IUT nancy Charlemagne Département Informatique

Conception et programmation objet avancées
2014-2015

TP Noté

Mardi 21 octobre 2014

Durée 3h30 – sortie non autorisée avant 2h –
Documents autorisés après 30 minutes

JEU DE PENDU GRAPHIQUE SELON L'ARCHITECTURE MVC

Un premier document papier est à rendre au bout de 30 minutes, pendant cette période les ordinateurs ne seront pas utilisés. Le reste du devoir sera réalisé sur machine et l'ensemble des fichiers devra être déposé sur arche dans un fichier archive.

L'objectif de cet exercice est de programmer un jeu de pendu graphique selon l'architecture MVC dont l'interface graphique sera similaire à celle présentée ci-dessous.

DEROULEMENT DU JEU :

L'utilisateur doit deviner un mot en maximum 9 essais avec échec. Pour cela, il choisit les lettres en les sélectionnant avec les **boutons** présents dans l'interface.

- **Si la lettre choisie appartient au mot à découvrir**, la lettre apparaît à la bonne place (ou aux bonnes places si elle est plusieurs fois dans le mot) dans la zone **JLabel** correspondant au mot en cours de découverte. De plus, **si le mot est entièrement découvert** alors le texte de la zone **JLabel** contenant *Mot à découvrir* est remplacé par le texte *Bravo !*, le jeu est terminé et l'appui sur un autre bouton ne doit plus avoir d'effet.
- **Si la lettre choisie n'appartient pas au mot à découvrir**, l'image de la potence (placée dans un **JPanel**) correspondant au nombre de mauvaises lettres tentées, est modifiée dans la partie droite de l'interface (si *pendu2.jpg* était visible avant l'essai, *pendu3.jpg* sera visible après). De plus, **si le nombre d'essais avec échec est de 10**, le texte *Mot à découvrir* est remplacé par le texte *Perdu !*, le jeu est terminé et l'appui sur un autre bouton ne doit plus avoir d'effet.

Dans les deux cas, au cours du jeu, le bouton de la lettre sélectionnée par l'utilisateur n'est plus cliquable.

NOM : Prénom :

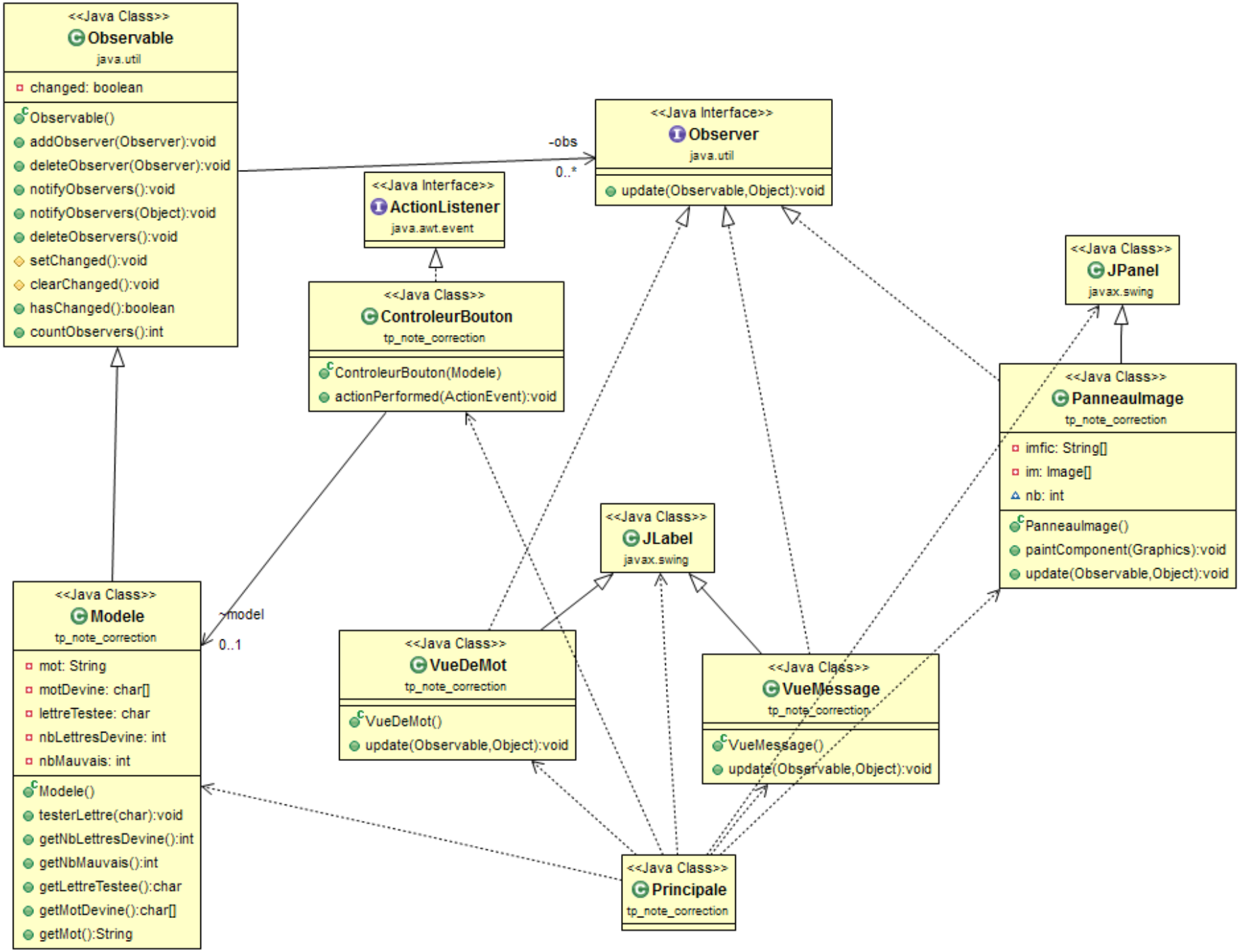
1. MODELE VUE CONTROLEUR - CONCEPTION – Les explications demandées dans les questions 1 et 2 ci-dessous sont à rendre 30 minutes après la distribution du sujet.

