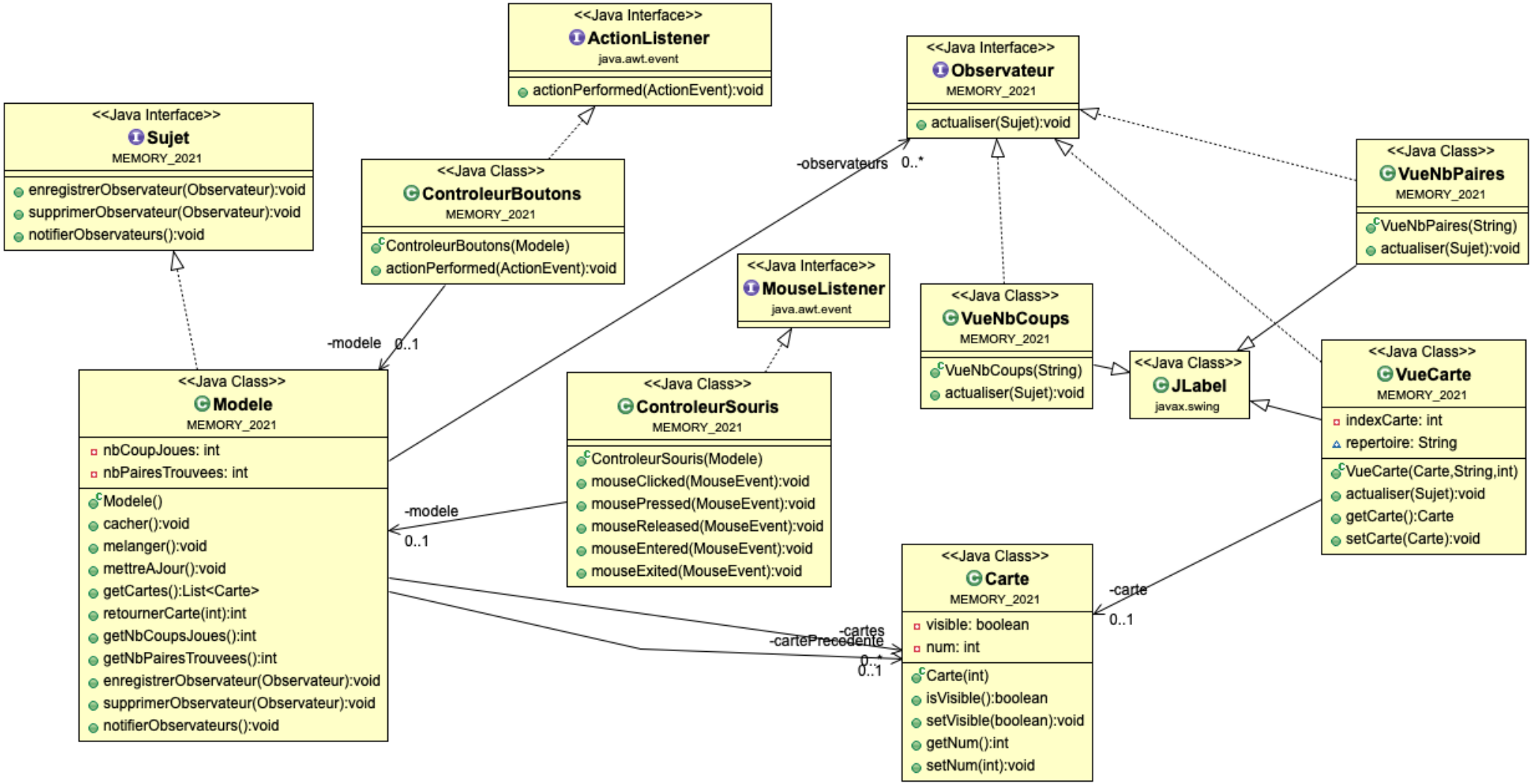
<<Java Interface>>
ActionListener
java.awt.event
actionPerformed(ActionEvent):void
<<Java Interface>>
Observateur
MEMORY_2021
actualiser(Sujet):void
<<Java Interface>>
Sujet
MEMORY_2021
enregistrerObservateur(Observateur):void
supprimerObservateur(Observateur):void
notifierObservateurs():void
<<Java Class>>
ControleurBoutons
MEMORY_2021
ControleurBoutons(Modele)
actionPerformed(ActionEvent):void
-observateurs
0..*
<<Java Class>>
VueNbPaires
MEMORY_2021
VueNbPaires(String)
actualiser(Sujet):void
<<Java Interface>>
MouseListener
java.awt.event
<<Java Class>>
VueNbCoups
MEMORY_2021
VueNbCoups(String)
actualiser(Sujet):void
<<Java Class>>
JLabel
javax.swing
<<Java Class>>
VueCarte
MEMORY_2021
indexCarte: int
repertoire: String
VueCarte(Carte,String,int)
actualiser(Sujet):void
getCarte():Carte
setCarte(Carte):void
-modele
0..1
<<Java Class>>
Modele
MEMORY_2021
nbCoupJoues: int
nbPairesTrouvees: int
Modele()
cacher():void
melanger():void
mettreAJour():void
getCartes():List<Carte>
retournerCarte(int):int
getNbCoupsJoues():int
getNbPairesTrouvees():int
enregistrerObservateur(Observateur):void
supprimerObservateur(Observateur):void
notifierObservateurs():void
<<Java Class>>
ControleurSouris
MEMORY_2021
ControleurSouris(Modele)
mouseClicked(MouseEvent):void
mousePressed(MouseEvent):void
mouseReleased(MouseEvent):void
mouseEntered(MouseEvent):void
mouseExited(MouseEvent):void
-modele
0..1
-cartes
-cartePrecedente
0..*
0..1
<<Java Class>>
Carte
MEMORY_2021
visible: boolean
num: int
Carte(int)
isVisible():boolean
setVisible(boolean):void
getNum():int
setNum(int):void
-carte
0..1

