Rapport de TP

Complément JAVA

YALCINTEPE Ersagun, VERHOOF Tom

❖ **Tp 1 –** Classe Réseau

Nous avons crée la classe réseau. Il fallait respecter les noms des méthodes et classes, afin que la classe Affiche fournie sur l'ENT puisse fonctionner correctement.

❖ **Tp 2 –** Débugger et Reseau (clonage)

Dans un premier temps, l'utilisation du débuggeur ne semble pas si facile. La numérotation de la carte est disponible dans le dossier (Voir reseau.jpeg).

❖ **Tp 3 –** Gestion des packages (scripts de compilation)

Les packages nous permettent de clarifier la structure du projet. Les scripts de compilation devaient posséder les chemins relatifs corrects (libre, src, bin,…). Ce travail a nécessité quelques essais pour obtenir le résultat souhaité (compil.bat, execut.bat).

❖ **Tp 4 –** Dijkstra (assertions)

Notre première approche a été faite avec l'algorithme de Dijkstra. Ce dernier a pour objectif d'effectuer une recherche de chemin plus court. Dans ce TD, nous avons vu la différence entre les exceptions et les assertions.

❖ **Tp 5 –** Dijkstra (tests unitaires)

Dans ce TD, nous avons appris à utiliser Junit. Nous avons crée des classes de test pour les classes Chemin et Dijkstra, puis vérifié leurs méthodes. Ces classes de tests ont vérifié les différents problèmes probables, liés aux utilisateurs. Les assertions, quant à elles, ont pour objectif de vérifier les erreurs dues au programmeur.

❖ **Tp 8** – MVC réseau (interface graphique, contrôleur, ...)

Dans ce TD, nous avons appris à utiliser MVC (Modèle Vue Contrôleur) en implémentant la classe Observeur et en héritant de la classe Observable. Les difficultés dans ce travail étaient de séparer les modèles Vue et Contrôleur.

Les differents problèmes rencontrés

Nous avons rencontré de nombreuses difficultés a comprendre ce que l'on devait faire dans les TPs. Même en faisant appel au professeur, celui-ci nous disait qu'il ne comprenait pas ce qu'il fallait faire, et qui fallait simplement sauter la question. J'ai également eu des problèmes pour travailler avec l'environnement Eclipse : impossible de l'installer sur mon ordinateur portable (et ceux même apres avoir consulté l'aide sur le Web) et plantage automatique des que l'on veut utiliser le JUnit sur les applications Eclipse installées sur les ordinateurs a l'IUT.

Fonctionnement

Pour MVC on a eu besoin de la classe :

- Reseau hérité de la classe Observable
- ReseauSelection hérité de la classe Observable
- VueSelection hérité de la classe VueReseau et qu'on a implémenté la classe Observer
- Controleur
- ControleurSelection hérité de MouseInputAdapter
- TestVue (la classe main)
- VueReseau herité de JPanel et qu'on a implementé la classe Observer

ReseauSelection

Dans la classe ReseauSelection.java il y a un reseau.On manipule cette reseau.ReseauSelection est le modèle, donc il contient toute les methodes qui vont modifier la classe reseau.Dans cette classe on selectionne, deselectionne et on deplace un nœud.A chaque changement on apelle les methodes ; this.setChanged(),this.notifyObservers() qui va tenir au courant la classe vue.

VueSelection

La classe vue est VueSelectionne.java. Elle permet d'afficher le reseauSelectionné.Elle contient une methode update qui met a jour l'objet observé.

ControleurSelection

La classe ControleurSelection.java permet de gerer les actions de souris a l'aide des methodes de la classe MouseInputAdapter.Elle apelle les methodes de la classe reseauSelectionné pendant les manipulations de souris.

VueReseau

La classe VueReseau.java contient la methode paint qui affiche les nœuds de l'attribut reseau.Elle contient aussi une methode update qui met a jour l'attribut reseau en cas de changement.

Controleur

La classe Controleur.java a pour l'objectif de controler un objet VueReseau, deplacer un nœud mais elle s'occupe egalement de charger une fenetre.

Reseau

La classe Reseau.java a pour l'objectif d'avoir une methode deplacer qu'il va deplacer les nœuds.Cette classe est ecouté par VueReseau.

VueSelection observe la classe ReseauSelection : rs.addObserver(vs);

On fait une premiere affichage : rs.permierDessin();

On ajoute au VueSelection les ControleurSelection

vs.addMouseListener(cs);

vs.addMouseMotionListener(cs);

VueReseau observe la classe Reseau : r.addObserver(vr);

On fait une premiere affichage : r.permierDessin();

Les commandes utilisées

Execute_MVC .bat execute la classe TestVue.java qui est la classe main de MVC et qui affiche deux reseaux dont un qui permettre de commander.

java -classpath bin TestVue

Execute_TestClone.bat à pour l'objectif de cloner un reseau. A partir d'un reseau on créer deux réseaux identiques et on les affiche. La classe main de clonage est appéllé TestClone.java.

java -classpath bin TestClone

Pour créer des points jar a partir des package on a utilisé la commande ;

jar -cf carte.jar bin/carte/*.class

compile.bat compile toute les classes de TP. Pour compiler toutes les classes on a utilisé la commande ;

javac -sourcepath src -classpath bin;
lib/affiche.jar;lib/TestVue.jar;lib/Principale.jar;lib/algo.jar;lib/carte.jar;lib/vue.jar;lib/controle.jar;lib/test.jar src/*.java src/vue/*.java src/algo/*.java src/carte/*.java src/vue/*.java src/controle/*.java src/test/*.java -d bin