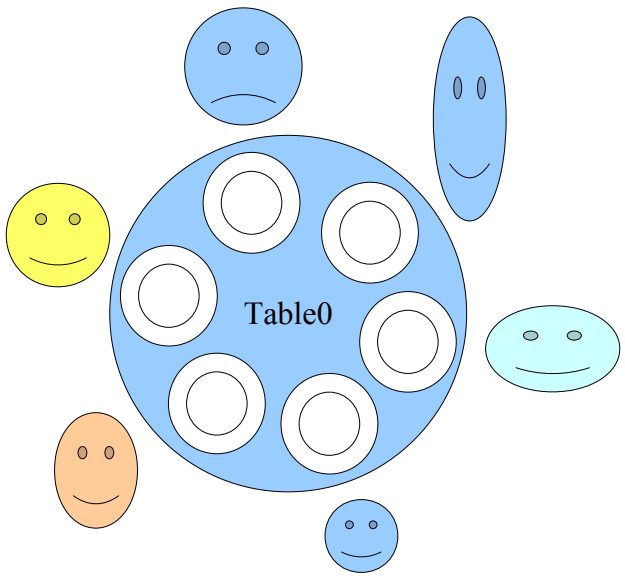
TPs

Le problème (version simplissime)

- Un ensemble de NBP philosophes (NBP vaut 5 typiquement) sont réunis autour d'une table dans un restaurant chinois.

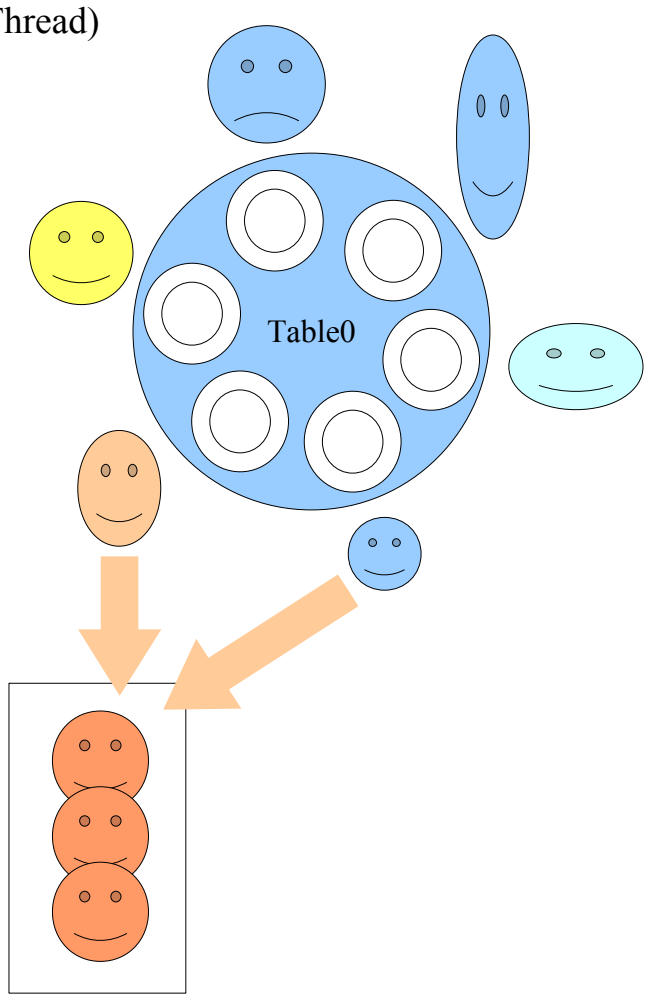
```
loop :
   <réfléchir/discuter> ;
   <manger>
endloop
```



Le problème (version simple)

- Un ensemble de NBP philosophes (NBP vaut 5 typiquement) sont réunis autour d'une table dans un restaurant chinois.
- Un ensemble de NBS serveurs (pour remplir les assiettes toutes les NBI itérations)
- Les serveurs se manifestent via un ExecutorService (un pool de NBS Thread)

