le cnam

Sujet UE NFA035 : Programmation Java : bibliothèques et patterns

Année universitaire 2015 – 2016
Examen 2e session : 6/9/2016
Responsable : Serge ROSMORDUC

Durée : 3 heures

Tout document *papier* autorisé. Tout support électronique est interdit : pas d'ordinateur, de tablette, de liseuse…

Les téléphones mobiles et autres équipements communicants doivent être éteints et rangés dans les sacs pendant toute la durée de l'épreuve.

Le barême est donné à titre indicatif ; il est susceptible de modifications.

Sujet de 7 pages, celle-ci comprise.

Exercice 1 *10 points (les collections)*

On souhaite gérer l'installation de logiciels sur une machine. Un logiciel est décrit par un nom, un numéro de version et par l'ensemble des logiciels dont il dépend : ils sont nécessaires à son installation/utilisation. Exemple : Netbeans a besoin de JDK pour être installé/utilisé. Donc, Netbeans dépend de JDK. A la création, l'ensemble de dépendances d'un logiciel est vide. Elles sont ajoutées une à une via la méthode `ajoutDependance`. Un objet de la classe `InstallMachine` contiendra tous les logiciels installés sur la machine. On se donne les classes suivantes :

```
public class Logiciel {
   private String nom;
   private int version;
   // INVARIANT: dependances ne contient pas de doublons.
   private Set<Logiciel> dependances = new HashSet<Logiciel>();

   public Logiciel(String n, int v) {
        this.nom=n; this.version=v;
   }
   public String getNom(){
        return nom;
   }
   public int getVersion(){
        return version;
   }
   public boolean dependDe(Logiciel p){
        return dependances.contains(p);
   }
   public boolean ajoutDependance(Logiciel l){
       boolean added = dependances.add(l);
       return added;
   }
   public Set<Logiciel> getDependances(){
        return new HashSet<Logiciel>(dependances);
  }
 }

public class InstallMachine {
   // NOMS + LOGIELS INSTALLES SUR LA MACHINE
   private HashMap<String,Logiciel> installes= new HashMap<String,Logiciel>();

   /**
    * Dit si le logiciel nommé nl est installé.
    * @param nl un nom de logiciel
    * @return true si et seulement si
    *   une version de nl est installée.
    */
   public boolean estInstalle(String nl){
      return installes.containsKey(nl);
   }
   public boolean estInstalle(Logiciel l){
      ... // A COMPLETER
```

```
    }
    /**
     * Enregistre un logiciel comme installé,
     * si c'est possible.
     * un logiciel peut être installé s'il ne l'est pas déjà,
     * et si ses dépendances sont installées, avec les bonnes
     * versions.
     * post-condition: si le logiciel peut être
     * installé, il est enregistré dans la map.
     * @return true si le logiciel a pu être installé.
     */
    public boolean installer(Logiciel l){
        if (estInstalle(l)){
            return false;
    } ... // COMPLETER
}
```

Note importante : Sauf mention du contraire, on suppose dans toutes les questions que le code donné est complet et se comporte correctement, et en particulier, que les invariants sont respectés.

Note importante 2 : Supposons que Netbeans dépende de JDK et que JDK dépende de `gcc`. On dit que Netbeans dépend *directement* de JDK et *indirectement* de gcc. Dans tout cet exercice **on ne travaille que sur les dépendances directes** : celles figurant explicitement dans la liste de dépendances d'un logiciel.

Question 1.1 *(1 point)*

Dans la classe `InstallMachine` on vous demande de compléter le code de la méthode `estInstalle(Logiciel l)`. Cette méthode doit tester non seulement le nom du logiciel mais aussi son numéro de version. Par exemple, si JDK 6 est installé, un appel qui teste si JDK 8 est installé doit renvoyer false.

Question 1.2 *(2 points)*

Dans `InstallMachine`, ajoutez la méthode `depNonInstallees(Logiciel p)` qui retourne un `Set<Logiciel>` correspondant aux logiciels se trouvant dans la liste de dépendances de p **et qui par ailleurs ne sont pas installés** sur la machine. Supposons que k soit dans les dépendances de p : s'il est déjà installé (avec la version qui convient), k ne doit pas figurer dans le résultat. Si k n'est pas installé, ou si k est installé mais dans une autre version, il doit figurer au résultat.

