Université de Bretagne Sud
UFR SSI, Dept MIS
CONCURRENCE et REPARTITION
Construction d'un système de messagerie pour les threads Java

Luc.Courtrai@univ-ubs.fr

On désire mettre en place un système de messagerie entre threads Java d'une même JVM. La messagerie sera construite au dessus d'un système de boîte aux lettres. Un message contient des objets Java. Des filtres permettent d'extraire les messages d'une boîte aux lettres en fonction de critères donnés (thread émetteur, class de l'objet contenu dans le message ...).

La synchronisation à utiliser est le wait/notify

Voici l'interface du système :

```
package messagebox;
/**
 * structure des messages
 * contenu dans les MessageBox
 */
public class  Message{
  /**
   * contruire un Message
   *  avec l'objet comme donnée
   *@param o objet du message
   */
   public Message(Object o){};
  /**
   * extrait l'objet du message
   *@return objet
   */
  public  Object getObject(){};
}
```

```
package messagebox;
/**
   *  class Filter
   * pour filtere les Message d'une MessageBox
   */
public class Filter{
  /**
   * contruire un filtre vide
   *  le plus viel objet de la MessageBox
   */
  public Filter(){};
  /**
   *  le plus viel objet de la MessageBox
   *        venant d'un thread particulier
   * @param emt thread emetteur du message
   **/
  public Filter(Thread emt){};
```

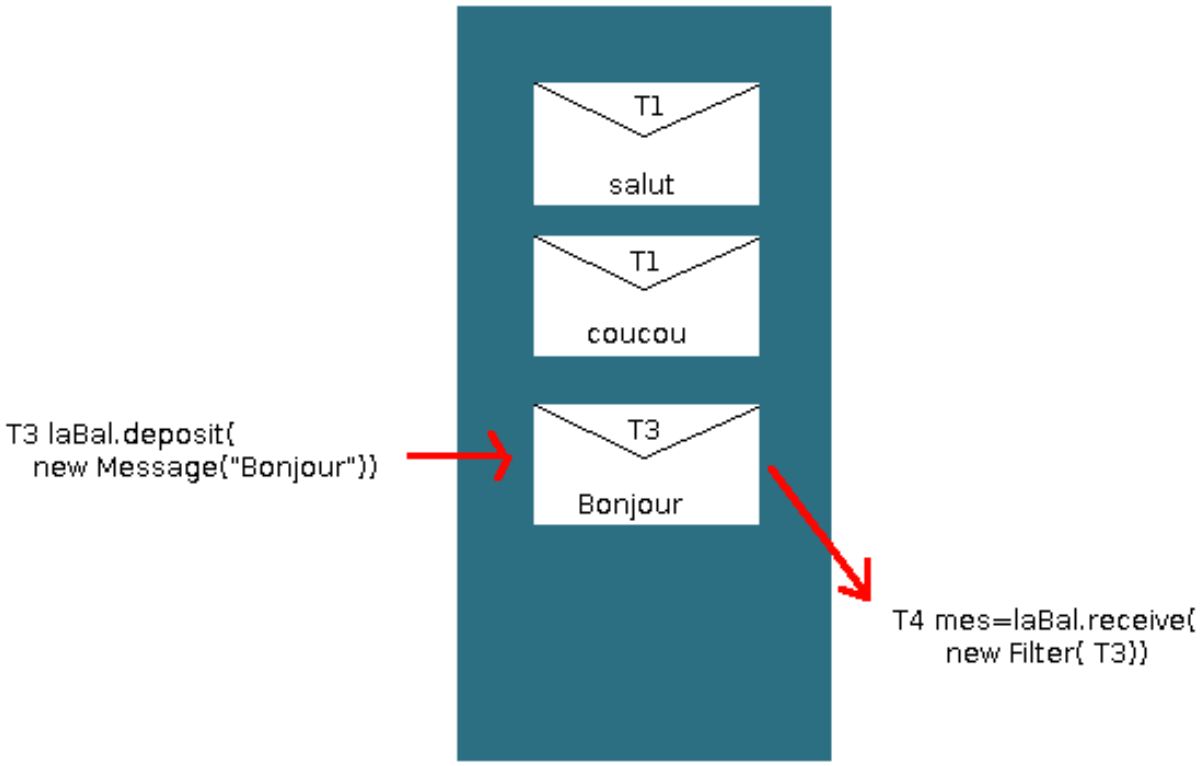
```
}
```

```
package messagebox;
/**
 * class MessageBox
 *  boite aux lettres
 *  pour la communication de threads
 */
public class MessageBox {

   /**
   * Depose un message dans la boite aux lettres
   * @param mes le message a deposer
   **/
  public void deposit(Message mes){};

  /** demande un message dans la boite aux lettres
  * repondant au filtre f
  * receive peut être bloquant
  * @param f : le filtre
  * @return le message extrait
  **/
  public Message receive(Filter f){};


  /**
   * test si la MessageBox contient un message qui
   *        repond a un fitre donne
   * @param f filtre a appliquer (eventuellement null)
   * @return Message en cas de succes (Null sinon)
   **/
   public Message probe(Filter f){};
}
```



Implantation

L'application utilisera les Threads java avec comme synchronisation les primitives wait/notify.

Le message doit contenir l'objet du message (ex "Bonjour"), ainsi que le thread émetteur (Thread.currentThred()).

La MessageBox doit contenir la liste des messages ainsi qu'une liste de threads en attente de message. Ex T5 maBal.receive(newFilter(T6)) doit être bloqué en attente d'un dépôt venant de T6. Il faut dans la structure de ThreadBloqué y mettre aussi le filtre sur lequel attend T5. Il faut aussi y mettre dans cette structure, une réf d'un futur objet où le thread, ici T6, pourra déposer le message demandé par T5 (pour pas qu'un autre thread ne lui prenne).

Un thread qui dépose un message doit, avant de le mettre dans la liste des messages de la boîte aux lettres, parcourir la liste des threads bloqués et vérifier pour l'un eux si son filtre répond à son message. Auquel cas, il doit déposer le message dans la structure du thread bloqué et le réveillé. Le thread réveillé récupère l'objet dans la structure ThreadBloqué qu'il a lui même créé et sort du receive. Sinon lors du dépôt, si aucun thread bloqué ne veut du message, le thread émetteur ajoute son message à la liste des messages de la boîte.

Un thread qui veut un message (receive), doit parcourir la liste des messages de la Bal et tester chaque message avec son filtre (filtre du receive). Si le message correspond avec son filtre, il le retire de la liste. Sinon, il doit créer une structure ThreadBloqué , y mettre son filtre, ajoute cette structure dans la liste des threads bloqués, puis s'endort.

Je vous conseille de vous inspirer de l'implémentation de la classe BlockingQueue vue en cours avec les méthodes Put et Take.