1.1. Dans le cadre de l'implantation du modèle MVC pour l'interface graphique présentée à la page précédente, identifier et décrire précisément :

- le modèle (donner les attributs, les méthodes envisagées en précisant leurs rôles),
- les vues (indiquez la nature de leur composant graphique, le(s) méthode(s) envisagée(s) en précisant leur(s) rôle(s)),
- le contrôleur (indiquer son interaction avec certains composants de l'interface graphique, ses relations avec le modèle, les méthodes envisagées, …).

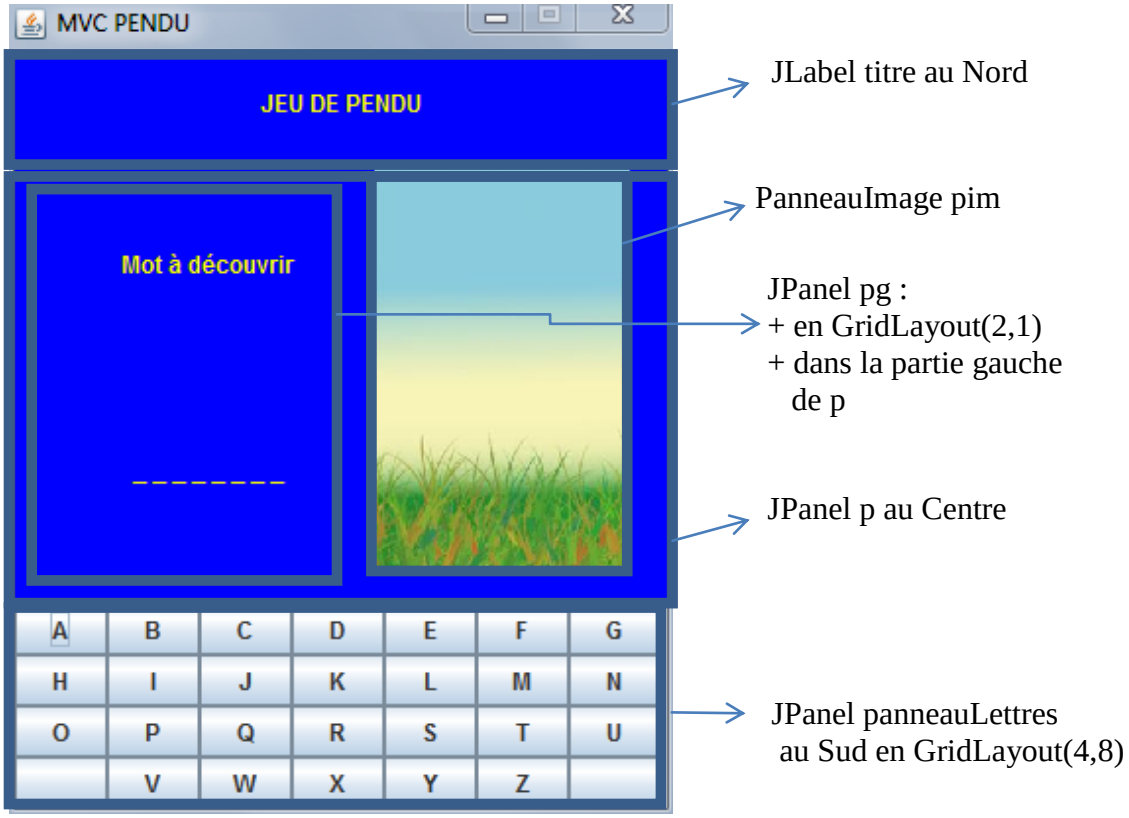
1.2. Faire SUR PAPIER le diagramme des classes java que vous envisagez pour la mise en œuvre du modèle MVC en précisant les liens d'**héritage**, d'**implémentation**, et d'**associations**.

