Licence Sciences et Technologies – 1ère année
Module Info2B - Interfaces visuelles

CM7
Le jeu de memory

En raison des conditions particulières de travail actuel, et pour permettre d'avancer plus rapidement sur le projet, ce CM7 sera à étudier avant le CM6 (sur les Images et dont les polycopiés ont été donnés le vendredi 13 mars).

Ce CM est rédigé sous forme textuelle, et non en diaporama de transparents puisqu'il ne sera pas présenté directement. Quelques parties sont laissées en travail personnel.

On s'intéresse ici à la mise en place du fonctionnement du jeu du memory tel qu'il est présenté dans le sujet qui vous a été envoyé et qui est disponible sur le site (http://ufrsciencestech.u-bourgogne.fr/licence1/Info2B_InterfacesVisuelles/).

Le jeu de memory proposé comporte des fonctionnalités supplémentaires par rapport aux règles du jeu classique, permettant d'enrichir le jeu.

- Si un joueur réussit à obtenir tous les personnages de sa famille de préférée, le jeu s'arrête.
- Si un joueur réussit à obtenir tous les personnages d'une famille qui n'est pas sa famille préférée, il peut alors, selon la famille complète en sa possession, effectuer une action supplémentaire (récupérer les cartes d'un autre joueur, faire une bataille ou un combat).

- Si le joueur a obtenu tous les personnages de la famille « rares » ou « communs », il peut prendre à un autre joueur, toutes ses cartes d'une famille au choix.
- Si le joueur a obtenu tous les personnages de la famille « legendaires » ou «epiques », un jeu de bataille de cartes entre le joueur courant ~~contre~~ et un autre joueur est exécuté. Cet autre joueur est choisi par le joueur courant.
- Si le joueur a obtenu tous les personnages de la famille « alpins-femmes » ou « as-des-pistes », un combat est engagé entre le joueur courant et tous les autres joueurs.

Un traitement est donc exécuté en fonction de la famille complète de personnages obtenue par ce joueur. Il existe donc trois types d'actions qui peuvent exécuter des traitements : le transfert, la bataille, ou le combat.

Le fonctionnement du jeu de Memory tel qu'il est proposé (gestion des paquets de cartes des joueurs, du plateau, de chaque tour de jeu) se fait par le biais d'un classe « Jeu » qui comporte une méthode importante : la gestion d'un tour de jeu lorsque le joueur courant a retourné deux cartes identiques (c'est-à-dire a gagné un personnage).

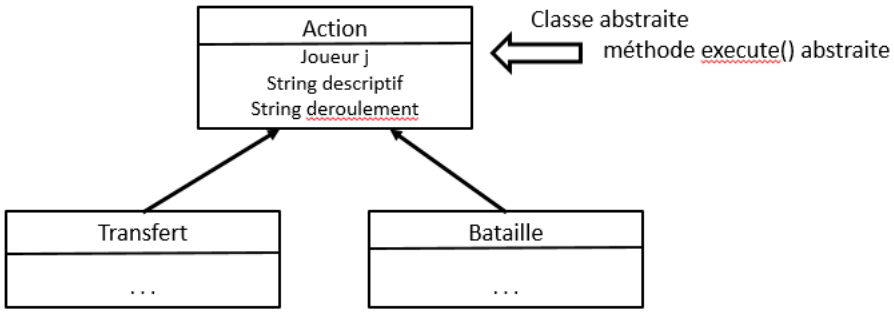
Nous allons nous intéresser dans la suite de ce cours à l'étude des classes représentant les actions, et la classe dédiée à la gestion du jeu.
La gestion des combats étant facultative dans le projet, elle ne sera pas présentée dans ce document.

1. Classe « Action »

Une action concerne un joueur qui est le joueur courant dans le déroulement du jeu. Elle comporte un descriptif qui sera « Transfert de cartes » pour une action de transfert, ou « Bataille » pour une action de type bataille, et également un déroulement qui sera complété selon l'action réalisée pour indiquer comment l'action s'est terminée. Par exemple s'il s'agit d'un transfert de cartes, il indiquera quelles sont

les cartes qui ont été prises et à quel joueur, pour une bataille qui a joué contre qui et ce que le joueur courant a fait (perdu ou gagné).

Pour modéliser ces actions, il a été décidé de mettre en place une hiérarchie de classe comme décrit ci-dessous.



La classe « Action » comporte les attributs communs à l'ensemble des actions « Transfert » et « Bataille » notamment. Elle comporte le joueur concerné, le descriptif et le déroulement. Elle comporte une méthode abstraite « public abstract execute() » qui sera implémentée dans les sous-classes selon l'action.
Cette classe comporte donc :

- les attributs qui représentent les propriétés d'une action (comme décrit ci-dessus),
- un constructeur avec deux paramètres (le descriptif et le joueur) pour créer des objets de ce type dans l'application,
- des accesseurs (get, set) qui permettent de connaître les valeurs des propriétés de l'action,
- une méthode nommée « toString » qui permet d'obtenir toutes les informations sur l'action sous forme d'une chaîne de caractères
- une méthode abstraite « execute() » qui ne prend pas de paramètres mais retourne un entier indiquant comment s'est passée l'action.