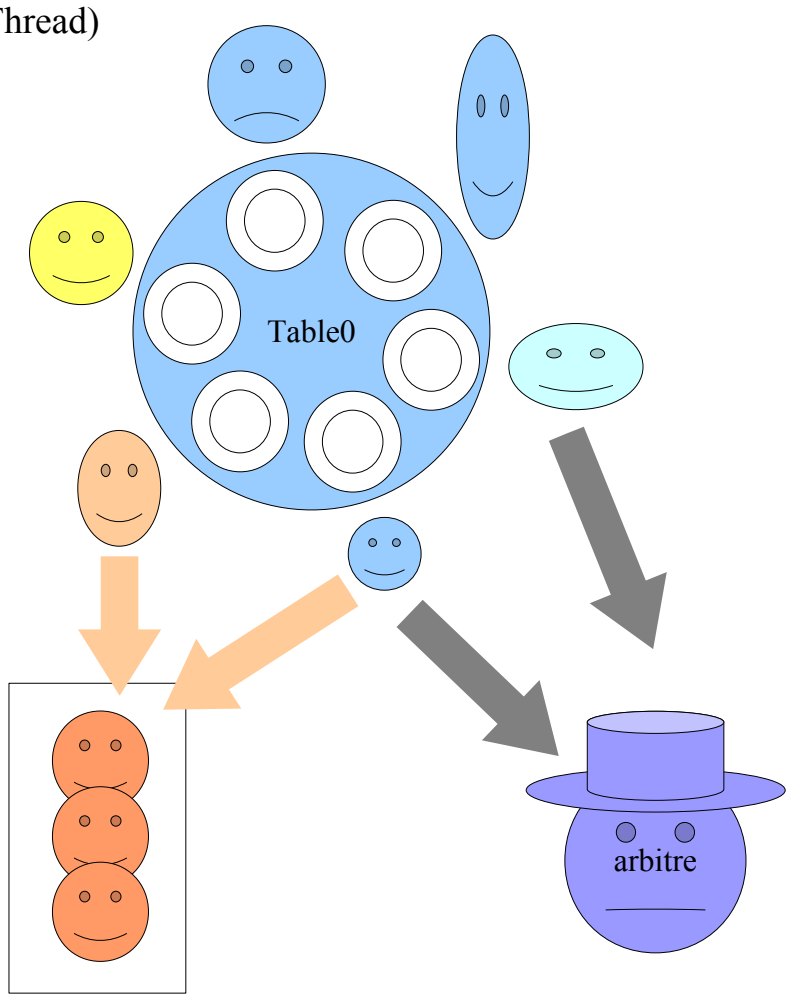
```
loop :
   <soumettre une requete de remplissage de l'assiette>
   <réfléchir/discuter> ;
   <attendre la fin du remplissage de l'assiette>
   <manger>
endloop
```



Le problème (version complète)

- Un ensemble de NBP philosophes (NBP vaut 5 typiquement) sont réunis autour d'une table dans un restaurant chinois.
- Un ensemble de NBS serveurs (pour remplir les assiettes toutes les NBI itérations)
- Les serveurs se manifestent via un ExecutorService (un pool de NBS Thread)
- Un arbitre pour gérer l'allocation des baguettes

```
loop :
   <soumettre une requete de remplissage de l'assiette>
   <réfléchir/discuter> ;
   <attendre la fin du remplissage de l'assiette>
   <obtenir les baguettes>
   <manger>
   <libérer les baguettes>
endloop
```



Le problème (la spécification de la synchronisation)

- Un tableau de boolean indiquant si la baguette(i) est utilisée ou pas : isUsed[]

