Correction du TPPOO

Auteurs : Pelisse ROMAIN
Date :27/10/07

38, rue Molière
94200 Ivry Sur Seine
Tél. : 01 49 54 07 50
Fax : 01 49 54 07 40
http://www.esme.fr

Table of Contents

38, rue Molière
94200 Ivry Sur Seine
Tél. : 01 49 54 07 50
Fax : 01 49 54 07 40
http://www.esme.fr

1. Introduction

1.1. Objet du document

Ce document intitulé abusivement «Correction du TPPOO» regroupe un ensemble de remarques sur la conception du sujet de TPPOO de cette année. Il propose aussi, dans les grandes lignes, une modélisation possible du sujet.

1.2. Termes et abréviations

Terme	Définition
TPPOO	TP d'intiation à la programmation orienté objet
IHM	Interface Homme Machine

1.3. Documents associés et références

[1] TP POO – Sujet 2007
Romain PELISSE – Octobre 2007 – Esme Sudria

38, rue Molière
94200 Ivry Sur Seine
Tél. : 01 49 54 07 50
Fax : 01 49 54 07 40
http://www.esme.fr

2. Architecture Statique

Comme souvent dans la conception d'un logiciel, on distingue les 3 problématiques au sein de ce sujet:

- La logique «métier» du logiciel, ici recevoir, conserver, restiturer des messages,
- La modélisation des données, ici la pile et ses messages,
- La persistance de ces données ou la sauvergarde de ces données.

L'architecture interne de notre application se divisera donc en 2 modules, un chargé d'implémenter cette logique métier, l'autre de modéliser la « pile ».

2.1. Diagramme UML

Le shéma ci dessous présente la structure des principales classes de la solution proposée en guise de correction:

- La classe **DefaultEngine** implémente l'interface MessageEngine et représente surtout l'implémentation de l'aspect métier de notre pile. Pour effectuer son travail, elle s'appuie sur 2 services internes définies dans les interface MessagesStack et MailboxesStack. DefaultEngine prend en charge les opérations suivantes:
  - *postMessage*: A l'arrivée d'un message, DefaultEngine, effectuer les traitements suivants:
    - création d'une instance de MetaMessage, qu'elle associe au message reçu.
    - Place l'instance de MetaMessage dans son instance de MessagesStack, en l'indexant la référence par l'identifiant du message (m.getId()).
    - Elle demande au service MailboxesStack si le destinataire a déjà une boite de réception. Si c'est le cas, elle ajoute la référence du Message à cette boite, sinon, elle créer la boite de réception et l'ajoute à MailboxesStack. *Remarque : MailboxesStack indexe les boites de reception par leur nom.*
  - *isRecevied:* La réception d'une demande d'accusé de reception, DefaultEngine effectue le traitement suivant:
    - DefaultEngine utilise son service MessagesStack, qui indexe les MetaMessage par l'identifiant du Message qu'il contient, pour récupér le message directement à partir de son identifiant.
    - Elle retourne la valeur du champs 'read' du MetaMessage.
  - *checkMessages*: A la reception d'une demande de relève de messages, DefaultEngine gère ainsi la demande:
    - A l'aide de la methode get(K<String> key), elle récupère directement l'instance associé à la boite de message ou 'null' si la boite est vide. L'implémentation de MailboxesStack utilisé par DefaultEngine a alors la charge de « tagguer » l'ensemble des messages comme « releve » (setRead(true); sur l'ensemble des messages de la boite).
    - DefaultEngine utilise la méthode 'checkout' de MailboxesStack pour indique que les messages ont été lues. Mailboxes va simplement invoquer 'checkout' du service MessagesStack sur chaque message, de manièr à assurer que
      - Le message soit bien taggué « lu » (read = true;)
      - Que l'information soit propager à l'implémentation s'occupant de la persitance le cas écheant (voir plus loin).
    - DefaultEngine utilise la méthode 'checkout' de MailboxesStack pour indiquer que les m
    - Si la valeur retournée n'est pas null, DefaultEngine extrait les messages de la MailBoxes (getMessages():List) et les retourne.

- Si la valeur retournée est null, DefaultEngine créer une boite vide dans MailboxesStack et retourne une instance de List vide (return new List(0);).
-

La gestion des paramètres d'initialisation ainsi que la construction des services sur lesquelles s'appuie DefaultEngine, sera explicité plus loin.

