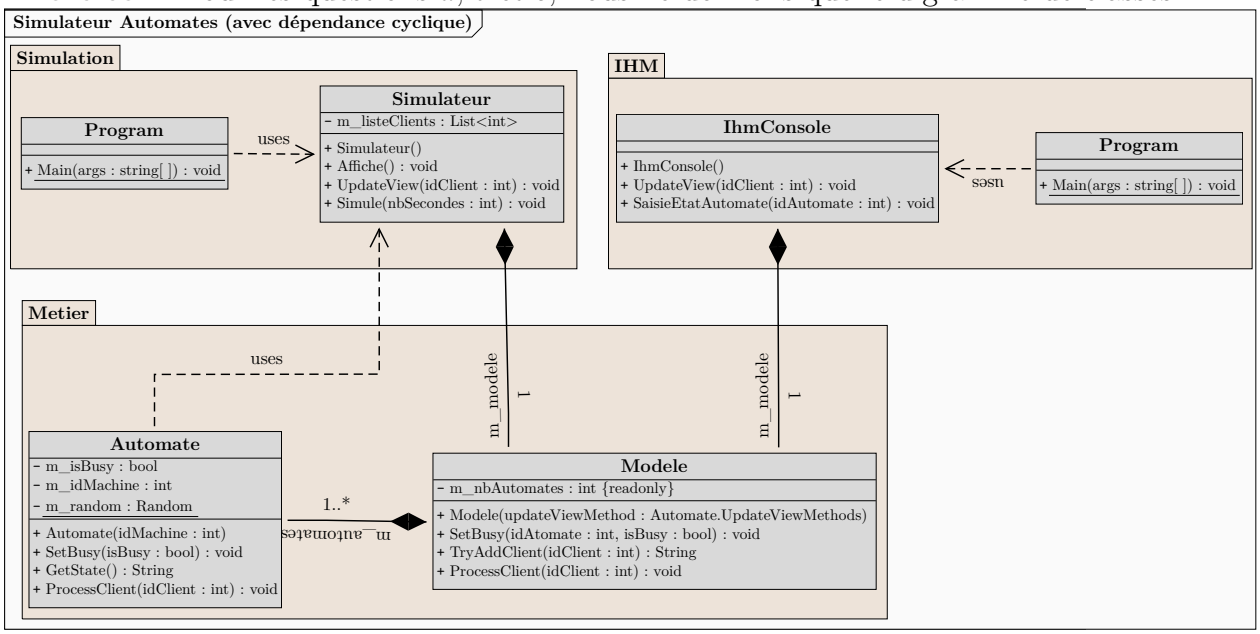
Université d'Auvergne
Clermont-Ferrand Aurillac Le Puy-en-Velay

Conception Objet et Programmation *C#*,
IUT département Informatique,
1ère année, 2015-2016

Conception Objet et Programmation en *C#*

Corrigé du *DM* n° 1
Delegates et Découplage

Exercice 1 Pour les questions *a*, *b* et *c*, nous ne donnons que le digramme de classes :



Diag 1. Le diagramme de classe avec l'*IHM* (questions *a* et *b*)). Problème de dépendance cyclique entre `Modele` et `Simulateur`.