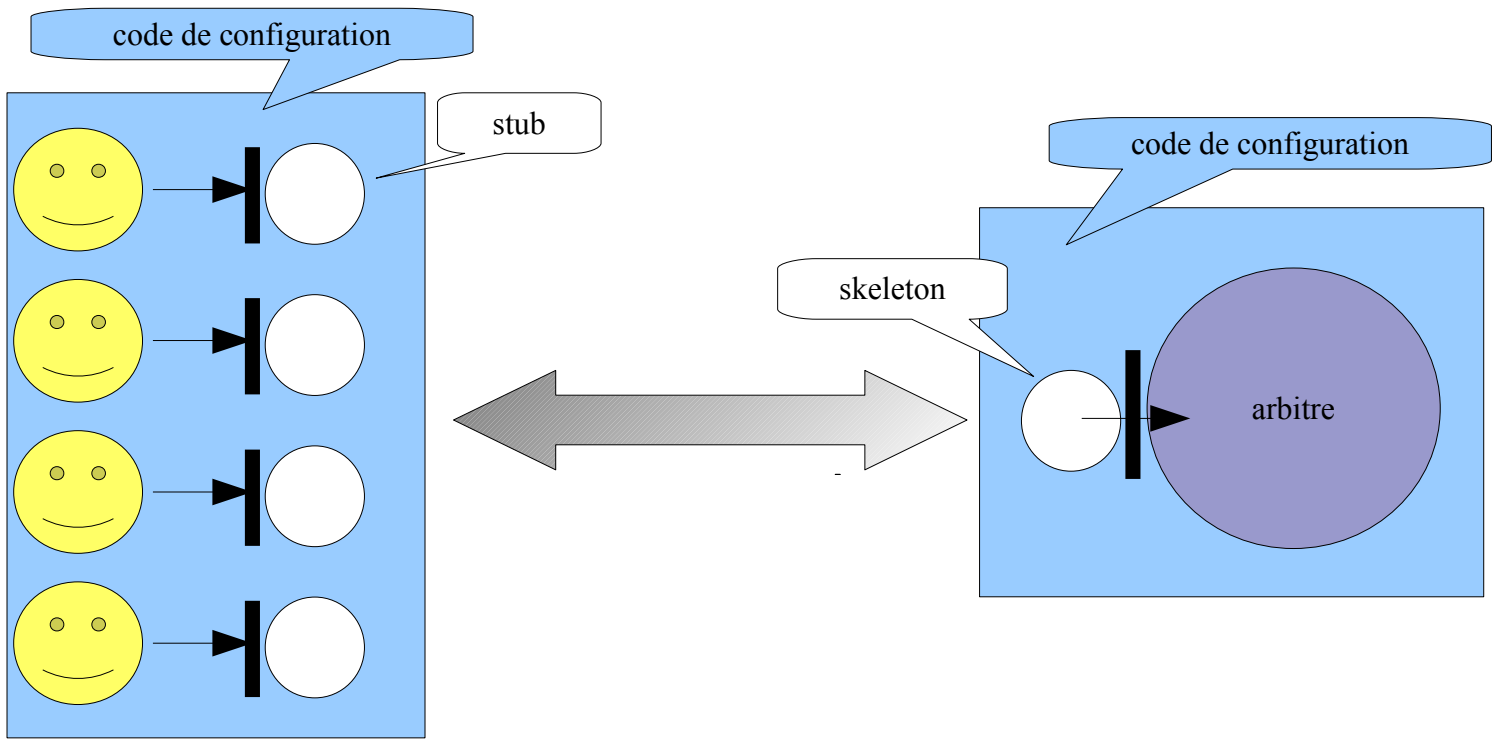
```
loop :
   <soumettre une requete de remplissage de l'assiette>
   <réfléchir/discuter> ;
   <attendre la fin du remplissage de l'assiette>
   ps1(b1, b2) //obtenir les baguettes

   <manger>
   ps1(b1, b2) ://libérer les baguettes b1 et b2
endloop
```

PreW :
CB : isUsed[b1] = true || isUsed[b2] = true
PostW : isUsed[b1] <- true; isUsed[b2] <- true

PreW : isUsed[b1] <- false; isUsed[b2] <- false
CB :
PostW :

