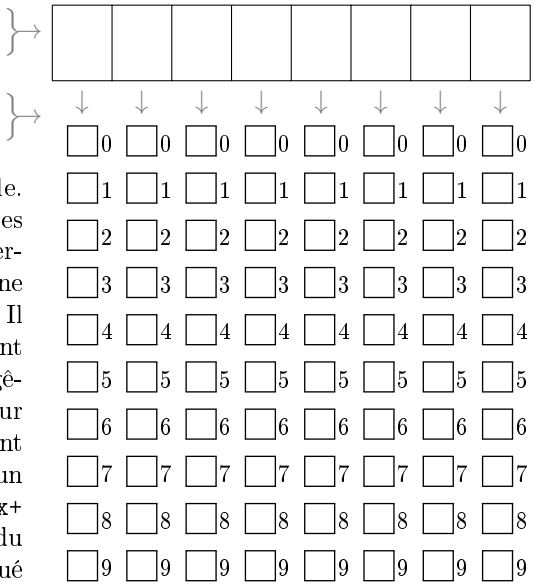
Correction

Université d'Évry Val d'Essonne
Master 1 Info/MIAGE
2019-2020

Partiel de Systèmes et Applications Répartis (SAR)
Durée : 2h30

---

1. Inscrivez ci-contre votre numéro d'étudiant, en écrivant un chiffre par case.
2. Reportez un chiffre par colonne en cochant la case correspondante.



Répondez directement sur l'énoncé et rendez le. Lisez *attentivement* l'énoncé. Les documents, les calculatrices et les téléphones mobiles sont interdits. Toutes les questions comportent *au moins* une bonne réponse et *au moins* une mauvaise réponse. Il est préférable de noircir les cases en les remplissant plutôt que de simplement les cocher, il n'est pas gênant de déborder. Vous pouvez utiliser du correcteur blanc pour effacer une réponse, il n'est pas gênant alors de masquer complètement la case. Utilisez un stylo à encre, de préférence noire. Le code `+1/p/x+` en haut de page indique qu'il s'agit de la page `p` du sujet numéro 1, vérifiez qu'on vous a bien distribué les 8 pages de votre sujet (numéro 1).

---

1 Questions de cours (4pts)

Question 1 ♣
Un système distribué est :

- ☐ un ensemble de processus complètement indépendants
- ☐ un système avec une architecture producteur/consommateur
- ☐ un système avec une architecture client/serveur
- ☐ un système monolithique
- ☐ un système réactif
- ☐ un système temps réel
- ☐ un système dont les données sont distribuées
- ■ un système dont les traitements sont distribués
- ■ un ensemble de processus coopératifs

Question 2 ♣
Quelles sont les différentes formes de distribution des systèmes ?

- ☐ distribution conceptuelle
- ■ distribution structurelle
- ☐ distribution logique
- ☐ distribution synchrone
- ☐ distribution linéaire
- ■ distribution fonctionnelle
- ■ distribution physique
- ☐ distribution événementielle
- ☐ distribution temporelle
- ☐ *Aucune de ces réponses n'est correcte.*

2 Workflow et Synchronisation (9pts)

Soit les trois processus suivants :

P1 = while(true) {B; V; N;}

P2 = {R; M;}

P3 = while(true) {A; D; K;}

Nous voulons réaliser le Workflow représenté par la figure 1 en respectant la forme itérative des processus P1 et P3.

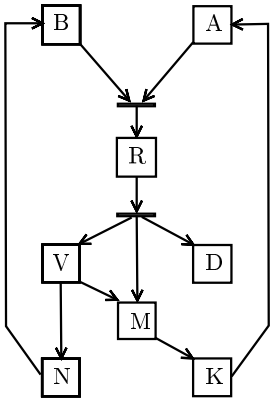


Figure 1: Le Workflow des trois processus

Nous allons utiliser la technique du *JobController* vu en cours et en TD (donné dans la figure 2) afin de réaliser l'application répartie définie par le workflow de la figure 1.

```
34 public class JobController {
35     private boolean done;
36     public JobController() {
37         done = false;
38     }
39     synchronized public void jobDone() {
40         done = true;
41         notifyAll();
42     }
43     synchronized public void isJobDone() {
44         while (!done) { try { wait(); }catch(Exception e){}
45         }
46     }
47 }
```

Figure 2: La classe Java du JobController

CORRECTION

Question 3
Étant données trois instances de processus, respectivement de type P1, P2 et P3, de combien d'instances de *JobController* avons-nous besoin pour réaliser le Workflow de la figure 1 (selon les règles vues en cours et en TD) ? Choisissez le bon nombre parmi les propositions ci-dessous.