----Rendez ce document papier avant de passer à la programmation----


<<Java Class>>
Observable
java.util
changed: boolean
Observable()
addObserver(Observer):void
deleteObserver(Observer):void
notifyObservers():void
notifyObservers(Object):void
deleteObservers():void
setChanged():void
clearChanged():void
hasChanged():boolean
countObservers():int
-obs
0..*
<<Java Interface>>
Observer
java.util
update(Observable,Object):void
<<Java Interface>>
ActionListener
java.awt.event
<<Java Class>>
ControleurBouton
tp_note_correction
ControleurBouton(Modele)
actionPerformed(ActionEvent):void
<<Java Class>>
JPanel
javax.swing
<<Java Class>>
PanneauImage
tp_note_correction
imfic: String[]
im: Image[]
nb: int
PanneauImage()
paintComponent(Graphics):void
update(Observable,Object):void
<<Java Class>>
JLabel
javax.swing
<<Java Class>>
Modele
tp_note_correction
mot: String
motDevine: char[]
lettreTestee: char
nbLettresDevine: int
nbMauvais: int
Modele()
testerLettre(char):void
getNbLettresDevine():int
getNbMauvais():int
getLettreTestee():char
getMotDevine():char[]
getMot():String
-model
0..1
<<Java Class>>
VueDeMot
tp_note_correction
VueDeMot()
update(Observable,Object):void
<<Java Class>>
VueMessage
tp_note_correction
VueMessage()
update(Observable,Object):void
<<Java Class>>
Principale
tp_note_correction

2. REALISATION DU JEU

Le code de la construction de l'interface graphique se trouve dans la classe Principale qui est fournie dans le fichier ***tp_note_2014.zip***. Les 10 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 :



2.1. En tenant compte du **diagramme distribué** après la phase de conception de 30 minutes, modifier le code fourni de la classe Principale afin de programmer un jeu de pendu 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. **Pour cette première partie de la réalisation, le mot à découvrir sera fixé et sera BONJOUR.**

- Commencer par coder la classe Modele puis inclure l'instanciation d'un objet de la classe Modele dans la classe Principale :
  - Attributs de Modele : *mot* est le mot à deviner, *motDevine* est le tableau des caractères associé au mot en cours de découverte, si une lettre a été trouvée, elle est présente à (ou aux) la bonne place dans *motDevine* sinon il y a le caractère _. *lettreTestee* est la lettre en cours de test. *nbLettreDevine* et *nbMauvais* sont respectivement le nombre de lettres du mot devinées et le nombre de mauvaises lettres proposées.
  - La méthode *testerLettre* modifie le modèle en fonction de la lettre proposée par l'utilisateur.
- Coder les classes vues en modifiant en conséquence la classe Principale
- Coder le contrôleur, c'est-à-dire la classe ControleurBouton, et l'associer aux composants concernés.

2.2. Générer le diagramme des classes de votre programme, vérifier sa conformité par rapport au diagramme distribué, faites ressortir les éléments caractéristiques du MVC.

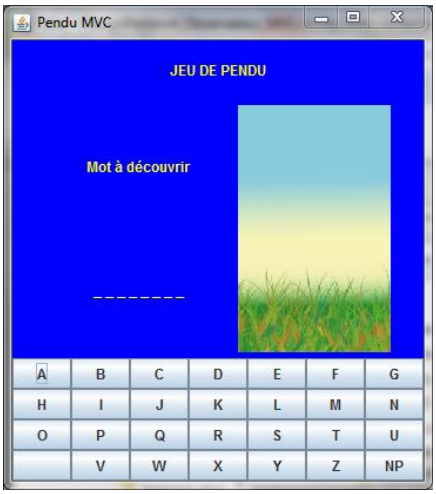
2.3. Commenter les classes afin de pouvoir générer la documentation (javadoc) du programme.
Plus précisément, le commentaire associé à la classe Modele devra décrire le mécanisme du **patron de conception Observateur** mis en œuvre dans votre programme. Le commentaire associé à la classe ControleurBouton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

Enregistrer l'ensemble des fichiers réalisés pour les questions 2.1 à 2.3 dans un fichier nommé *votreGroupe_VotreNom_Partie1.zip*

2.4. Dans cette question, on fait évoluer le jeu **tout en restant dans une architecture MVC** :

- La lecture du mot à deviner se fait aléatoirement dans le fichier texte de mots *listeMots.txt*
- A la fin de la partie, en appuyant sur le bouton en bas à droite sur lequel vous mettrez la chaîne *NP*, une nouvelle partie est réinitialisée (nouveau mot à deviner, boutons réactivés, …).

Pour cela, vous modifierez la classe Modele et vous pourrez aussi ajouter une classe VueBouton permettant de gérer simplement la désactivation et réactivation de chaque bouton.



Enregistrer l'ensemble des fichiers réalisés pour cette question dans un fichier nommé *votreGroupe_VotreNom_Partie2.zip*

Déposer les archives sur arche.