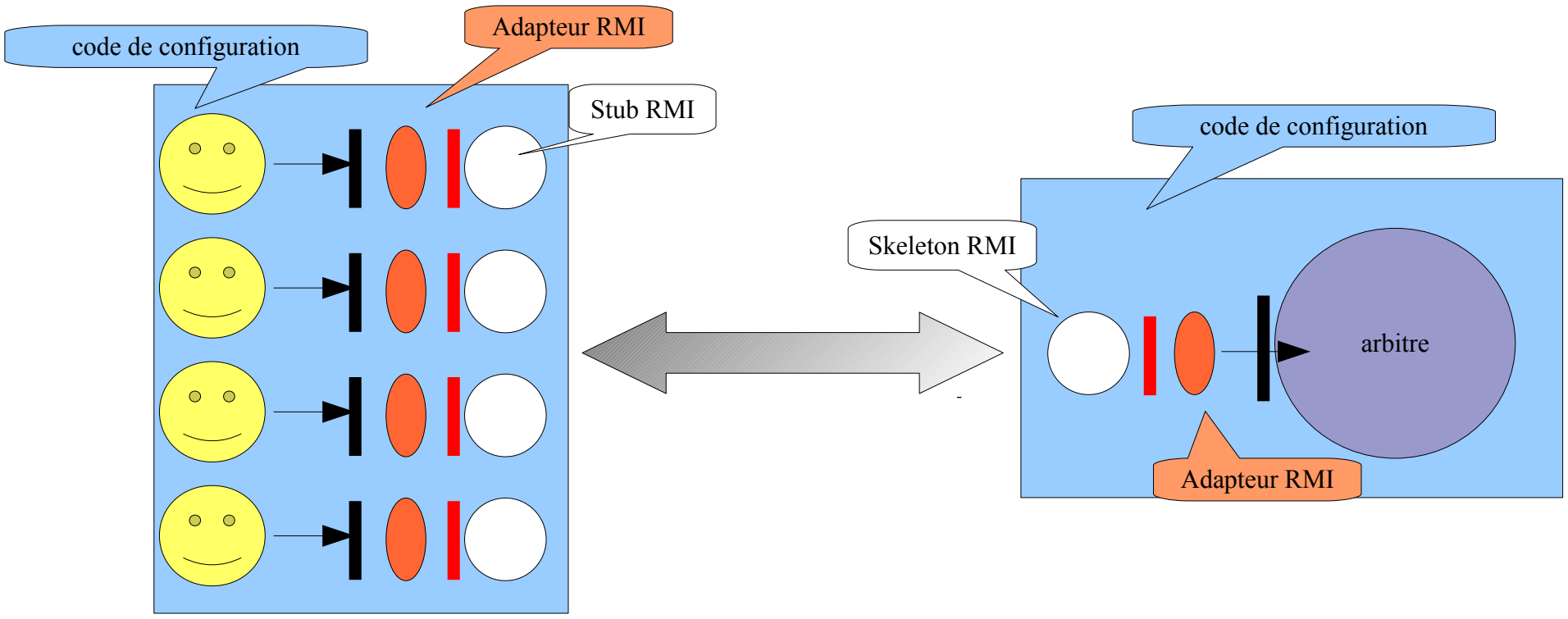
L'architecture applicative

- L'arbitre et les stubs implémentent l'interface de service
- *toutes les références au service utilisent l'interface de service*

