Rapport de Génie Logiciel

Titouan Loiseau et Baptiste Marchand
4A SAGI - TD2

Sommaire

Introduction

1 Exercice 1

1.1 Q1

La figure 1 présente le premier diagramme de classes imaginé.

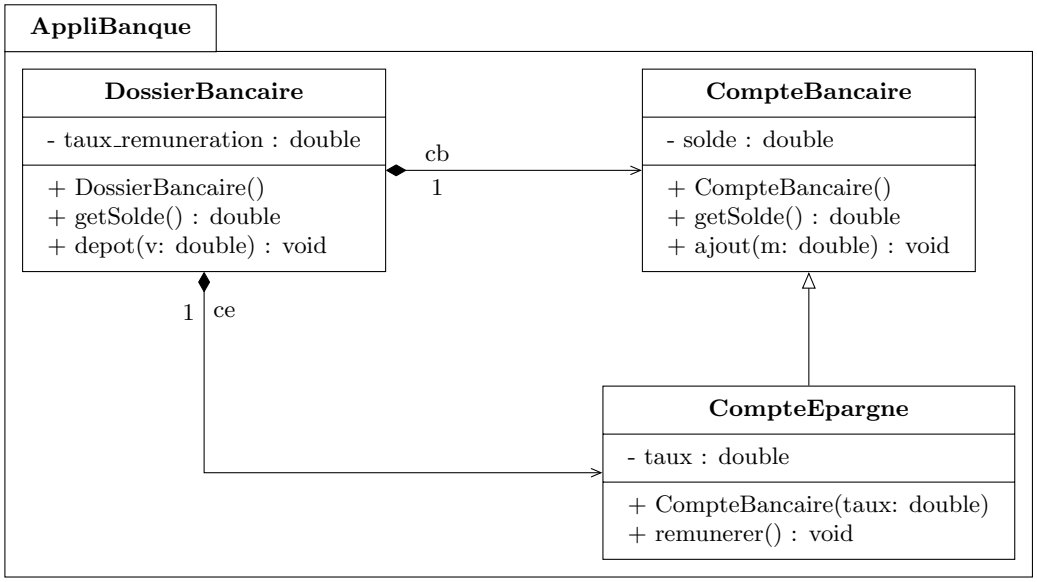

Figure 1: Diagramme de classes

1.2 Q2

La figure 2 présente le premier diagramme de classes imaginé.

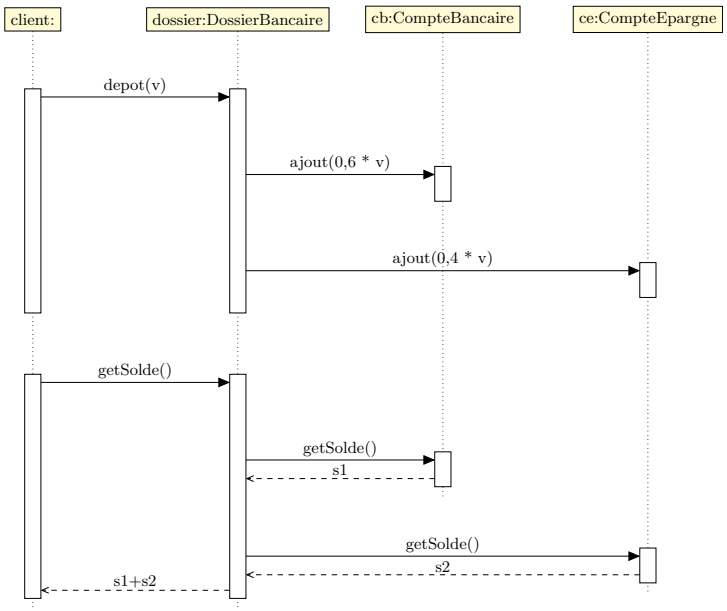

Figure 2: Diagramme de séquence

1.3 Q3

En plus des diagrammes précédents, on peut faire un diragramme d'objets modélisant l'état du programme après l'instanciation de l'objet de la classe DossierBancaire. La figure 3 présente un tel diagramme.