- La classe **MetaMessage**: Cette classe adresse l'aspect modélisation des données du sujet. Dès que notre pile recevra un message, elle va créer une instance de cet classe et lui ajoutera la référence vers l'instance de message reçu. Cette classe sera donc un enrobage (un « wrapper » en anglais) qui nous permettra de placer des informations (des métadonnées) sur chaque message contenu dans la pile. Dans le cadre du TP, on place une valeur booléenne pour indiquer si le message a été ou non relevé par son destinataire. Ce classe utilise une implémentation de l'interface nommée **SerializableToFile** décrite dans la partie Persistence de ce document.
- La classe **Mailboxes**: C'est classe adresse l'aspect modélisation des données du sujet., elle classe modélise une boite de message. Elle se caractèrise donc par son nom (le champs recipient du message), et par une suite de message. A chaque ajout d'une instance de MetaMessage, la classe met à jour une liste (metaMessages()) avec les références de tout ses MetaMessage, mais aussi une autre liste (messages()) en récupérant la référence du message contenu dans le MetaMessage (meta.getMessage()). Ce mécanisme permet de retourner indifférament une liste de Message ou de MetaMessage sans passer par une itération complexe (Ce genre de traitement est donc éviter : « Pour chaque message, je récupère la référence vers le message que j'ajoute dans une nouvelle liste... »).

  La classe propose aussi une méthode 'checkout' qui permet de tagguer tout les messages qu'elle contient comme lu.

  Suppression des messages : pour pouvoir indiquer à un client si le message envoyé a été lu, il est nécessaire de conserver les métadonnées d'un message (le champs read de la classe MetaMessage), mais potentiellement, les messages peuvent s'accumuler à l'infini ! Pour éviter que les messages ne prennent trop de place en mémoire, la MailBoxes proposer une méthodes 'emptyBox' qui supprime le contenu (le champs content, avec un setContent(null);) de tout message ayant été relevé (read == true). Ca ne résout pas entièrement le problème, car plus il y aura de message, plus la pile continuera de se remplir, mais le garbage collector pourra au moins supprimé le contenu (potentiellement très lourd) des messages, pour ne garder que les données minimales de ce dernier.

- L'interface **MessagesStack** est un fait qu'une simple Map indexant les instances de MetaMessage par l'identifiant du Message qu'il contient (put(<String>idMessage, <MetaMessage>message)). L'interface se content d'hériter de l'interface Map pour offrir un type précis à ce service L'utilisation d'un interface pour définir ce service nous permettra de changer facilement d'implémentation si besoin est et assure aussi une bonne séparation entre notre couche métier (logique de gestion des messages) et la persistance des messages. **La sauvegarde des messages sera donc transparente pour la couche métier.** La principale raison d'exister de ce service est de permettre un accès rapide à un message simplement à partir de son identifiant.
- L'interface **MailboxesStack** n'est aussi qu'une redéfinition de l'interface Map. Ce service indexe donc les Mailbox par leur nom, permettant un accès rapide aux messages d'une boite.L'utilisation d'un interface pour définir ce service nous permettra de changer facilement d'implémentation si besoin est et assure aussi une bonne séparation entre notre couche métier (logique de gestion des messages) et la logique interne de ce composant technique.

  Les classes qui suivent adresse la problématique de la persistance des données.
- La classe **MemoryStack** est une classe fille de l'implémenation de l'interface Map nommée HashMap fournit par le JDK. Par défaut, il n'y a pas à surcharger les méthodes de HashMap, car pour une utilisation en mémoire seule, HashMap fournit déjà tout ce dont notre programme a besoin. Néanmoins, nous allons surcharger les méthodes put et get pour générer des journaux d'éxectution (logs en anglais):

```
package org.esme.tppoo.correction;
```

38, rue Molière
94200 Ivry Sur Seine
Tél. : 01 49 54 07 50
Fax : 01 49 54 07 40
http://www.esme.fr

```
import java.util.HashMap;
import java.util.logging.Logger;

public class MemoryUsage<Message> extends HashMap implements
MessagesStack  {

	// On créer un logger pour cette classe
	Logger logger = Logger.getLogger(this.getClass().getName());

	@Override
	public Object put(Object key,Object m) {
		logger.info("Adding this message" + m.toString());
		Object obj = super.put(key, m);
		if ( obj != null ) {
			logger.info("Message added.");
		}
		else
			logger.info("Message wasn't added.");
		return obj;
	}

	@Override
	public Object get(Object key) {
		logger.info("Retrivieng message with id:" + key);
		Object m = super.get(key);
		if ( m != null ) {
			logger.info("Message retrieved:" + m.toString());
		}
		else
			logger.info("Message is not in the stack");
		return m;
	}

	/**
	 * "Taggue" le message associant Ã  l'identifiant fourni
	 * comme "lu"
	 * @param mssgId
	 */
	public void checkout(String mssgId) {
		// FIXME: un usage habile des 'generics' en Java5
		// permettrait de supprimer ce 'cast'.
		MetaMessage m = super.get(mssgId);
		if (m != null ) m.setRead(true);
	}
}
```

La méthode *checkout(mssgId:String)* change simplement le champs 'read' du MetaMessage désigné.