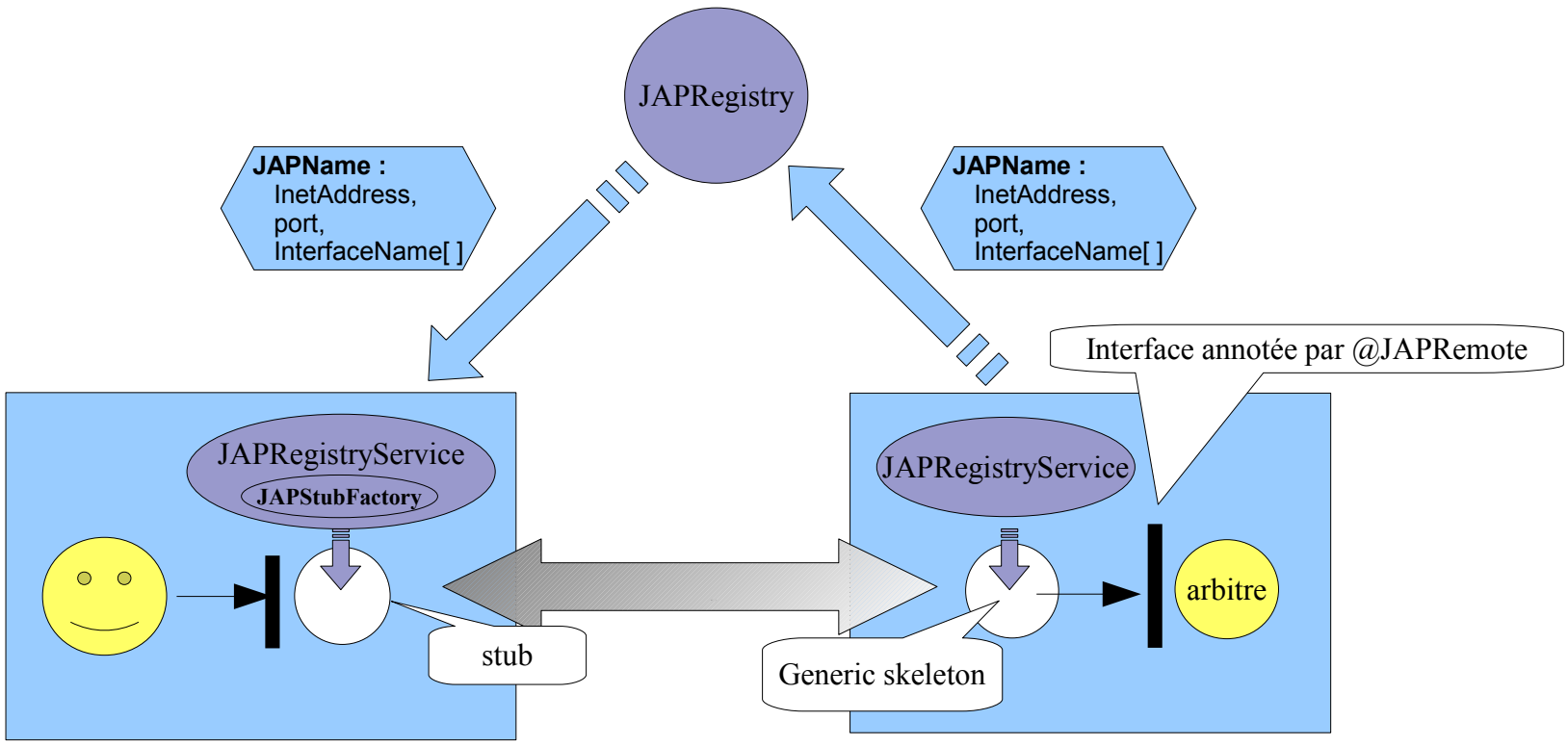


L'adaptation à RMI

- L'arbitre et les stubs implémentent l'interface de service
- *toutes les références au service utilisent l'interface de service*