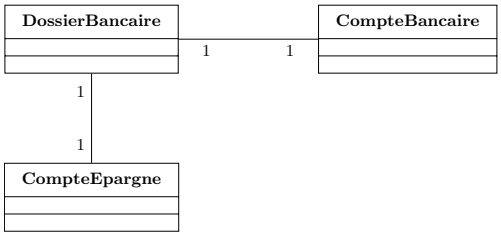

Figure 3: Diagramme d'objets

2 Exercice 2

Pour la compilation du projet simplement, les commandes trouvées dans le README permettent de compiler le projet, comme montré dans la figure 4.

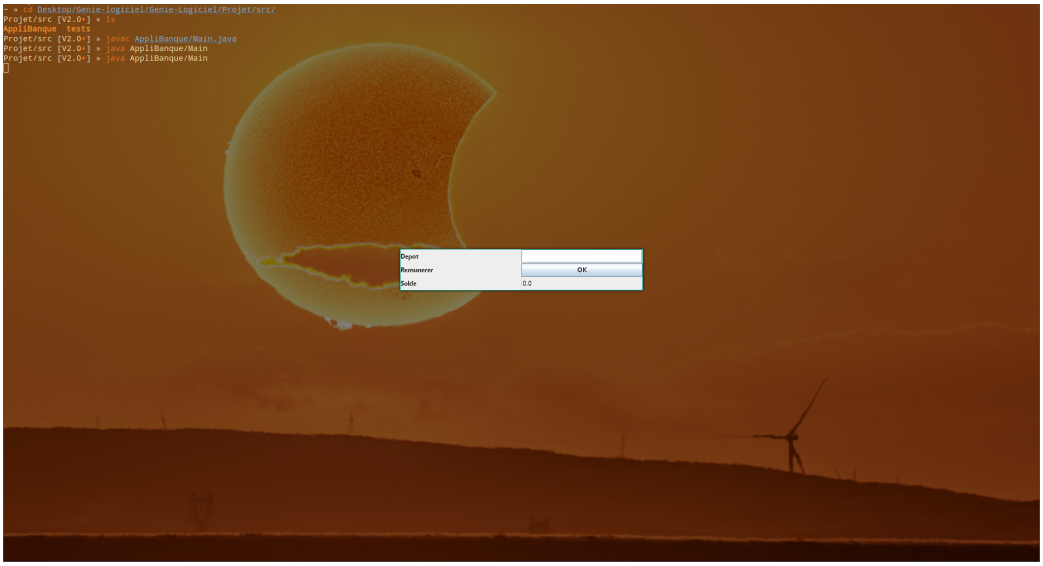

Figure 4: Compilation du programme

Pour la compilation des tests, il faut d'abord télécharger JUnit 4 et hamcrest-core et placer les fichiers .jar à un endroit connu (l'emplacement du projet par exemple). En utilisant les commandes du README en adaptant les chemins en fonction de l'emplacement local, on peut compiler exécuter les tests.

3 Exercice 3

3.1 Q1-Q2

Pour travailler parallèlement sur le projet, nous avous utilisé le logiciel de versionning git couplé à un outil de dépôt en ligne appelé GitHub. Les étapes effectuées pour initialiser le dépôt sont:

- Création de l'arboresence des fichiers, avec un dossier "Projet" contenant le code Java, et un dossier "Rapport" contenant le rapport
- Ecriture du fichier .gitignore, permettant de ne pas avoir de fichiers inutiles dans le dépôt, notamment le bytecote généré par Java et les fichiers de log de LaTeX.
- Initialisation du dépôt local avec la commande 'git init'
- Ajout des fichiers à prendre en compte dans le commit avec 'git add .'
- Commit la version de départ avec 'git commit -m "Commit initial"'
- Création du dépôt distant sur github avec l'utilitaire invoqué par la commande'gh repo create'
- Push du contenu local sur le dépôt distant avec 'git push -u origin master'