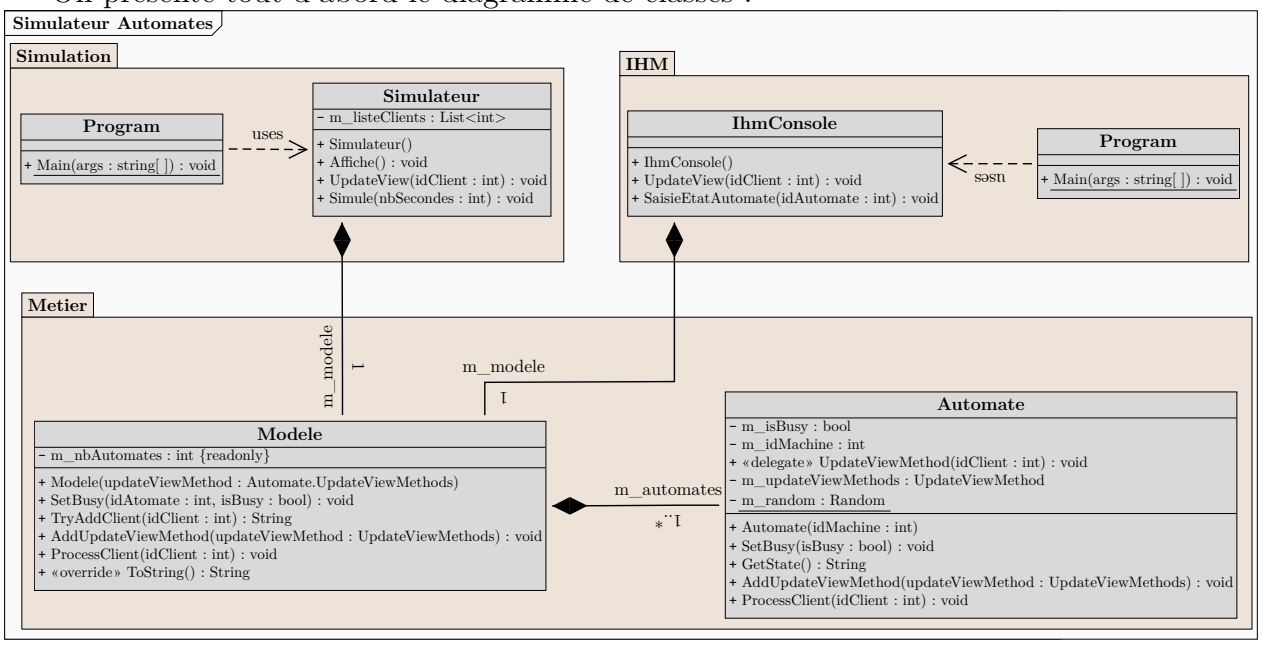
d) Sur le même diagramme 1, nous présentons la tentative pour modéliser le simulateur. Il se présente le problème suivant :

1. Le simulateur appelle `Modele.TryAddClient` qui appelle (si un automate est libre) `Automate.Proces`
2. La méthode `Automate.ProcessClient` a une durée aléatoire (pendant laquelle le simulateur continue de travailler), et se déroule pour celà dans un thread.
3. À la fin de la méthode `Automate.ProcessClient`, l'automate redevient libre, et la vue doit donc être mise à jour. Comme le simulateur ne peut pas attendre la fin de `Automate.ProcessClient` pour mettre à jour la vue lui-même (car il doit continuer à traiter les autres clients), le plus simple est que l'automate lui-même, à la fin de

`Automate.ProcessClient`, demande la mise à jour de la vue en appelant `Simulateur.UpdateView`. Cependant, comme le suggère l'énoncé, cela crée une dépendance cyclique entre les packages `Simulation` et `Metier`.

e) L'idée de la solution est la suivante : Nous stockerons dans chaque automate, dans une donnée de type *delegate*, la référence vers la méthode de mise à jour de la vue (`Simulation.Simulateur.UpdateView()` ou `IHM.IhmConsole.UpdateView()`, selon le cas). Cette référence vers la méthode de mise à jour de la vue sera passé via le constructeur de `Modele`, au constructeur d'`Automate`. Ensuite, lorsque l'automate doit mettre à jour la vue à la fin de la méthode `Automate.ProcessClient`, il invoque le délégué, ce qui provoque l'appel de la méthode de mise à jour de la vue dont la référence se trouve dans le délégué en question. Ceci constitue une *inversion de dépendances*, car la méthode de mise à jour de la vue est passée au constructeur de modèle (*package* `Metier`) à partir de simulateur (*package* `Simulation`), mais aucune classe du *package* `Simulation` n'utilise directement le *package* `Metier`. L'inversion de dépendances est un exemple de *découplage*, par lequel on réduit la dépendance entre des *packages* ou des classes.

