SCIENCES SORBONNE UNIVERSITÉ

SRCS : Systèmes Répartis Client/Serveur

TME 1 : Objets persistants

Objectifs pédagogiques :
- **I/O java sur des objets**
- **Sensibilisation à la sérialisation d'objet**

Introduction

En java, l'API basique des entrées/sorties se caractérise par les classes abstraites `InputStream` et `OutputStream`. Nous avons vu différentes implantations et différentes manières de décorer ces implantations à travers la définition de filtres. Un filtre permet de non seulement redéfinir des méthodes de l'interface initiale mais également d'en rajouter. C'est le cas notamment pour les classes `DataInputStream` et `DataOutputStream` qui permettent respectivement de recevoir et d'envoyer des données de type primitif java (`int`, `long`, `double`, `boolean`, ...) et enrichissant ainsi l'API de base qui n'offrait la possibilité que d'envoyer ou de recevoir des octets (`byte`). Ceci est une première étape vers la notion de sérialisation qui sera prochainement abordée.

Préparation

- Récupérer les ressources java associées à ce TME
- Vous utiliserez au fur et à mesure de votre avancement les différentes classe Junit permettant de tester votre code
- Créer un package `srcs.persistance`
- Dans votre code vous gérerez les exceptions avec la clause `throws` (implique pas de bloc `try catch`).

Exercice 1 – Persistance d'un tableau d'entiers

Question 1

Dans le package `srcs.persistance` créer la classe `PersistanceArray` écrire une méthode statique `void saveArrayInt(String f, int[] tab)` qui sauvegarde l'ensemble des entiers de `tab` dans le fichier `f`.

Question 2

Dans cette même classe `PersistanceArray` écrire une méthode statique `int[] loadArrayInt(String fichier)` qui charge à partir d'un fichier passé en paramètre un tableau d'entier.

Question 3

Testez vos méthodes avec la classe `srcs.persistance.test.TableauIntTest`.

Exercice 2 – Persistance d'un objet

Nous souhaitons à présent pouvoir sauvegarder ou charger un objet java vers ou depuis un fichier. Dans cet exercice, nous allons appliquer ceci sur la classe `Compte` fournie. Couper-coller la classe Compte fournie dans votre dossier source.

Question 1

Dans la classe `Compte` ajouter une méthode `void save(OutputStream out)` qui permet d'écrire l'instance dans le flux `out`.

Question 2

Dans la classe `Compte` ajouter un constructeur `Compte(InputStream in)` qui permet d'initialiser un compte à partir de données lues depuis le flux `in`.

Question 3

Dans le package `srcs.persistance` créer la classe `PersistanceCompte` et y écrire une méthode statique `void saveCompte(String f, Compte e)` qui sauvegarde un compte dans un fichier `f`.

Question 4

Dans cette même classe `PersistanceCompte` écrire une méthode statique `Compte loadCompte(String f)throws IOException` qui instancie un `Compte` à partir des données sauvegardées dans le fichier `f`.

Question 5

Testez votre code avec la classe `srcs.persistance.test.CompteTest`.

Nous souhaitons à présent abstraire la capacité à un objet quelconque de se sauvegarder et de s'instancier en se chargeant à partir d'un flux d'I/O. L'idée est de définir une interface `Sauvegardable` qui offre la méthode `void save(OutputStream out)` et de supposer que toute classe qui implantera cette interface offrira un constructeur qui prend en paramètre un `InputStream`.

Question 6

Dans le package `srcs.persistance`, définir l'interface `Sauvegardable` et la faire implanter par la classe `Compte`.

Question 7

Dans le package `srcs.persistance` créer une classe `PersistanceSauvegardable` et y écrire les méthodes statique `void save(String fichier, Sauvegardable s)` et `Sauvegardable load(String fichier)` qui permettent respectivement de sauvegarder dans un fichier un objet `Sauvegardable` et instancie un Sauvegardable à partir des données d'un fichier.

Question 8

Testez votre code avec la classe `srcs.persistance.test.SauvegardableTest`.

Exercice 3 – Persistance d'un graphe d'objets

Nous allons maintenant généraliser le concept de sauvegarde/chargement non pas à un objet mais à un graphe d'objets, c'est à dire un ensemble d'objet liés par des références. Dans cet exercice nous allons appliquer ceci à la classe `Banque` qui référence un ensemble de clients

Question 1

Couper-coller les classes `Banque` et `Client` fournies dans votre dossier source. Les faire implanter de l'interface `Sauvegardable` définie dans l'exercice précédent.

Question 2

Testez votre code avec le fichier `srcs.persistance.test.Banque2Clients2ComptesTest`.

Question 3

Testez à nouveau votre code avec le test unitaire `srcs.persistance.test.BanqueCompteJoinTest` en décommentant la dernière ligne de la méthode test. Après avoir analyser le scénario du test, expliquez pourquoi cette assertion ne passe pas. Le problème peut-il se résoudre facilement ?