Par la suite, nous avons effectué les différents commits via la console en ajoutant des tags quand nécéssaire avec la commande 'git tag'. Pour ajouter les tags dans le dépôt distant, il faut exécuter la commande 'git push origin –tags'

3.2 Q3

```
// TestDossierBancaire.java
@Test
public void test_constructeur()
{
   DossierBancaire dossier=new DossierBancaire();
   assertNotNull(dossier);
}

@Test
public void test_deposer()
{
   DossierBancaire dossier=new DossierBancaire();
   dossier.deposer(100);
   assertEquals(100,dossier.get_solde(),0);
```

```
}

@Test
public void test_remunerer()
{
   DossierBancaire dossier=new DossierBancaire();
   dossier.deposer(100);
   dossier.remunerer();
   double apres_remuneration = 0.4 * 100 + 0.6 * 100 * 1.032;
   assertEquals(apres_remuneration,dossier.get_solde(),0);
}
```

L'éxecution de ces tests fonctionne sauf pour rémunérer qui n'a pas encore été implémentée, comme montré sur la figure 5.

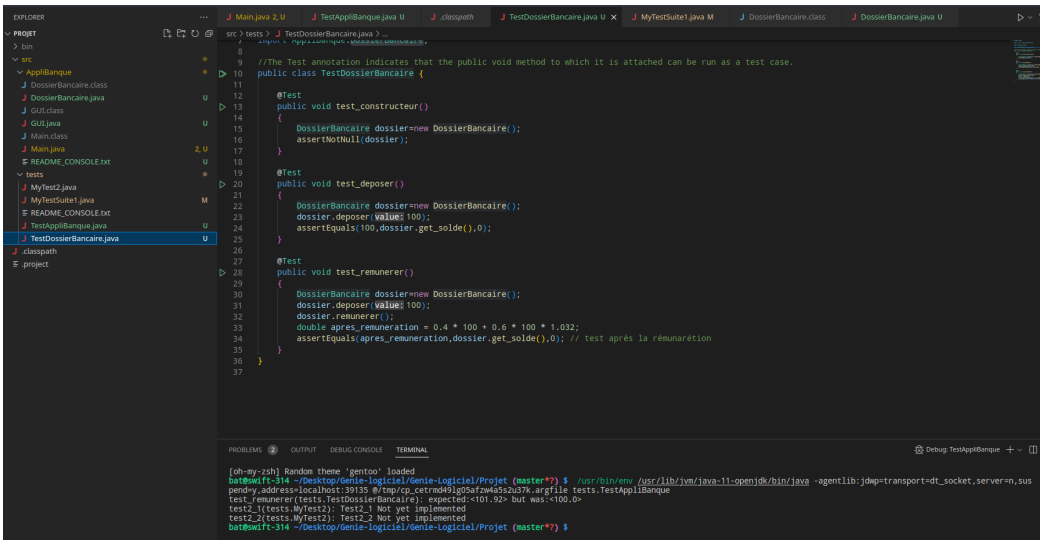

Figure 5: Tests

3.3 Q5

```
// CompteBancaire.java
package AppliBanque;

public class CompteBancaire {
```

```
    protected double m_solde;

    public CompteBancaire()
    {
        m_solde=0;
    }
    public void deposer(double value) {m_solde+=value;}
    public double get_solde() {return m_solde;}

}
```

4 Exercice 4

4.1 Q2

Le retour à une version précédente se fait via la commande

```
git checkout V2.0
```

(pour retourner au commit avec le tag V2.0 par exemple)

La création de la nouvelle branche à partir de V2.0 se fait avec la commande

```
git branch new_dev
```

Pour se rendre sur la branche (déplacer la HEAD sur la branche), il faut faire la commande

```
git checkout new_dev
```

4.2 Q3

Pour retourner sur la branche principale (déplacer la HEAD sur la branche), il faut faire la commande

```
git checkout master
```

4.3 Q4

L'intégration des modification apportees dans new_dev sur la branche master se fait au moyen d'un merge. Se dernier s'effectue avec la commande:

```
git merge new_dev master
```