Nombre d'instances nécessaires = ☐1 ☐2 ☐3 ☐4 ☐5 ■6 ☐7 ☐8 ☐9

Question 4
Combien de séquences (traces) possibles donne le workflow de la figure 1 lors de la **première itération** ? Codez votre réponse à l'aide des cases ci-dessous.

ligne 1 pour les dizaines : ☐0 ☐1 ■2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9
ligne 2 pour les unités : ☐0 ☐1 ■2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9

Question 5 ♣
Parmi les séquences (traces) ci-dessous, lesquelles respectent le workflow de la figure 1 à partir de la **deuxième itération** des processus P1 et P3 ?

- ☐ BARVDNMK
- ■ ABVDKANB
- ☐ ABRDVMKN
- ■ KBVANDBK
- ☐ BAVNDKADV
- ■ BAVNDKADK
- ☐ BARVDMNK
- ■ BVNBADK
- ■ DKNBVAD

Question 6 ♣
Quelles sont les modifications qu'on doit apporter au contrôleur de la figure 2 afin de l'adapter à la forme itérative des deux processus P1 et P3 ?

- ☐ rendre le constructeur synchronized
- ☐ initialiser la nouvelle variable d'itération à faux
- ☐ définir une nouvelle méthode itérative
- ☐ ajouter un compteur entier pour l'itération
- ☐ initialiser la nouvelle variable d'itération à vrai
- ■ Aucune modification à faire
- ☐ ajouter une autre variable booléenne
- ☐ modifier le constructeur en ajoutant un paramètre (done)
- ☐ initialiser la variable done à vrai
- ☐ mettre done à false dans isJobDone après le while
- ☐ incrémenter le compteur dans jobDone
- ☐ ajouter notifyAll() dans isJobDone après le while

Question 7 ♣

On souhaite écrire la classe du processus P1. Soit l'implémentation Java ci-dessous. Les contrôleurs déclarés permettent la synchronisation entre les tâches du workflow de la figure 1 selon la correspondances suivantes : a=BR, b=AR, c=RV, d=RM, e=RD, f=VN, g=VM, h=MK, i=NB, j=KA. Les tâches 1, 2 et 3 indiquées dans cette implémentation correspondent respectivement aux tâches B, V et N du processus P1 du workflow.

```
34 public class P1 extends Thread {
35     private JobController a;
36     private JobController b;
37     private JobController c;
38     private JobController d;
39     private JobController e;
40     private JobController f;
41     private JobController g;
42     private JobController h;
43     private JobController i;
44     private JobController j;
45     public P1(JobController x1, ...) {
46         ... // initialisation des controlleurs
47     }
48     public void run() {
49       while (true) {
50         i.isJobDone();
51         a.isJobDone();
52         b.isJobDone();
53         tache("1");
54         i.jobDone();
55         a.jobDone();
56         b.jobDone();
57         c.isJobDone();
58         f.isJobDone();
59         g.isJobDone();
60         tache("2");
61         c.jobDone();
62         f.jobDone();
63         g.jobDone();
64         f.isJobDone();
65         h.isJobDone();
66         i.isJobDone();
67         j.isJobDone();
68         tache("3");
69         f.jobDone();
70         h.jobDone();
71         i.jobDone();
72         j.jobDone();
73       }
74     }
75 }
```

Sélectionnez ci-dessous **uniquement** les lignes nécessaires à la définition de la classe du processus P1.

☐ 50	■ 57	☐ 70	☐ 43	☐ 40	■ 41	■ 55
☐ 38	☐ 66	☐ 67	☐ 64	☐ 58	☐ 52	■ 68
☐ 71	☐ 62	☐ 54	☐ 72	■ 63	☐ 36	■ 53
☐ 59	☐ 69	☐ 51	☐ 44	☐ 61	■ 37	■ 49
☐ 39	■ 60	☐ 56	☐ 42	■ 35	☐ 65	■ 48

3 CORBA (7pts)

Les trois processus P1, P2 et P3 de l'exercice précédent sont répartis sur trois sites différents (P1 sur le site 1, P2 sur le site 2 et P3 sur le site 3) et ne sont pas implémentés avec le même langage de programmation. Seul le processus P3 utilise le langage JAVA comme langage de programmation.
Nous voulons toujours réaliser le workflow représenté par la figure 1. Nous allons donc utiliser à la fois la technique du JobController et les objets répartis distants à travers CORBA.

Question 8
En vous basant sur le code de la figure 3, complétez le contrat IDL des objets distants qui permettent la synchronisation des trois processus P1, P2 et P3. Respectez les noms des packages, des classes, des interfaces et des méthodes.

☐z ☐a ☐b ■c *Cadre reservé*

```
34 module
35 {
36
37
38
39
40
41
42
43
44 };
```

Question 9
En vous basant sur le code de la figure 3, complétez l'implémentation de la classe des objets distants *JobControllerImpl* (Servant). Respectez les noms des packages, des classes, des interfaces et des méthodes.