On présente tout d'abord le diagramme de classes :



Diag 2. Le diagramme de classe avec l'*IHM* et le simulateur (question *e*)), basés sur le même modèle (même package `Metier` pour les deux applications).

Voici maintenant le code correspondant :

testCSharp/Automate.cs

```
1  using System;
2  using System.Threading;
3  using System.Threading.Tasks;
4
5  namespace Metier
6  {
7   public class Automate
8   {
9      // true si l'automate est occupé, false sinon
10     private bool m_isBusy;
```

```
11
12      private int m_idMachine;
13
14      // Type délégué pour les méthodes de mise à jour de vues.
15      // La mise à jour de la vue utilise l'ID du client traité
16      public delegate void UpdateViewMethods(int idClient);
17
18      // Instance de type délégué avec les méthodes pour
19      // mettre à jour la vue
20      private UpdateViewMethods m_updateViewMethods;
21
22      private static Random m_random = new Random();
23
24      public void SetBusy(bool isBusy)
25      {
26        m_isBusy = isBusy;
27      }
28
29      public Automate(int idMachine)
30      {
31        m_isBusy = false;
32        m_idMachine = idMachine;
33      }
34
35      // Ajoute une méthode de mise à jour de vue
36      // (il peut y en avoir plusieurs
37      public void AddUpdateViewMethod(UpdateViewMethods updateViewMethod)
38      {
39        m_updateViewMethods += updateViewMethod; // Ajout de la méthode dans le
              delegate
40      }
41
42      // Méthode qui fait un affichage après un certain temps
43      // Nombre de millisecondes passé en paramètre.
44      public void ProcessClient(int idClient)
45      {
46        SetBusy(true);
47        int nMilliseconds = m_random.Next(8001);
48        Task.Run( () => {
49                              // Code du thread
50                              Thread.Sleep(nMilliseconds);
51                              // SetBusy(false);
52                              // Invocation du delegate
53                              // de mise à jour de la vue
54                              Console.Write("Client␣{0}␣par␣machine␣{1}␣terminé.␣"
                                    , idClient, m_idMachine);
55                              SetBusy(false);
56                              m_updateViewMethods(idClient);
57                            } );
58      }
59
60      public String GetState()
61      {
62        return m_isBusy ? "O" : "L";
63      }
64  }
```

```
65 }
```

testCSharp/Modele.cs

```
1  using System;
2  using System.Text;
3
4  namespace Metier
5  {
6   public class Modele
7   {
8       private Automate[] m_automates;
9
10      private readonly int m_nbAutomates = 4;
11
12      public void SetBusy(int idAutomate, bool isBusy)
13      {
14        m_automates[idAutomate].SetBusy(isBusy);
15      }
16
17      public Modele(Automate.UpdateViewMethods updateViewMethod)
18      {
19        m_automates = new Automate[m_nbAutomates];
20        for (int i=0 ; i<m_nbAutomates ; i++)
21        {
22          m_automates[i] = new Automate(i);
23          m_automates[i].AddUpdateViewMethod(updateViewMethod);
24        }
25      }
26
27      // Essaie de traiter le client si un automate est libre
28      // retourne faux si tous les automates sont occupés.
29      public bool TryAddClient(int idClient)
30      {
31
32        for (int i=0 ; i<m_nbAutomates ; i++)
33        {
34          if (m_automates[i].GetState() == "L")
35          {
36            Console.Error.WriteLine("Client {0} traité par machine {1}", idClient,
                  i);
37            m_automates[i].ProcessClient(idClient);
38            return true;
39          }
40        }
41        return false;
42      }
43
44      public override string ToString()
45      {
46        StringBuilder sb = new StringBuilder("----------\n
              Automate ");
47        for (int i=0 ; i<m_nbAutomates ; i++)
48        {
49          sb.Append("| " + i + " ");
50        }
```

```
51          sb.Append("|\n-------------------------------\n␣␣␣État␣␣␣");
52          for (int i=0 ; i<m_nbAutomates ; i++)
53          {
54            sb.Append("|␣" + m_automates[i].GetState() + "␣");
55          }
56          sb.Append("|\n-------------------------------\n");
57          return sb.ToString();
58        }
59    }
60  }
```