Question 1.3 *(1.5 points)*

Dans la classe `InstallMachine` complétez la méthode `installer(Logiciel l)`. Note : un logiciel pourra être installé uniquement si ce logiciel (même nom, même numéro de version) n'est pas déjà installé, et si toutes ses dépendances sont déjà installées. Dans les autres cas, le logiciel ne sera pas ajouté aux logiciels installés de la machine et la méthode renverra false.

Question 1.4 *(2 points)*

Complétez les 4 tests JUNIT suivants, qui doivent permetre de vérifier le bon comportement de la méthode `installer`. Vos tests doivent correspondre à des cas différents de cette méthode. Vous sup-

poserez que les déclarations suivantes figurent dans votre fichier de tests :

```
public class LogicielTest {
        public static Logiciel jdk6 = new Logiciel("JDK",6);
        public static Logiciel jdk8 = new Logiciel("JDK",8);
        public static Logiciel netbeans = new Logiciel("netbeans",4);
        public static Logiciel x11 = new Logiciel("X11",7);


    @BeforeClass  // CECI s'execute avant de commencer l'ensemble des tests
    public static void avantTous() {
        netbeans.ajoutDependance(jdk6);
        netbeans.ajoutDependance(x11);
    }
    @Test
     public void test5() {
        InstallMachine ins = new InstallMachine();
        boolean res = ins.installer(x11);
        // completer...
     }
     @Test
      public void test6() {
        InstallMachine ins = new InstallMachine();
        boolean res = ins.installer(netbeans);
        // completer...
     }
     @Test
      public void test7() {
        InstallMachine ins = new InstallMachine();
        // completer
      }
      @Test
      public void test8() {
        InstallMachine ins = new InstallMachine();
        // completer
      }
}
```

Question 1.5 *(1.5 points)*

Dans cette question, on n'est pas certain du respect de l'invariant. Pour chaque test JUNIT ci-dessous, indiquez s'il réussit ou s'il échoue, et expliquez la raison de la réussite ou de l'échec, en une ligne maximum. **Attention : tout réponse non justifié ne sera pas notée**.

```
@Test
 public void test2() {
        Logiciel a = new Logiciel("A", 0);
        Logiciel b0 = new Logiciel("B", 0);
        Logiciel b00 = new Logiciel("B", 0);
        assertTrue(a.ajoutDependance(b0));
        assertFalse(a.ajoutDependance(b00));
}
```

```
@Test
 public void test3() {
     Logiciel a = new Logiciel("A", 0);
     Logiciel b0 = new Logiciel("B", 0);
     Logiciel b1 = new Logiciel("B", 1);
     a.ajoutDependance(b0);
     a.ajoutDependance(b1);
     assertTrue(a.dependDe(b0));
     assertTrue(a.dependDe(b1));
}
@Test
 public void test4() {
    Logiciel a = new Logiciel("A", 0);
    Logiciel b0 = new Logiciel("B", 0);
    Logiciel b00 = new Logiciel("B", 0);
    a.ajoutDependance(b0);
    assertTrue(a.dependDe(b00));
}
```

Question 1.6 *(2 points)*

Si d'après vous, le code des classes ne respecte pas l'invariant, expliquez ce qu'il faut faire pour résoudre le problème (le code exact n'est pas demandé). Si au contraire, vous pensez qu'il le respecte, justifiez votre affirmation (2 lignes maximum).

Exercice 2 *Entrées/Sorties (7 points)*

Question 2.1 *3 points*

Écrire le code de la procédure suivante :

```
public static void copierLignes(Reader r, Writer w,
        int pos, int nombreLignes)
  throws IOException{
}
```

La fonction doit recopier sur w les lignes de texte lues dans r, en copiant `nombreLignes` lignes, en commençant à la ligne `pos`.

- la numérotation des lignes commence à 0 ;
- Formellement, on doit copier toute ligne dont le numéro est compris entre pos (inclus) et pos+ nombreLignes (exclu).

Donc, si pos=0 et nombreLignes=10, la procédure copiera les 10 premières lignes lues.