38, rue Molière
94200 Ivry Sur Seine
Tél. : 01 49 54 07 50
Fax : 01 49 54 07 40
http://www.esme.fr

- La classe **FileStackImpl** est conçu pour gérer la persitance. Elle hérite de la classe précendante pour bénéficier du la logique de journalisation déjà implémenter. Elle surcharge les appels à put pour persiter les messages dans un fichier texte. Cette classe a pour objectif de persister les messages à leur arrivée (par la méthodes put()); une fois ceci fait, elle se contente d'appeler la méthode de adapté de MemoryUsage (super.put()). De même, lors du retrait d'un message (appel à la méthode checkout()), la pile va mettre à jour le message dans son fichier de persistance (voir plus loin) et retirer le contenu du message de la Map pour ne garder que les métadonnées.

  Suppression des messages: Dans cette classe, on peut aussi définir une stratégie pour la suppression des messages en mémoire (super.remove(id);). En effet, on peut par exemple, supprimer un message dès qu'il est checkout, et simplement vérifier sa présence ou non dans le fichier de persistance lors d'un get. Néanmoins, ce mécanisme entrainera un accès au fichier par get. Il baissera donc les performances, mais permettra d'alléger la pile en terme de volume de données.

- La classe **FileBatchStackImpl** implémente une stratégie légèrement différente de sa classe mère. Elle surcharge la méthode put() qu'elle hérite de sa classe mère. Dès qu'elle reçoit un nouveau message, elle ne le sauvegarde pas systématiquement. Elle place sa référence dans une liste (waitingToPersist). Quand la taille de cette liste a atteint la valeur défini dans nbMessageInBatch, elle persite alors les messages qu'ils ne l'étaient pas encore. Cette classe permet donc de diviser les accès disque, vers le fichier de persistance, par 'nbMessageInBatch'.

38, rue Molière
94200 Ivry Sur Seine
Tél. : 01 49 54 07 50
Fax : 01 49 54 07 40
http://www.esme.fr

3. Initialisation et gestion des paramètres de la pile

38, rue Molière
94200 Ivry Sur Seine
Tél. : 01 49 54 07 50
Fax : 01 49 54 07 40
http://www.esme.fr

4. Persistance des données

Comme dans de nombreuse application la persistance des données jouait un rôle crucial dans ce sujet. Voici quelques indications sur la manière d'implémenter celle ci.

4.1. Fichier texte

Le formalisme retenu pour ce fichier est simpliste au possible:

- Une ligne par message, chaque champs est séparé par une tabulation (« \t »).
- Le dernier champs contient le corps du message, sous forme de chaine de caractères.
- Les champs persisté son les suivants:
  - date d'arrivée du message au format ,
  - date où le message a été relevé, au même format ou simplement « not-checkout »,
  - id du message (message.getId();),
  - destinataire (message.getRecipient();),
  - contenu du message (message.getRecipient();)

En plaçant le contenu comme dernier champ, on permet à celui ci de contenir des tabulations. La classe se contentera de placer « toute la fin » de la ligne dans le champs 'content' du message, quand elle devra extraire le message du fichier.

Le fichier sera donc consistue de ligne semblable à celle ci :

```
06:43-27-10-07  06:43-27-10-07  535838747pysca@esme.fr  contenu du message
```

L'interface **SerializableToFile** décrit t les deux méthodes suivantes:

- **serialize :** Cette méthode retourne l'instance de MetaMessage sous forme d'une chaine de caractère..
- deserialize: Cette méthode reconstruit l'objet depuis la chaîne fournit.

La classe **SerializerImpl** implemente donc ces deux méthodes suivant le formalisme décrit ci dessus. Si on l'on souhaite changer de formalisme de persistance (par exemple persiter les messages sous formes d'élement XML), il suffit donc de créer une nouvelle implémentation de cette interface, sans changer le reste du code.

Remarque, le serialize est un « outil » commun à toutes les instances de la classe MetaMessage, il est donc pertinent d'en faire un champs static et de leur construire en utilisant leur Design Pattern Singleton.

Si on l'on souhaite changer le medium de persistance, par exemple remplacer le fichier par une base de données, ou pourquoi par l'envoi d'un message sur le réseau, il suffit d'implémenter une nouvelle version de l'interface **MessagesStack**, et y implémenter la nouvelle stratégie de persistance...

4.2. Pour aller plus loin...

Dans le cadre de ce TP, une simple approche par fichier texte était suffisante. En effet, le modélisation des données se limitait à une seule classe, Message. Dans un cas plus complexe, ce qui sera le cas très probablement en projet, vous devrez gérer une arborescence de données relativement complexe. Pour gérer ce genre de considération, je vous invite à étudier les *Design Pattern* DAO (Data Access Object) et ORM (Objet Relation Mapping), sur lequel vous trouverez une littérature abondante.

38, rue Molière
94200 Ivry Sur Seine
Tél. : 01 49 54 07 50
Fax : 01 49 54 07 40
http://www.esme.fr

5. La performance et la gestion de la mémoire

5.1. Accès rapide au message recherché

Comme vous l'aurez compris, dans la solution décrite, chaque message est doublement référencé, une fois dans l'instance de MessagesStack et une autre fois dans l'instance MaiboxesStack. Cette architecture permet d'améliorer les performances, en récupérant, via l'utilisation de Map, très vite le message, quelque soit le critère utilisé pour retrouver un message, on profite dans tout les cas dans la puissance de l'algorithme de HashMap.

