Donner le code de la méthode **arbitrer,** prenant en compte cette solution et donc indépendante des changements de labels.

Annexe :

```
public class Random {
        public int nextInt(int n) {...} // retourne une valeur entre 0 et n-1
        public String next() {…} // retourne une chaîne de caractères
        ...
}
public class ArrayList<T> {
          public int size() {…}; // retourne la taille de la collection
          public boolean add(T x) {…}; // ajoute en fin de la collection
          public void set(int i, T x) {…}; // modifie le ieme élément
          public void get(int i) {…}; // retourne le ieme élément
          public void remove(int i) {…}; // supprime le ieme élément
          public void clear() {…}; // vide la collection
          ...
}
```

Enum :

Créer une énumération simple :

```
public enum NomEnum {
        LABEL1, LABEL2, …, LABELN ;
        // si nécessaire ajouter des méthodes
}
```

Récupérer un label à partir d'un String :

```
NomEnum ne = NomEnum.valueOf(chaineCaracteres) ;
Récupérer le rang d'un label :
int rang = ne.ordinal() ;
```

Parcourir un tableau contenant tous les labels de l'énumération :

```
for (NomLabel nl : NomEnum.values())
        … traitement appliqué à nl …
```

Scanner :

```
Scanner entree = new Scanner(System.in);
// Lecture d'un entier
int i = entree.nextInt();
// Lecture d'une chaîne
String s = entree.next();
// Lecture d'un float
float f = entree.nextFloat() ;
```