Si le flux lu sur reader n'est pas assez long, par exemple, s'il comporte 6 lignes en tout et qu'on a demandé pos=4 et nombreLignes=8), ça ne sera pas considéré comme une erreur. Dans ce cas précis, on se contentera de copier les lignes 4 et 5 (comme la numérotation commence à 0, il n'y a pas de ligne 6). De même, si pos est supérieur ou égal au nombre de lignes lisibles sur r, w sera le fichier vide.

Question 2.2 *4 points*

Dans un logiciel de blog, on a décidé que les rédacteurs pourraient taper des fichiers textes simplifiés au lieu de html. Dans le format qu'on a défini, un lien entre une page et un site web est donnée de la manière suivante :

```
[ LABEL DU LIEN | URL DU LIEN ]
```

Comme par exemple :

```
voir le [cours de NFA035|http://www.cnam.fr/nfa035]
pour plus de détails
```

Le label du lien et son URL sont considérés comme du texte quelconque, avec comme seule condition que le label ne peut pas contenir le caractère '|' et que l'URL n'a pas le droit de contenir le caractère ']'.

On considère la classe java (que vous n'avez pas à écrire).

```
class Lien {
  private String label, url;
  public Lien(String label, String url) {...}
  public String getLabel() {...}
  public String getUrl() {...}
}
```

Écrivez la méthode

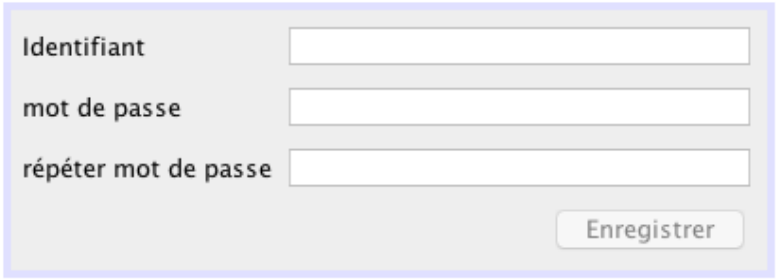
```
public static List<Lien> listerLiens(Reader r) throws IOException {
...
}
```

Qui crée et retourne la liste des liens trouvés dans le flux lu par `r`.

Vous supposerez que le fichier est correctement écrit (on ne vous demande pas de gérer les erreurs).

Exercice 3 *Swing (3 points)*

On considère l'interface graphique, qui est supposée servir à sauvegarder les informations d'un nouvel utilisateur.



Comme vous le voyez, le bouton pour enregistrer le nouvel utilisateur est désactivé (on l'a configuré avec `bouton.setEnabled(false)`. Pour le réactiver, il faut appeler `bouton.setEnabled(true)`.

On vous demande d'implémenter le comportement suivant :

— le bouton ne doit être activé **que** si l'identifiant et les mot de passe ne sont pas vides, et que d'autre part, les deux mots de passe sont égaux (pour simplifier, on considèrera que les deux mots de passe sont édités par des `JTextField`). L'activation (ou la désactivation) du bouton doit se faire en cours de frappe : dès que le contenu des champs texte est correct, le bouton doit s'activer.

— quand on presse le bouton, il doit appeler la méthode `sauverUtilisateur()`, qu'on ne vous demande pas d'écrire.

Complétez le code suivant, en écrivant la méthode `activer()` (et éventuellement les classes et méthodes auxiliaires dont vous avez besoin)

Vous pouvez ajouter des méthodes à la classe.

```
public class CreateurUtilisateur {
        private JTextField identifiantField= new JTextField(10);
        private JTextField motDePasse1Field= new JTextField(10);
        private JTextField motDePasse2Field= new JTextField(10);
        private JButton bouton= new JButton("Enregistrer");

        public CreateurUtilisateur() {
                mettreEnPage();
                activer();
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setVisible(true);
        }

        private void mettreEnPage() {
          // ne pas écrire...
        }

        private void activer() {
                // met en place les divers listeners...
                // À ÉCRIRE !!!

        }

        /**
         * méthode auxiliaire que vous pouvez utiliser.
         *
         * la méthode est déjà écrite, on ne vous demande
         * pas de l'écrire
         */
        public void sauverUtilisateur(String identifiant, String motDePasse) {
                // NE PAS ÉCRIRE...
        }

}
```