5.2. Risque de fuite mémoire

A l'inverse, cette architecture présente un risque en terme de consommation mémoire. En effet, comme le message est référencé **deux fois**, il faut s'assurer que ces deux référence soient mise à 'null' pour que le « garbage collector » puisse libérer la mémoire associée à ses messages, une fois inutile.

Notez bien néanmoins, que ce sont les référence d'un message qui sont stocké dans les différentes Stack, et qu'il n'y a donc pas de « doublon » en mémoire des messages. De plus, si l'on souhait modifier un message, que l'on passe par MessagesStack pour par MailboxesStack, le message en lui même sera mise à jour.

5.3. La pile principale

Il est important aussi de comprendre que MailboxesStack est facilité d'accès au messages et que c'est bien MessagesStack qui est « l'épine dorsale » de notre pile. C'est elle qui gère la persistance et la cohérence des messages. Si on effectue, par le biais, de MailboxesStack, une opération qui change la cohérence des messages (comme l'opération 'checkout'), il est donc important de propager cette opération à MessagesStack pour assurer la cohérence de nos données.

38, rue Molière
94200 Ivry Sur Seine
Tél. : 01 49 54 07 50
Fax : 01 49 54 07 40
http://www.esme.fr

6. Déroulement du TP

6.1. Questions posées

Pendant le déroulement du TP, j'ai reçu en tout une trentaine de mails de questions, dont les sujets étaient :

- renommage dans le cvs
- implémentation de la persistance (utilisation de la sérialization java)
- problème de compréhension sur la notion de persistance
- Gestion des accès concurrent
- Question sur DummyStack
- Erreur et concept du test greatPayLoad()
- Utilisation de RMI pour gérer la concurrence
- Question sur le DAL
- Questions sur le tests unitaires
- Qualification de la pile
- Réutilisation de StackParameter
- Conception de l'IHM du TP (il n'y en avait pas à faire !)
- Pile sous forme de serveur
- Protocole de communication réseau de la pile

Au moment de l'écriture de ce document, je n'ai pas encore corrigé vos Tps, néanmoins, je me permet de vous signaler le point suivant : Dans aucun de ces questions ne se trouve de questions relative à la **modélisation Objet de la problématique du sujet**. On trouve en vrac des questions de compréhension sur le sujet, ce qui est naturel, ainsi que des questions techniques, là aussi, c'est bien naturel, mais je n'ai reçu aucune question sur la modélisation ... Je vous laisse en tirer les conclusions qui s'imposent sur le manque de « recul » dont vous faites souvent preuve dans la réalisation d'un TP ;)

6.2. Utilisation du CVS

Commité le dernier jours !

```
pelisse@flops:/cvsd/esme/tp$ ls 3ct*/ -l | sed -e '/build/d'
3ct101/:
total 20
drwxrwxr-x 3 cvsd cvsd 4096 2007-10-22 21:44 CVSROOT
drwxr-xr-x 2 cvsd cvsd 4096 2007-10-03 19:08 tpbuilt
drwxr-x--- 5 cvsd cvsd 4096 2007-10-22 21:33 tpooremise
drwxr-x--- 8 cvsd cvsd 4096 2007-10-22 21:44 tppoo

3ct102/:
total 16
drwxrwxr-x 3 cvsd cvsd 4096 2007-10-22 21:22 CVSROOT
drwxr-xr-x 2 cvsd cvsd 4096 2007-10-05 14:06 tp1
drwxr-x--- 6 cvsd cvsd 4096 2007-10-27 20:55 tppoo
```

```
3ct103/:
total 12
drwxrwxr-x 3 cvsd cvsd 4096 2007-10-27 20:57 CVSROOT
drwxr-x--- 3 cvsd cvsd 4096 2007-10-27 20:57 tppoo

3ct104/:
total 16
drwxr-x--- 7 cvsd cvsd 4096 2007-10-23 14:39 2007
drwxrwxr-x 3 cvsd cvsd 4096 2007-10-27 20:56 CVSROOT
drwxr-x--- 6 cvsd cvsd 4096 2007-10-27 20:56 tppoo

3ct105/:
total 16
drwxrwxr-x 3 cvsd cvsd 4096 2007-10-22 19:54 CVSROOT
drwxr-xr-x 6 cvsd cvsd 4096 2007-10-05 14:50 TP_BUILD_1_0
drwxr-x--- 7 cvsd cvsd 4096 2007-10-27 20:56 tppoo

3ct106/:
total 12
drwxrwxr-x  3 cvsd cvsd 4096 2007-10-22 21:34 CVSROOT
drwxr-x---  8 cvsd cvsd 4096 2007-10-22 21:39 tppoo

3ct201/:
total 16
drwxrwxr-x 3 cvsd cvsd 4096 2007-10-22 18:46 CVSROOT
drwxr-x--- 8 cvsd cvsd 4096 2007-10-27 20:56 tppoo
drwxr-x--- 5 cvsd cvsd 4096 2007-10-13 15:53 tppoo-2007

3ct202/:
total 12
drwxrwxr-x 3 cvsd cvsd 4096 2007-10-22 18:46 CVSROOT
drwxr-xr-x 6 cvsd cvsd 4096 2007-10-06 20:25 tp1
drwxr-x--- 7 cvsd cvsd 4096 2007-10-27 20:55 tppoo

3ct203/:
total 16
drwxrwxr-x 3 cvsd cvsd 4096 2007-10-22 21:31 CVSROOT
drwxr-x--- 2 cvsd cvsd 4096 2007-10-11 20:51 Dossier_Spec
drwxr-x--- 8 cvsd cvsd 4096 2007-10-27 20:56 tppoo

3ct204/:
total 16
drwxr-x---  6 cvsd cvsd 4096 2007-10-18 17:26 2007
drwxrwxr-x  3 cvsd cvsd 4096 2007-10-27 20:56 CVSROOT
drwxr-x---  7 cvsd cvsd 4096 2007-10-27 20:56 tppoo
```

