TDM – Sérialisation Java

O. Aktouf

Rappels

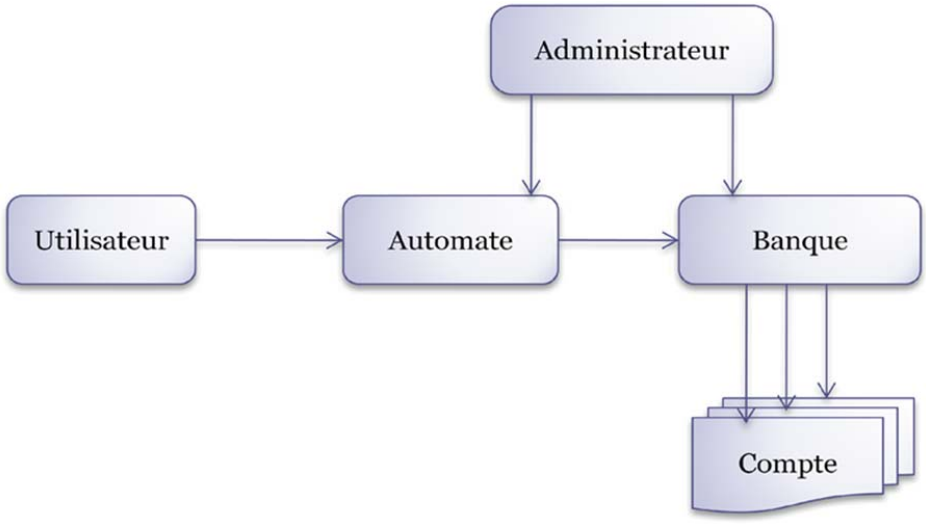
La persistance permet de rendre permanents les objets stockés en mémoire. En Java, le mécanisme qui permet d'assurer la persistance s'appelle la **sérialisation**. Sans le mécanisme de sérialisation, le programmeur est obligé d'imaginer une structure de stockage pour ses objets et d'implémenter toutes les procédures qui permettent de réaliser le stockage mais aussi de gérer toutes les dépendances entre les différents objets. La sérialisation intervient aussi dans la transmission de l'état des objets entre deux JVM distantes.

Pour qu'un objet puisse être sérialisé (transformé en une suite séquentielle d'octets), sa classe doit implémenter l'interface **`java.io.Serializable`**.

Exercice 1 – Sérialisation Java « binaire »

Partie A)

Soit l'application bancaire représentée par le diagramme ci-dessous.



Toutes les classes de l'application sont déjà implémentées, cependant, certaines doivent être complétées. Vous pouvez télécharger ces classes à partir de Chamilo.

A.1. Complétez la classe **`Utilisateur`** en implémentant la méthode qui permet de créditer un compte particulier de la banque. Copiez le code suivant :

```
public void crediter()
{
	try
	{
		int code = Integer.parseInt(champNumCompte.getText());
		int somme = Integer.parseInt(champSomme.getText());
		champSomme.setText("");
		boolean resultat = automate.crediter(code, somme);
		if (resultat)
			champStatut.setText("Opération bien déroulée.");
		else
			champStatut.setText("Opération mal déroulée.");
	}
	catch (NumberFormatException e)
	{
		champStatut.setText("Attention : saisie incorrecte !");
	}
}
```

A.2. En vous inspirant du code de cette méthode, complétez la méthode **`debiter()`** qui permet de débiter un compte particulier de la banque.

A.3. Ecrivez la classe **`Main`** qui permet de tester cette application. Vous pouvez par exemple utiliser le code suivant :

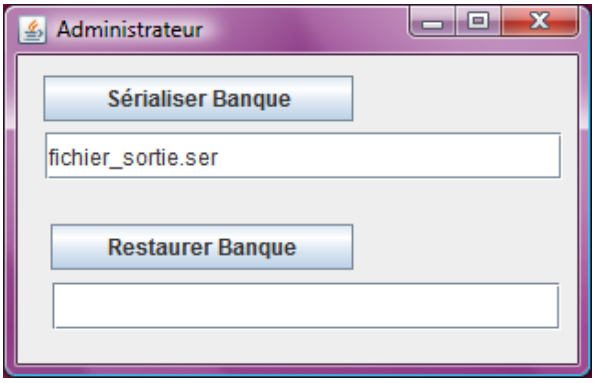
```
public class Main
{
	public static void main(String[] args)
	{
		// créer une banque
		Banque banque = new Banque("Crédit pour Tous");
		// créer quelques comptes
		banque.ajouterCompte(1001, "client1");
		banque.ajouterCompte(1002, "client2");
		banque.ajouterCompte(1003, "client3");

		// créer un automate et l'associer à la banque
		Automate automate = new Automate(banque);

		// créer l'utilisateur de l'ATM
		Utilisateur user = new Utilisateur(automate);
	}
}
```

Partie B)

Les exploitants de l'application souhaitent la rendre *persistante*. On voudrait plus particulièrement pouvoir sauvegarder l'état de la banque (avec ses comptes bien sûr !) pour pouvoir le retrouver en cas de panne. On voudrait, dans un premier temps, utiliser la sérialisation Java (package **`java.io`**) tel que vous l'avez vue en cours.

B.1. Quelles sont les modifications nécessaires ?

La classe **`Administrateur`** fournit une interface graphique qui permet de sérialiser ou de restaurer l'état de la banque à un moment donné.



Complétez cette classe en implémentant les deux méthodes de sérialisation et de restitution de l'état de la banque : **`serialiserBanque()`** et **`deserialiserBanque()`**.

Important : lors de la restauration d'un état précédemment sauvegardé, ne pas oublier de mettre à jour la valeur de l'attribut **`banque`** de la classe **`Administrateur`** et aussi celui de la classe **`Automate`**.

```
this.banque = ...;
this.automate.setBanque(...);
```

B.2. Modifiez la classe **`Main`** pour pouvoir créer l'administrateur. Ajoutez par exemple à la fin de la classe **`Main`** le code suivant :

```
// créer un administrateur de la banque
Administrateur admin = new Administrateur(banque, automate);
```

B.3. Testez maintenant l'application. Vérifiez surtout la sauvegarde et la restauration.

Exercice 2 – Sérialisation Java XML

2.1. Tentez d'ouvrir (avec un éditeur de texte) l'un des fichiers sérialisés. Que constatez-vous ?

Pour pouvoir visualiser et manipuler plus facilement les résultats de sérialisation, les concepteurs de l'application ont décidé d'avoir recours à la sérialisation en format XML.

2.2. Modifiez maintenant, en vous inspirant des exemples vus en cours, l'application pour permettre la sérialisation XML.

2.3. Que faut-il ajouter dans les deux classes **Banque** et **Compte**?

Important : pour la classe **Banque**, on s'intéresse aux attributs **nom** et **comptes**. Pour la classe **Compte**, on s'intéresse aux attributs **code**, **client**, **d_ouverture** et **solde**.

2.4. Modifiez la classe **Administrateur**(modifiez les deux méthodes de sérialisation) pour permettre la sérialisation XML.

Vérifiez les fichiers générés après la sérialisation. Quelle conclusion pouvez-vous en tirer ?