☐z ☐a ☐b ☐c ■d *Cadre reservé*

```
34 public class JobControllerImpl
35 {
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53 }                                                                    .
```

Une instance de *JobControllerImpl* reliant deux tâches X et Y, notée XY, sera déployée sur le site de chaque tâche source X. Par exemple, l'instance du *JobControllerImpl* qui contrôle la synchronisation entre les tâches M et K, notée MK, sera déployée sur le site 2.

Question 10
Combien d'instances de *JobControllerImpl* seront déployées sur le site 1 ?

Nombre d'instances du site 1 : ☐0 ☐1 ■2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9

Question 11
Combien d'instances de *JobControllerImpl* seront déployées sur le site 2 ?

Nombre d'instances du site 2 : ☐0 ☐1 ☐2 ■3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9

Question 12
Combien d'instances de *JobControllerImpl* seront déployées sur le site 3 ?

Nombre d'instances du site 3 : ☐0 ■1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9

Question 13 ♣

On souhaite écrire le serveur du site 3. Soit l'implémentation Java de la figure 3. Les contrôleurs créés permettent la synchronisation entre les tâches du workflow de la figure 1 selon la correspondances suivantes : a=BR, b=AR, e=RD, h=MK, j=KA.

Sélectionnez ci-dessous **uniquement** les lignes nécessaires à la définition du serveur du site 3, la classe des processus P3 étant donnée.

■ 68	☐ 62	■ 65	☐ 54	☐ 59	☐ 52
☐ 48	■ 51	■ 64	☐ 63	■ 67	☐ 60
☐ 66	■ 50	☐ 55	☐ 58	☐ 53	■ 61
☐ 57	☐ 56	☐ 47	☐ 46	■ 49	

```
34 import Controller.IController;
35 import Controller.IControllerHelper;
36 public class P3Serveur {
37    public static void main(String[] args) throws Exception {
38       Properties props = new Properties();
39       props.put("org.omg.CORBA.ORBInitialHost", "localhost");
40       props.put("org.omg.CORBA.ORBInitialPort", "8888");
41       ORB orb = ORB.init(args, props);
42       org.omg.CORBA.Object oc = orb.resolve_initial_references("RootPOA");
43       POA rootPOA = POAHelper.narrow(oc); rootPOA.the_POAManager().activate();
44       org.omg.CORBA.Object oc2 = orb.resolve_initial_references("NameService");
45       NamingContextExt context = NamingContextExtHelper.narrow(oc2);
46       org.omg.CORBA.Object oa = rootPOA.servant_to_reference(new JobControllerImpl());
47       IController a = IControllerHelper.narrow(oa);
48       context.rebind(context.to_name("aController"), a);
49       org.omg.CORBA.Object ob = rootPOA.servant_to_reference(new JobControllerImpl());
50       IController b = IControllerHelper.narrow(ob);
51       context.rebind(context.to_name("bController"), b);
52       org.omg.CORBA.Object oe = rootPOA.servant_to_reference(new JobControllerImpl());
53       IController e = IControllerHelper.narrow(oe);
54       context.rebind(context.to_name("eController"), e);
55       org.omg.CORBA.Object oh = rootPOA.servant_to_reference(new JobControllerImpl());
56       IController h = IControllerHelper.narrow(oh);
57       context.rebind(context.to_name("hController"), h);
58       org.omg.CORBA.Object oj = rootPOA.servant_to_reference(new JobControllerImpl());
59       IController j = IControllerHelper.narrow(oj);
60       context.rebind(context.to_name("jController"), j);
61       orb.run();
62       IController a = (IController) context.resolve_str("aController");
63       IController b = (IController) context.resolve_str("bController");
64       IController e = (IController) context.resolve_str("eController");
65       IController h = (IController) context.resolve_str("hController");
66       IController j = (IController) context.resolve_str("jController");
67       ProcessusP3 p3 = new ProcessusP3(...);
68       p3.run();
69    }
70 }
```

Figure 3: La classe Java du serveur du site 3

```
#include <date.idl>
module annuaire {
  typedef string Nom;
  typedef sequence<Nom> DesNoms;
  struct Personne {
    Nom nom;
    string telephone;
    ::date::Date date_naissance;
  };
  typedef sequence<Personne> DesPersonnes;
  readonly attribute string libelle;
  exception ExisteDeja { Nom nom; };
  exception Inconnu { Nom nom; };
  interface AdministrationAnnuaire {
    void ajouterPersonne (in Personne personne) raises(ExisteDeja);
    void retirerPersonne (in Nom nom) raises(Inconnu);
  };
  interface ConsultationAnnuaire {
    Personne obtenirPersonne (in Nom nom) raises(Inconnu);
    DesNoms listerNoms ();
  };
};
```

Figure 4: Exemple d'une description d'interfaces IDL