38, rue Molière
94200 Ivry Sur Seine
Tél. : 01 49 54 07 50
Fax : 01 49 54 07 40
http://www.esme.fr

```
3ct205/:
total 12
drwxrwxr-x 3 cvsd cvsd 4096 2007-10-22 20:00 CVSROOT
drwxr-x--- 7 cvsd cvsd 4096 2007-10-27 20:56 tppoo

3ct206/:
total 20
drwxr-xr-x  7 cvsd cvsd 4096 2007-10-22 14:38 2007
drwxrwxr-x  3 cvsd cvsd 4096 2007-10-22 20:35 CVSROOT
drwxr-xr-x  5 cvsd cvsd 4096 2007-10-12 13:17 projet2
drwxr-x---  6 cvsd cvsd 4096 2007-10-27 20:55 tppoo

Logging in to :pserver:pelisse@127.0.0.1:2401/esme/tp/3ct102
CVS password:
cvs [checkout aborted]: no such tag `TPOO_REMISE'
mkdir: ne peut créer le répertoire `tppoo/3ct103': Aucun fichier ou répertoire de ce type
getTp.sh: line 8: cd: tppoo/3ct103: Aucun fichier ou répertoire de ce type
Logging in to :pserver:pelisse@127.0.0.1:2401/esme/tp/3ct103
CVS password:
cvs checkout: Updating tppoo
cvs checkout: Updating tppoo/tppoo
U tppoo/tppoo/.classpath
U tppoo/tppoo/.cvsignore
U tppoo/tppoo/.project
U tppoo/tppoo/build.xml
U tppoo/tppoo/esme-ruleset.xml
U tppoo/tppoo/pmd.xsl
cvs checkout: Updating tppoo/tppoo/docs
U tppoo/tppoo/docs/DAL.doc
cvs checkout: Updating tppoo/tppoo/lib
U tppoo/tppoo/lib/junit.jar
cvs checkout: Updating tppoo/tppoo/lib/asm
cvs checkout: Updating tppoo/tppoo/lib/asm/asm
cvs checkout: Updating tppoo/tppoo/lib/asm/asm/3.0
U tppoo/tppoo/lib/asm/asm/3.0/asm-3.0.jar
cvs checkout: Updating tppoo/tppoo/lib/backport-util-concurrent
cvs checkout: Updating tppoo/tppoo/lib/backport-util-concurrent/backport-util-concurrent
cvs checkout: Updating tppoo/tppoo/lib/backport-util-concurrent/backport-util-concurrent/3.0
U tppoo/tppoo/lib/backport-util-concurrent/backport-util-concurrent/3.0/backport-util-concurrent-3.0.jar
cvs checkout: Updating tppoo/tppoo/lib/jaxen
cvs checkout: Updating tppoo/tppoo/lib/jaxen/jaxen
cvs checkout: Updating tppoo/tppoo/lib/jaxen/jaxen/1.1-beta-10
U tppoo/tppoo/lib/jaxen/jaxen/1.1-beta-10/jaxen-1.1-beta-10.jar
cvs checkout: Updating tppoo/tppoo/lib/pmd
```

