Maitrise d'Informatique **Module POD**	**Programmation Objet et Distribution** *Documents autorisés*	**Université Pierre et Marie Curie** **Année 1999–2000** *Durée 3 heures*

Examen du 3 février 2000

Exercice : Simulation du modèle de classes d'O'CAML en Java

Les modèles objet d'O'CAML et Java diffèrent principalement par leur système de types, ce qui modifie leur relation de sous-typage. De plus O'CAML accepte l'héritage multiple. Le but de cet exercice est de montrer une traduction possible des classes O'CAML en Java.

1. Donner un exemple d'une classe (ou d'une hiérarchie de classes) O'CAML qui ne peut pas être traduit directement en Java.

2. On traduit une classe (de nom `C`) O'CAML en une interface Java (de nom `I_C`) pour le typage et en une classe Java (de nom `J_C`) pour le code. La classe `J_C` doit implanter l'interface `I_C`. Traduisez en Java les classes O'CAML suivantes selon ce schéma :

```
class point a b   =
  object(self:'a)
    val mutable x = a
    val mutable y = b
    method get_x = x
    method get_y = y
    method rmoveto p q = x <- x + p;  y <- y + q
    method to_string () = "("^(string_of_int x)^","^(string_of_int y)^")"
  end;;

class point_col a b c =
  object
    inherit point a b as super
    val mutable col = c
    method get_col = col
    method to_string () = (super#to_string())^"="^col
  end;;
```

3. Traduisez en Java, dans le corps d'une méthode `main`, le programme O'CAML suivant :

```
  let p1 = new point 2 3;;
  let c2 = new point_col 4 4 "bleu";;
  p1#moveto 1 2;;
  c2#moveto 0 1;;
  print_string( p1#to_string());;
  print_string( c2#to_string());;
  let p3 = (c :> point);;
  print_string (p3#to_string());;
```

4. Indiquer comment traduire l'héritage multiple d'OCAML en Java, en particulier pour rester compatible avec le sous-typage. Illustrer votre solution sur un petit exemple.

5. On ajoute à la classe `point` la méthode `eq`, qui contient le type de `self` comme paramètre :
   `method eq (p:'a) = ( x = p#get_x ) && ( y = p#get_y)`
   Indiquer en O'CAML quel est le type de la méthode `eq` dans les classes `point` et `point_col`.

6. Les classes `point` et `point_col` sont-elles en relation de sous-typage en O'CAML?
7. Indiquer comment traduire en Java les classes contenant des méthodes où apparait le type de la classe dans les types des paramètres. binaires . A-t-on le même comportement vis à vis de la relation de sous-typage? Justifiez votre réponse.

Problème : Service d'enregistement de services

Ce problème se propose de construire un service d'enregistement de processus locaux ou distants. Ces processus sont eux-même des client ou des serveurs.
Le serveur d'enregistrement est unique. Quand un nouveau client apparaît, il s'enregistre auprès du serveur en lui indiquant le nom du service auquel il désire accéder, le numéro du port du service et s'il est serveur ou client de ce service.
Le service d'enregistrement communique alors à tous ses clients déjà enregistrés et concernés par ce service les informations de ce nouvel arrivant. De même le nouvel arrivant reçoit les informations des autres clients enregistrés. Quand un processus termine, il prévient le service d'enregistrement de son départ. Cette information est répercutée aux clients concernés.
Ainsi tous les clients du service d'enregistrement, travaillant sur le même service, possèdent les informations nécessaires à leur propre communication sans passer par le service d'enregistrement. Néanmoins ceux-ci restent en écoute du serveur d'enregistrement pour tenir compte des arrivées et des départs.
Le service d'enregistrement utilise le port `3000` par défaut. Le serveur de ce service s'appelle `recserv.ici.fr`.

Réalisation du service d'enregistrement

1. Définir un protocole choisi pour la communication entre les clients et le serveur du service d'enregistrement pour l'arrivée et le départ d'un client.
2. Écrire la fonction principale du serveur d'enregistrement en O'CAML pour que celui-ci maintienne à jour la liste de ses clients et répercute à ceux-ci les informations les concernant.
3. Écrire un client générique, en Java ou en O'CAML, valable pour tout type de service, qui établit le contact avec la machine `recserv.ici.fr` sur le port `3000` et reste en écoute pour mettre à jour ses tables. Ce client indique s'il est client ou serveur du service proposé et active le processus client ou serveur de ce service. Lorsque ce client termine, il le signale au serveur d'enregistrement.

Réalisation d'une application client/serveur On cherche à écrire une application d'envoi de messages à tous les participants actifs d'un forum. Un message contient le nom de l'expéditeur sur une ligne suivi immédiatement d'une ligne de texte. Un message est envoyé d'un client vers le serveur de messagerie. Ce dernier répercute ce message à tous ses clients. Le serveur de messagerie connaît ses clients grâce au service d'enregistrement. La messagerie est connue sous le nom de service `MESSAGE` et utilise par défaut le port `2000`.

4. Indiquer les communications entre le serveur d'enregistrement, un serveur de messagerie et trois clients de messagerie. Vous préciserez les protocoles utilisés.
5. Écrire un client de messagerie, en Java ou O'CAML, en l'intégrant au client générique du service d'enregistrement. Suite à son enregistrement, si les informations transmises par le serveur d'enregistrement contiennent le nom d'un serveur de messagerie, alors le client s'y connecte. Si ce n'est pas le cas, il se met en attente de l'apparition d'un tel serveur. Si plusieurs serveurs de messagerie sont actifs, le client en choisit un.
6. Écrire le serveur de cette messagerie, en l'intégrant au client générique du service de messagerie.

Extensions des fonctionalités du service enregistrement Pour être tolérant aux arrêts d'une machine sur le réseau, un enregistement n'est valide que pour un certain temps. Il devient alors nécessaire de confirmer son enregistrement à intervalle de temps régulier.

7. Modifier le protocole, pour introduire les confirmations d'enregistrement.
8. Implanter le mécanisme de limite de temps du côté du serveur d'enregistrement. On utilisera la fonction O'CAML `Unix.time` qui appliquée à `()` retourne le nombre de secondes écoulées depuis le 1er janvier 1970.