REALISATION DU JEU

PARTIE 1 – MVC ET PATRON DE CONCEPTION STRATEGIE

Le code de la construction de l'interface graphique se trouve dans la classe `Principale` dans le fichier ***tp_note_2021.zip***. La classe `Carte` est aussi fournie dans ce fichier, elle permet de modéliser les cartes associées au jeu de memory. Les deux séries de 11 images sont dans un répertoire ***images*** à déposer à la racine de votre projet Eclipse.
Les `JPanel` sont organisés selon la description indiquée sur l'image ci-dessous :

JPanel plateau

Ce `JPanel`, situé au nord et organisé en GridLayout(4,5), contient 20 JLabels, **numérotées de 0 à 19**, et correspondant aux positions des cartes du jeu de memory.

`JPanel` panneauScore : + en `GridLayout(2,1)`

`JPanel` panneauChoix

1. En tenant compte du **diagramme distribué** après la phase de conception, modifier le code fourni de la classe `Principale` afin de programmer un jeu de memory selon l'architecture MVC proposée sur le diagramme et en respectant le fonctionnement du jeu comme il est décrit sur la première page de l'énoncé. Vous travaillerez dans un package nommé **`memory_tp_note`**. La mise en place du MVC sera réalisée dans cette question **pour une seule série d'images.**

   - Commencer par coder la classe **`Modele`** puis inclure l'instanciation d'un objet de la classe `Model` dans la classe **`Principale`** :

     - **Attributs de `Modele`** :

       - **`cartes`** est une liste d'objets **`Carte`**. La position d'un objet **`Carte`** dans la liste correspond à sa position sur le plateau.

         - Classe **`Carte`** : L'attribut **`num`** d'un objet **`Carte`** correspond au numéro du fichier image associé à la carte. L'attribut **`visible`** d'un objet **`Carte`** a pour valeur `true` si l'image de la carte est visible et `false` sinon. La valeur par défaut au moment de la création d'un objet **`Carte`** est `false`.

- **`cartePrecedente`** est un objet Carte, gardant en mémoire la première carte cliquée pendant un tour du jeu ou 2 cartes sont retournées.
- **`nbPairesTrouvees`** est le nombre de paires d'images trouvées.
- **`nbCoupsJoues`** est le nombre de coups joués c'est-à-dire le nombre de fois où deux images ont été dévoilées.

▪ La méthode **`melanger()`** permet de *battre les cartes*, c'est à dire de les mélanger dans la liste **`cartes`**.

▪ La méthode **`cacher()`** parcourt la liste de cartes et positionne l'attribut **`visible`** de chaque carte à `false`. Tous les attributs de **`Modele`** doivent être réinitialisés.

▪ La méthode **`int retrournerCarte(int i)`** , traite le retournement d'une carte (nommée carte courante) pendant un coup joué, après un clic sur cette carte : l'entier i correspond à l'indice de la carte dans la liste de cartes. Cette carte devient alors visible.