38, rue Molière
94200 Ivry Sur Seine
Tél. : 01 49 54 07 50
Fax : 01 49 54 07 40
http://www.esme.fr

```
cvs checkout: Updating tppoo/tppoo/lib/pmd/pmd
U tppoo/tppoo/lib/pmd/pmd/maven-metadata-local.xml
cvs checkout: Updating tppoo/tppoo/lib/pmd/pmd/4.0
U tppoo/tppoo/lib/pmd/pmd/4.0/pmd-4.0.jar
cvs checkout: Updating tppoo/tppoo/src
cvs checkout: Updating tppoo/tppoo/src/org
cvs checkout: Updating tppoo/tppoo/src/org/esme
cvs checkout: Updating tppoo/tppoo/src/org/esme/tppoo
U tppoo/tppoo/src/org/esme/tppoo/Erreur.java
U tppoo/tppoo/src/org/esme/tppoo/Mailbox.java
U tppoo/tppoo/src/org/esme/tppoo/Message.java
U tppoo/tppoo/src/org/esme/tppoo/MessageEngine.java
U tppoo/tppoo/src/org/esme/tppoo/MessageException.java
U tppoo/tppoo/src/org/esme/tppoo/MessageStack.java
U tppoo/tppoo/src/org/esme/tppoo/PersistancePile.java
U tppoo/tppoo/src/org/esme/tppoo/StackParameter.java
cvs checkout: Updating tppoo/tppoo/tests
cvs checkout: Updating tppoo/tppoo/tests/org
cvs checkout: Updating tppoo/tppoo/tests/org/esme
cvs checkout: Updating tppoo/tppoo/tests/org/esme/tppoo
cvs checkout: Updating tppoo/tppoo/tests/org/esme/tppoo/testsuite
U tppoo/tppoo/tests/org/esme/tppoo/testsuite/BasicUseCase.java
mkdir: ne peut créer le répertoire `tppoo/3ct104': Le fichier existe.
Logging in to :pserver:pelisse@127.0.0.1:2401/esme/tp/3ct104
CVS password:
cvs checkout: Updating tppoo
cvs checkout: Updating tppoo/bin
cvs checkout: Updating tppoo/docs
cvs checkout: Updating tppoo/lib
cvs checkout: Updating tppoo/lib/asm
cvs checkout: Updating tppoo/lib/asm/asm
cvs checkout: Updating tppoo/lib/asm/asm/3.0
cvs checkout: Updating tppoo/lib/backport-util-concurrent
cvs checkout: Updating tppoo/lib/backport-util-concurrent/backport-util-concurrent
cvs checkout: Updating tppoo/lib/backport-util-concurrent/backport-util-concurrent/3.0
cvs checkout: Updating tppoo/lib/jaxen
cvs checkout: Updating tppoo/lib/jaxen/jaxen
cvs checkout: Updating tppoo/lib/jaxen/jaxen/1.1-beta-10
cvs checkout: Updating tppoo/lib/pmd
cvs checkout: Updating tppoo/lib/pmd/pmd
cvs checkout: Updating tppoo/lib/pmd/pmd/4.0
cvs checkout: Updating tppoo/tests
cvs checkout: Updating tppoo/tests/org
Logging in to :pserver:pelisse@127.0.0.1:2401/esme/tp/3ct105
CVS password:
```

```
cvs [checkout aborted]: no such tag `TPOO_REMISE'
Logging in to :pserver:pelisse@127.0.0.1:2401/esme/tp/3ct201
CVS password:
cvs [checkout aborted]: no such tag `TPOO_REMISE'
mkdir: ne peut créer le répertoire `tppoo/3ct202': Aucun fichier ou répertoire de ce type
getTp.sh: line 8: cd: tppoo/3ct202: Aucun fichier ou répertoire de ce type
Logging in to :pserver:pelisse@127.0.0.1:2401/esme/tp/3ct103
CVS password:
cvs checkout: Updating tppoo
cvs checkout: Updating tppoo/tppoo
U tppoo/tppoo/.classpath
U tppoo/tppoo/.cvsignore
U tppoo/tppoo/.project
U tppoo/tppoo/build.xml
U tppoo/tppoo/esme-ruleset.xml
U tppoo/tppoo/pmd.xsl
cvs checkout: Updating tppoo/tppoo/docs
U tppoo/tppoo/docs/DAL.doc
cvs checkout: Updating tppoo/tppoo/lib
U tppoo/tppoo/lib/junit.jar
cvs checkout: Updating tppoo/tppoo/lib/asm
cvs checkout: Updating tppoo/tppoo/lib/asm/asm
cvs checkout: Updating tppoo/tppoo/lib/asm/asm/3.0
U tppoo/tppoo/lib/asm/asm/3.0/asm-3.0.jar
cvs checkout: Updating tppoo/tppoo/lib/backport-util-concurrent
cvs checkout: Updating tppoo/tppoo/lib/backport-util-concurrent/backport-util-concurrent
cvs checkout: Updating tppoo/tppoo/lib/backport-util-concurrent/backport-util-concurrent/3.0
U tppoo/tppoo/lib/backport-util-concurrent/backport-util-concurrent/3.0/backport-util-concurrent-3.0.jar
cvs checkout: Updating tppoo/tppoo/lib/jaxen
cvs checkout: Updating tppoo/tppoo/lib/jaxen/jaxen
cvs checkout: Updating tppoo/tppoo/lib/jaxen/jaxen/1.1-beta-10
U tppoo/tppoo/lib/jaxen/jaxen/1.1-beta-10/jaxen-1.1-beta-10.jar
cvs checkout: Updating tppoo/tppoo/lib/pmd
cvs checkout: Updating tppoo/tppoo/lib/pmd/pmd
U tppoo/tppoo/lib/pmd/pmd/maven-metadata-local.xml
cvs checkout: Updating tppoo/tppoo/lib/pmd/pmd/4.0
U tppoo/tppoo/lib/pmd/pmd/4.0/pmd-4.0.jar
cvs checkout: Updating tppoo/tppoo/src
cvs checkout: Updating tppoo/tppoo/src/org
cvs checkout: Updating tppoo/tppoo/src/org/esme
cvs checkout: Updating tppoo/tppoo/src/org/esme/tppoo
U tppoo/tppoo/src/org/esme/tppoo/Erreur.java
U tppoo/tppoo/src/org/esme/tppoo/Mailbox.java
U tppoo/tppoo/src/org/esme/tppoo/Message.java
U tppoo/tppoo/src/org/esme/tppoo/MessageEngine.java
```