Travail personnel : Compléter le code de la classe « Action » décrite ci-dessous.

```
public abstract class Action {
   private Joueur j;  // joueur à l'origine de l'action
   private String descriptif; //nature de l'action
   private String deroulement; // description de ce qui s'est passé durant l'action

   public abstract int execute();

   public Action(Joueur sc, String s) {
      this.j=sc;
      this.descriptif=s;
      this.deroulement="";
   }

// à compléter : méthodes get, et set sur les attributs
   public String toString()
   { return "Action effectuée par "+j.getPseudo()+ " : " +descriptif+"\n"+this.deroulement+"\n";}
}
```

Note : Les sous-classes « Transfert » et « Bataille » seront étudiées dans les sujets de TD.

2. La classe « Jeu »

Les informations à gérer pour réaliser le jeu sont :

- Les personnages utilisés pour les cartes,
- Le plateau de jeu décrit à l'aide d'une instance de la classe « PlateauJeu » qui a été vue au TD4, et qui modélise l'état du jeu,
- Les joueurs qui ont été choisis ou ajoutés pour la partie en cours,
- L'indice du joueur courant,
- L'action que va réaliser le joueur (selon les cartes du joueur qui est en train de jouer).

Ce qui peut se traduire par la définition des attributs suivants :

```
private LesPersonnages lesPers;
private PlateauJeu monP;
private LesJoueurs lesJ;
private int indC;
private Action act;
```

La classe « Jeu » possède un constructeur avec paramètres, comme décrit ci-dessous. Il comporte le paramètre « lp » pour ~~les~~ gérer les personnages, « lj » pour la liste de joueurs, et la valeur « nbc » qui est le niveau de jeu (4 = enfant, 10= débutant, 16=avancé ou 32=expert).

```
public Jeu(LesPersonnages lp, LesJoueurs lj, int nbc)
  {  this.lesPers=lp;
     this.monP=new PlateauJeu(nbc);
     this.lesJ=lj;
     this.act=null;
     this.indC=0;
}
```

Travail personnel : définir l'accesseur en lecture (get) pour chacun des attributs et l'accesseur en écriture (set) pour l'attribut représentant l'indice du joueur courant.

Ces accesseurs sont complétés pour les méthodes ci-dessous.

```
public int getIndSuivant(int j){ return (j+1)%lesJ.getNbJoueurs(); }
public Joueur getJoueurCourant(){return lesJ.getJoueur(indC);}
public Joueur getJoueurSuivant(int j){ return lesJ.getJoueur(getIndSuivant(j)); }
```

Travail personnel : étudier et commenter chacune de ces méthodes.

La classe comporte également une méthode « public boolean finJeu() » qui teste si le jeu est terminé (c'est-à-dire si toutes les cartes sont invalidées) et une méthode « public int nbPers() » qui retourne le nombre total des personnages en possession des autres joueurs (joueurs autres que le joueur courant lui-même). Ces méthodes sont décrites ci-dessous.

```
public boolean finJeu()  { return monP.jeuVide();}
```

```
public int nbPers(){
      int n = 0;
      for(int i = 0; i < lesJ.getNbJoueurs(); i++)
         if (i !=this.indC)   n+=lesJ.getJoueur(i).getNbCartes();
      return n;
}
```

La méthode principale de la classe « Jeu » est la méthode de traitement d'un tour de jeu lorsque le joueur courant a retourné deux cartes identiques, c'est-à-dire a gagné un personnage.
Cette méthode « public int traiterTour(Joueur jo, int s) » prend en paramètre :

- Le joueur courant (qui vient de jouer)
- L'indice du personnage qui a été gagné.

Elle retourne une valeur de « bonus » qui vaut :

- 0 si le joueur a tous les personnages de sa famille préférée
- 1 si un transfert de cartes peut être réalisé
- 2 si une bataille peut être faite
- (3 si c'est un combat)
- -1 dans les autres cas.

L'algorithme général de cette méthode est le suivant :

- Initialisation de la valeur du bonus à -1
- Récupération du personnage gagné nommé « pers »
- Ajout de ce personnage dans le paquet du joueur courant
- Récupération de la famille nommée « f » du personnage gagné « pers »
- Récupération du nombre de personnages de cette famille nommé « npf » dans le jeu (pour l'ensemble des personnages)
- Récupération du nombre de personnages de cette famille nommé « nbj » dans le paquet du joueur courant
- Si le joueur a une famille complète (c'est-à-dire nbf est égal à nbj)
    - Alors
        - Si cette famille est la famille préférée du joueur,
        - Alors
            - le bonus vaut 0,
            - le jeu se termine (appel de la méthode « termineJeu() », du plateau de jeu)
        - Sinon
            - Si les autres joueurs ont des cartes
            - Alors
                - Si la famille gagnée est « rares » ou « communs »
                - Alors le bonus vaut 1
                - Sinon
                    - Si la famille gagnée est « legendaires » ou «epiques »
                    - Alors le bonus vaut 2
                    - (Sinon le bonus vaut 3 // cas du combat)
- Renvoie de la valeur « bonus »

Travail personnel : donner le code correspondant à cette méthode.