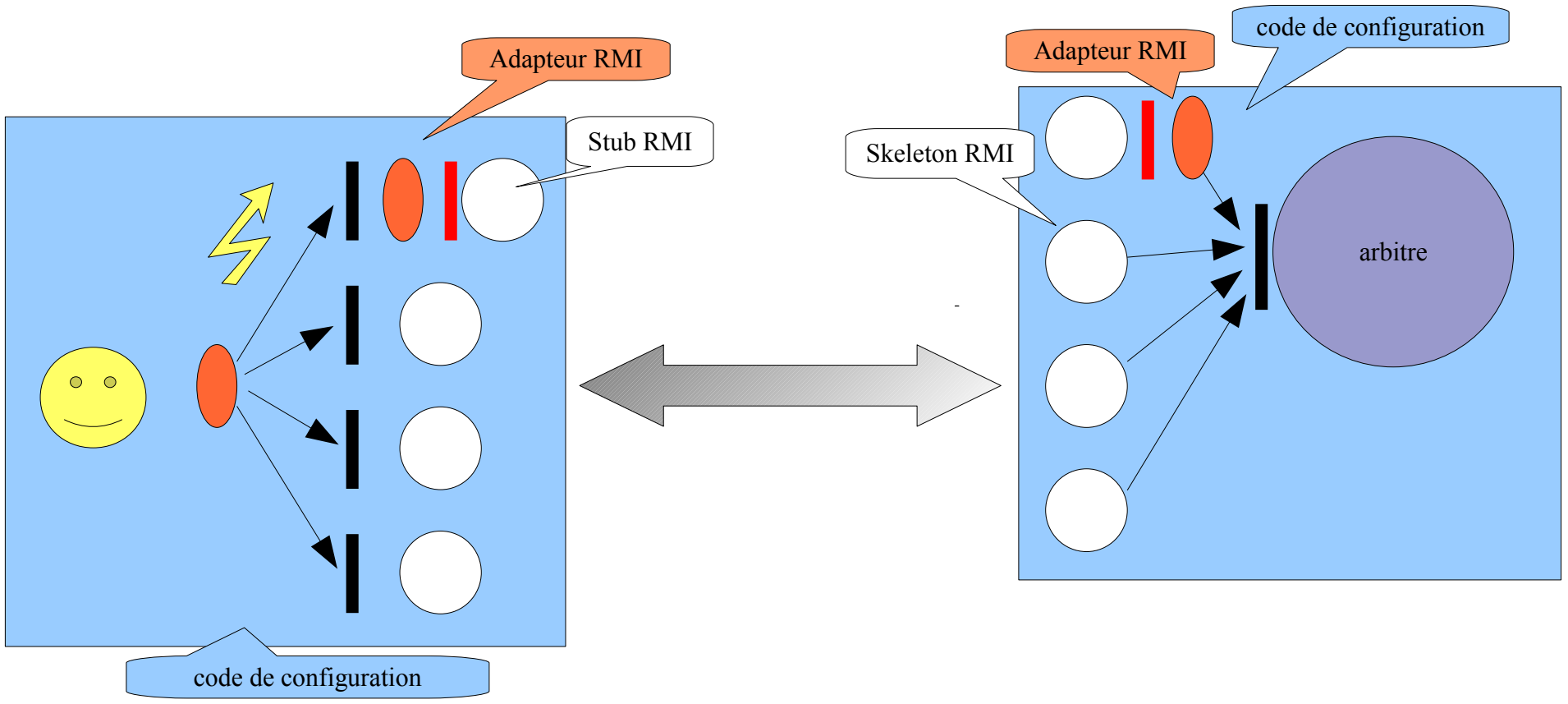


L'architecture de JAP.ORB

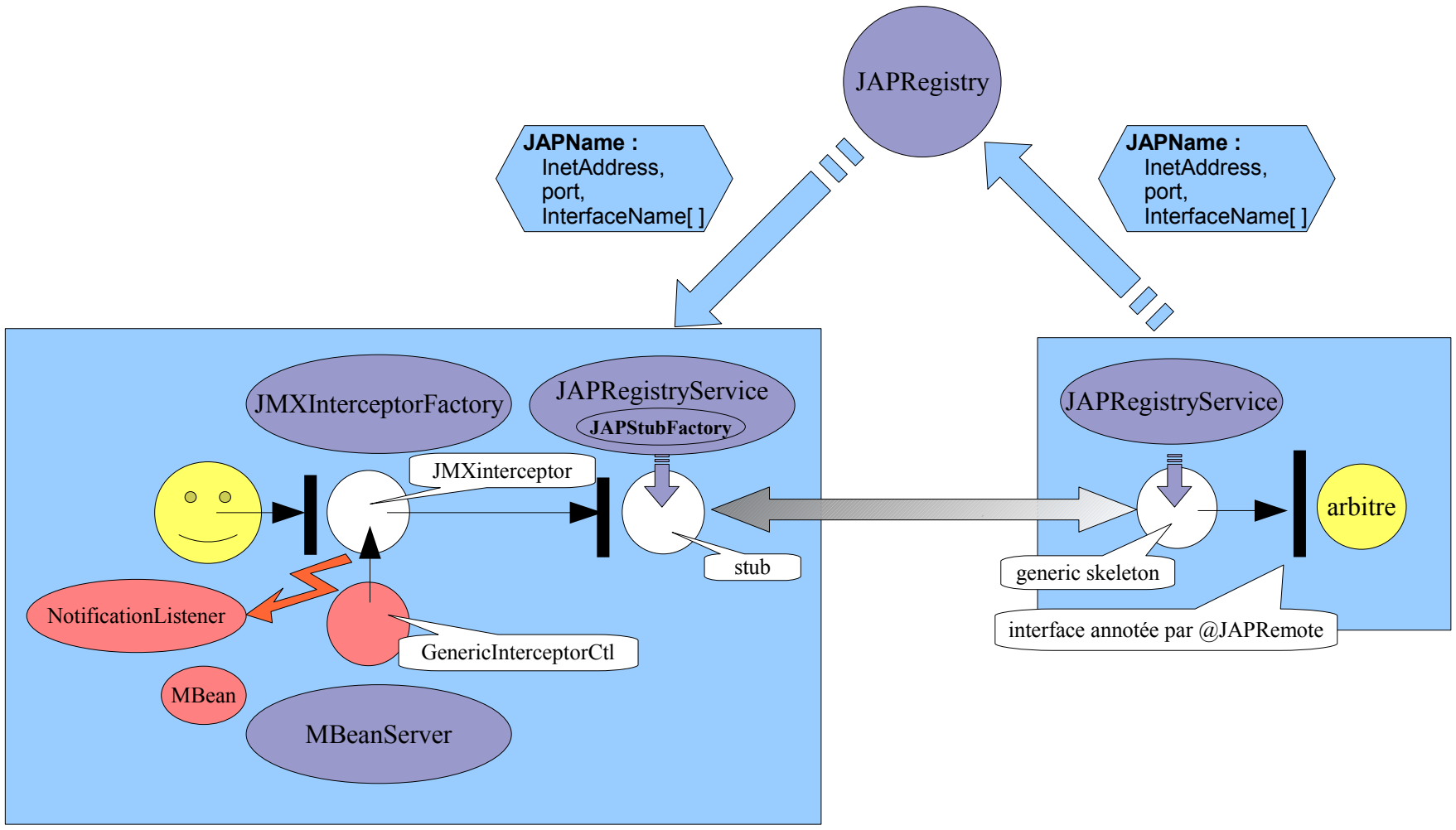


La supervision par JMX

- L'arbitre et les stubs implémentent l'interface de service
- *toutes les références au service utilisent l'interface de service*