U tppoo/tppoo/src/org/esme/tppoo/MessageException.java
U tppoo/tppoo/src/org/esme/tppoo/MessageStack.java
U tppoo/tppoo/src/org/esme/tppoo/PersistancePile.java
U tppoo/tppoo/src/org/esme/tppoo/StackParameter.java
cvs checkout: Updating tppoo/tppoo/tests
cvs checkout: Updating tppoo/tppoo/tests/org
cvs checkout: Updating tppoo/tppoo/tests/org/esme
cvs checkout: Updating tppoo/tppoo/tests/org/esme/tppoo
cvs checkout: Updating tppoo/tppoo/tests/org/esme/tppoo/testsuite
U tppoo/tppoo/tests/org/esme/tppoo/testsuite/BasicUseCase.java
Logging in to :pserver:pelisse@127.0.0.1:2401/esme/tp/3ct203
CVS password:
cvs [checkout aborted]: no such tag `TPOO_REMISE'
Logging in to :pserver:pelisse@127.0.0.1:2401/esme/tp/3ct204
CVS password:
cvs checkout: Updating tppoo
U tppoo/DAL TP2.doc
U tppoo/build.properties
U tppoo/build.xml
U tppoo/esme-ruleset.xml
U tppoo/junit-4.4.jar
U tppoo/pmd.xsl
cvs checkout: Updating tppoo/docs
U tppoo/docs/DAL TP2.doc
cvs checkout: Updating tppoo/lib
cvs checkout: Updating tppoo/lib/asm
cvs checkout: Updating tppoo/lib/asm/asm
cvs checkout: Updating tppoo/lib/asm/asm/3.0
U tppoo/lib/asm/asm/3.0/asm-3.0.jar
cvs checkout: Updating tppoo/lib/backport-util-concurrent
cvs checkout: Updating tppoo/lib/backport-util-concurrent/backport-util-concurrent
cvs checkout: Updating tppoo/lib/backport-util-concurrent/backport-util-concurrent/3.0
U tppoo/lib/backport-util-concurrent/backport-util-concurrent/3.0/backport-util-concurrent-3.0.jar
cvs checkout: Updating tppoo/lib/jaxen
cvs checkout: Updating tppoo/lib/jaxen/jaxen
cvs checkout: Updating tppoo/lib/jaxen/jaxen/1.1-beta-10
U tppoo/lib/jaxen/jaxen/1.1-beta-10/jaxen-1.1-beta-10.jar
cvs checkout: Updating tppoo/lib/pmd
cvs checkout: Updating tppoo/lib/pmd/pmd
U tppoo/lib/pmd/pmd/maven-metadata-local.xml
cvs checkout: Updating tppoo/lib/pmd/pmd/4.0
U tppoo/lib/pmd/pmd/4.0/pmd-4.0.jar
cvs checkout: Updating tppoo/src
cvs checkout: Updating tppoo/src/org
cvs checkout: Updating tppoo/src/org/esme

```
cvs checkout: Updating tppoo/src/org/esme/tppoo
U tppoo/src/org/esme/tppoo/MainClass.java
U tppoo/src/org/esme/tppoo/Message.java
U tppoo/src/org/esme/tppoo/MessageEngine.java
U tppoo/src/org/esme/tppoo/MessageStack.java
U tppoo/src/org/esme/tppoo/StackParameter.java
cvs checkout: Updating tppoo/src/org/esme/tppoo/exception
U tppoo/src/org/esme/tppoo/exception/InitializeStackException.java
U tppoo/src/org/esme/tppoo/exception/InvalidMessageException.java
U tppoo/src/org/esme/tppoo/exception/MailBoxMaxMessageException.java
U tppoo/src/org/esme/tppoo/exception/MessageException.java
U tppoo/src/org/esme/tppoo/exception/StackMaxMailBoxException.java
U tppoo/src/org/esme/tppoo/exception/StackMaxMessagesException.java
cvs checkout: Updating tppoo/src/org/esme/tppoo/persistence
U tppoo/src/org/esme/tppoo/persistence/AbstractPersistenceEngine.java
U tppoo/src/org/esme/tppoo/persistence/EachMessagePersistenceEngine.java
U tppoo/src/org/esme/tppoo/persistence/GroupMessagePersistenceEngine.java
U tppoo/src/org/esme/tppoo/persistence/TimePersistenceEngine.java
cvs checkout: Updating tppoo/tests
cvs checkout: Updating tppoo/tests/org
cvs checkout: Updating tppoo/tests/org/esme
cvs checkout: Updating tppoo/tests/org/esme/tppoo
U tppoo/tests/org/esme/tppoo/DummyStack.java
cvs checkout: Updating tppoo/tests/org/esme/tppoo/testsuite
U tppoo/tests/org/esme/tppoo/testsuite/BasicUseCase.java
Logging in to :pserver:pelisse@127.0.0.1:2401/esme/tp/3ct205
CVS password:
cvs [checkout aborted]: no such tag `TPOO_REMISE'
mkdir: ne peut créer le répertoire `tppoo/3ct206': Aucun fichier ou répertoire de ce type
getTp.sh: line 8: cd: tppoo/3ct206: Aucun fichier ou répertoire de ce type
Logging in to :pserver:pelisse@127.0.0.1:2401/esme/tp/3ct103
CVS password:
cvs checkout: Updating tppoo
cvs checkout: Updating tppoo/tppoo
U tppoo/tppoo/.classpath
U tppoo/tppoo/.cvsignore
U tppoo/tppoo/.project
U tppoo/tppoo/build.xml
U tppoo/tppoo/esme-ruleset.xml
U tppoo/tppoo/pmd.xsl
cvs checkout: Updating tppoo/tppoo/docs
U tppoo/tppoo/docs/DAL.doc
cvs checkout: Updating tppoo/tppoo/lib
U tppoo/tppoo/lib/junit.jar
cvs checkout: Updating tppoo/tppoo/lib/asm
```