Si la **`cartePrecedente`** est différente de null, l'attribut **`num`** des deux cartes est comparé,
- Si ils sont égaux, une paire de cartes identiques est trouvée, elles restent visibles. La **`cartePrecedente`** doit passer à **`null`**. Si 10 paires ont été trouvées, la partie s'arrête et un message spécifique est affiché.
- Sinon, les cartes ne doivent plus être visibles (attention, elles ne doivent disparaître qu'au prochain coup)

Sinon, la **`cartePrecedente`** devient la carte courante

Attention à la gestion de la mise à jour des affichages au cours de cette méthode.

- **Coder les vues :**

▪ **`VueNbPaires`** et **`VueNbCoups`** sont respectivement les JLabel dans lesquels le nombre de paires trouvées et le nombre de coups joués apparaissent.

▪ **`VueCarte`** est un JLabel associé à une carte du jeu, le plateau est donc constitué de 20 objets **`VueCarte`**, chacun associé à une carte du jeu. Il permet donc de visualiser sont état : si l'attribut **`visible`** de la carte est `true`, alors l'image associée à la carte est affichée dans le JLabel, sinon, l'image *fond.png* est affichée dans le JLabel. Au moment de la création des cartes, une instance de **`VueCarte`** doit être créée pour chacune des cartes. Les paramètres associés au constructeur de la classe **`VueCarte`** sont : la carte associée au JLabel, le répertoire où trouver le fichier image, l'indice de la carte dans la liste de cartes. L'affichage d'une image dans un JLabel se fait avec la méthode **`setIcon`**.

- **Coder les contrôleurs**, c'est-à-dire :
  ▪ La classe **`ControlerBoutons`**, et l'associer aux composants graphiques concernés (boutons du JPanel PanneauChoix). Les boutons fruits et smileys ne seront activés que dans la partie 2.

- La classe **`ControlerSouris`**, ce contrôleur est à l'écoute des événements souris sur les JLabel VueCarte du JPanel plateau ; la carte associée au JLabel *cliqué* doit alors être retournée.

A la fin de cette question, un utilisateur doit pouvoir jouer avec une série d'images fixée, le nombre de coups joués et le nombre de paires trouvées doivent s'afficher et les boutons cacher et mélanger doivent aussi fonctionner.

2. Générer le diagramme des classes de votre programme, vérifier sa conformité par rapport au diagramme distribué, faites ressortir les éléments caractéristiques du MVC. Enregistrer ce diagramme dans un *fichier png*.

3. Commenter les classes afin de pouvoir générer la documentation (javadoc) du programme. Plus précisément et **en priorité**, le commentaire associé à la classe `Modele` devra décrire le mécanisme du **patron de conception Observateur** mis en œuvre dans votre programme. Le commentaire associé aux classes contrôleurs devra décrire le mécanisme du **patron d'architecture MVC** mis en œuvre dans votre programme.

Enregistrer l'ensemble des fichiers réalisés pour les questions 1 à 3 dans un fichier nommé *votreGroupe_VotreNom_PARTIE1.zip*

PARTIE 2

Dans cette partie, on fait évoluer le jeu **tout en restant dans une architecture MVC**. Ajouter à votre programme les fonctionnalités manquantes :

- Changement des images du mémory après appui sur les boutons fruits ou smiley.
- Fin du jeu si toutes les paires ont été retournées ou si le nombre de coups maximal a été joué.

Vous utiliserez pour cela **au moins un patron de conception** de votre choix et expliquerez son rôle dans les commentaires de votre code.

QUESTIONS BONUS :

1. Dans le JPanel panneauChoix , un composant JTextField est ajouté permettant de mettre un nombre (de 0 à 19) pour indiquer la carte à retourner et donc permettant de jouer sans utiliser la souris. Modifier votre code source en respectant le patron MVC.
2. En plus des images, on voudrait pouvoir aussi utiliser des chaines de caractères dans les JLabels (par exemple des paires de nombres ou encore des paires constituées d'une addition et de son résultat). Les choix possibles seront proposés dans un composant graphique JComboBox avec smileys, fruits et nombres. Commenter les classes modifiées et/ou ajoutées pour expliquer votre conception.

Enregistrer l'ensemble des fichiers réalisés pour cette question dans un fichier nommé *votreGroupe_VotreNom_PARTIE2.zip*

Déposer les archives sur arche.