L'architecture de JAP.ORB+JMX



Les packages (1)

```
le package thread.philo0 contient les sources de la question n°1 du TP sur les threads
- pas de contrainte sur le comportement des philosophes

le package thread.philo1 contient les sources de la question n°2 du TP sur les threads
- un philosophe est implémenté par 2 threads (Manger et Reflechir)

le package thread.philo2 contient les sources de la question n°3 du TP sur les threads
- introduction d'un ExecutorService sous la forme d'un FixedThreadPool
- introduction d'une ThreadFactory fournissant des threads daemon

le package thread.philoActif contient les sources de la question n°3 du TP sur les threads
- introduction d'une interface IGestBaguettes
- introduction d'un gestionnaire de baguettes (l'état est représenté par un boolean[])

le package thread.philoPassif.base contient les sources de la question n°4 du TP sur les threads
- transformation du gestionnaire de baguettes : synchronisation par wait() et notifyAll()

le package thread.philoPassif.optim contient les sources de la question n°5 du TP sur les threads
- transformation du gestionnaire de baguettes : l'état est représenté par un Baguette[] (pb des Boolean[])

le package thread.philoAsync contient les sources de la question n°6 du TP sur les threads
- introduction d'une interface IPhiloCallback
- transformation des philosophes

le package stream contient les sources de la question n°1 du TP sur les stream
- les classes StRequest et StResponse
- le stub et le skeleton sans possibilité de factorisation

le package stream1 contient les sources d'une solution plus complexe à la question n°1 du TP sur les stream
- les classes StRequest1 et StResponse1
- le stub avec multiplexage des requetes

le package stream.async contient les sources de la question n°2 du TP sur les stream
-  définition du stub et du skeleton de callback (sans multiplexage)
-
```

Les packages (2)

```
le package reflect contient les sources de la question n°1 du TP sur la reflection
- il utilise les classes GenericRequest, GenericResponse et JAPRemoteException du package reflect.common
- définition de GenericSkel
- fourniture du stub

le package reflect.gener contient les sources d'un générateur de stub très simple pour GenericSkel

le package reflect1 contient les sources de la question n°2 du TP sur la reflection
- définition de GenericAnnotSkel
- définition de l'annotation @JAPRemote

le package rmi contient les sources de la question n°1 du TP sur RMI
- introduction de l'interface IRMIGestBaguettes
- définition des adapteurs philo -> stub, skel -> gestionnaire baguettes

le package rmi1 contient les sources de la question n°2 du TP sur RMI (avec philosophes asynchrones)
- introduction de l'interface IRMIGestBaguettes2 et IRMIPhiloCallback
- définition des nouveaux adapteurs pour les callbacks

le package jms contient les sources de la solution du TP sur JMS implémenté à l'aide de files statiques uniquement

le package jms1 contient les sources de la solution du TP sur JMS implémenté à l'aide d'une file temporaire
pour les réponses du gestionnaire de baguettes

le package jmx contient les sources de la solution du TP sur JMX
```

Une courte bibliographie

- R. Gordon : "**Java™ native interface**", Prentice Hall,1998
- E.R. Harold : "**Java network programming**", O'Reilly,1997
- J. Gosling, B. Joy, G. Steele, G. Bracha : "**The Java™ Language Specification, Third Edition"**, The Java Series, Sun Microsystems, 2005
- D. Lea : "**Concurrent Programming in Java™, Design Principles and Patterns"**, The Java Series, Sun Microsystems, 1999
- B. McLaughlin, D. Flanagan : : "**Java 1.5 Tiger : a Developper's Notebook"**, O'Reilly, 2005
- B. Venners : "**Inside the Java Virtual Machine"**, Computing Mc-Graw-Hill, 1998

- Li Gong : "**Inside Java™ 2 Platform Security**", The Java Series, Sun Microsystems, 1998
- S. Oaks : "**Java Security"**, O'Reilly,1998