38, rue Molière
94200 Ivry Sur Seine
Tél. : 01 49 54 07 50
Fax : 01 49 54 07 40
http://www.esme.fr

```
cvs checkout: Updating tppoo/tppoo/lib/asm/asm
cvs checkout: Updating tppoo/tppoo/lib/asm/asm/3.0
U tppoo/tppoo/lib/asm/asm/3.0/asm-3.0.jar
cvs checkout: Updating tppoo/tppoo/lib/backport-util-concurrent
cvs checkout: Updating tppoo/tppoo/lib/backport-util-concurrent/backport-util-concurrent
cvs checkout: Updating tppoo/tppoo/lib/backport-util-concurrent/backport-util-concurrent/3.0
U tppoo/tppoo/lib/backport-util-concurrent/backport-util-concurrent/3.0/backport-util-concurrent-3.0.jar
cvs checkout: Updating tppoo/tppoo/lib/jaxen
cvs checkout: Updating tppoo/tppoo/lib/jaxen/jaxen
cvs checkout: Updating tppoo/tppoo/lib/jaxen/jaxen/1.1-beta-10
U tppoo/tppoo/lib/jaxen/jaxen/1.1-beta-10/jaxen-1.1-beta-10.jar
cvs checkout: Updating tppoo/tppoo/lib/pmd
cvs checkout: Updating tppoo/tppoo/lib/pmd/pmd
U tppoo/tppoo/lib/pmd/pmd/maven-metadata-local.xml
cvs checkout: Updating tppoo/tppoo/lib/pmd/pmd/4.0
U tppoo/tppoo/lib/pmd/pmd/4.0/pmd-4.0.jar
cvs checkout: Updating tppoo/tppoo/src
cvs checkout: Updating tppoo/tppoo/src/org
cvs checkout: Updating tppoo/tppoo/src/org/esme
cvs checkout: Updating tppoo/tppoo/src/org/esme/tppoo
U tppoo/tppoo/src/org/esme/tppoo/Erreur.java
U tppoo/tppoo/src/org/esme/tppoo/Mailbox.java
U tppoo/tppoo/src/org/esme/tppoo/Message.java
U tppoo/tppoo/src/org/esme/tppoo/MessageEngine.java
U tppoo/tppoo/src/org/esme/tppoo/MessageException.java
U tppoo/tppoo/src/org/esme/tppoo/MessageStack.java
U tppoo/tppoo/src/org/esme/tppoo/PersistancePile.java
U tppoo/tppoo/src/org/esme/tppoo/StackParameter.java
cvs checkout: Updating tppoo/tppoo/tests
cvs checkout: Updating tppoo/tppoo/tests/org
cvs checkout: Updating tppoo/tppoo/tests/org/esme
cvs checkout: Updating tppoo/tppoo/tests/org/esme/tppoo
cvs checkout: Updating tppoo/tppoo/tests/org/esme/tppoo/testsuite
U tppoo/tppoo/tests/org/esme/tppoo/testsuite/BasicUseCase.java
```