testCSharp/Simulateur.cs

```
1   using System;
2   using System.Text;
3   using System.Threading;
4   using System.Collections.Generic;
5   using System.Linq;
6
7   using Metier;
8
9   namespace Simulation
10  {
11    public class Simulateur
12    {
13      private Modele m_modele;
14      private List<int> m_listeClients;
15
16      public Simulateur()
17      {
18        m_modele = new Modele(UpdateView);
19
20        m_listeClients = new List<int>();
21      }
22
23      public void Affiche()
24      {
25        Console.Error.WriteLine(m_modele);
26      }
27
28      public void UpdateView(int idClient){
29        Console.Error.WriteLine("Le␣client␣numéro␣{0}␣a␣été␣traité.\n", idClient);
30
31        Affiche();
32
33        if (m_listeClients.Count > 0){
34          int nextClient = m_listeClients.ElementAt(0);
35          Console.Error.WriteLine("Client␣numéro␣{0}␣sort␣la␣file␣d'attente.\n",
                nextClient);
36          m_listeClients.RemoveAt(0);
37          if (!m_modele.TryAddClient(nextClient)){
38            Console.Error.WriteLine("Client␣numéro␣{0}␣rentre␣dans␣la␣file␣d'
                  attente.\n", nextClient);
39            m_listeClients.Add(nextClient);
40          }
41        }else{
```

```
42          Console.Error.WriteLine("File d'attente vide...");
43        }
44
45        Affiche();
46      }
47
48      public void Simule(int nSecondes){
49        for (int i=0 ; i<nSecondes ; i++){
50          if (!m_modele.TryAddClient(i)){
51            Console.Error.WriteLine("Client numéro {0} rentre dans la file d'
                  attente.\n", i);
52            m_listeClients.Add(i);
53          }
54          Thread.Sleep(1000);
55        }
56        // On attend que tout soit fini
57        Thread.Sleep(10000);
58      }
59
60    }
61  }
```

testCSharp/programCorrige.cs

```
1  using System;
2
3  using Metier;
4  using Simulation;
5
6  namespace ApplicationConsole
7  {
8   public class Program
9   {
10   // Le Main
11   public static void Main(string[] args)
12   {
13     Simulateur simulateur = new Simulateur();
14   simulateur.Simule(10);
15   }
16
17  }
18 }
```

et la trace :

```
Client 0 traité par machine 0
Client 1 traité par machine 1
Client 2 traité par machine 2
Client 3 traité par machine 3
Client 0 par machine 0 terminé. Le client numéro 0 a été traité.

---------------------------
 Automate | 0 | 1 | 2 | 3 |
---------------------------
```

```
   État   | L | O | O | O |
--------------------------

File d'attente vide...
--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | L | O | O | O |
--------------------------

Client 3 par machine 3 terminé. Le client numéro 3 a été traité.

--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | L | O | O | L |
--------------------------

File d'attente vide...
--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | L | O | O | L |
--------------------------

Client 1 par machine 1 terminé. Le client numéro 1 a été traité.

--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | L | L | O | L |
--------------------------

File d'attente vide...
--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | L | L | O | L |
--------------------------

Client 4 traité par machine 0
Client 2 par machine 2 terminé. Le client numéro 2 a été traité.

--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | O | L | L | L |
```

```
--------------------------

File d'attente vide...
--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | O | L | L | L |
--------------------------

Client 5 traité par machine 1
Client 6 traité par machine 2
Client 7 traité par machine 3
Client 5 par machine 1 terminé. Le client numéro 5 a été traité.

--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | O | L | O | O |
--------------------------

File d'attente vide...
--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | O | L | O | O |
--------------------------

Client 8 traité par machine 1
Client 4 par machine 0 terminé. Le client numéro 4 a été traité.

--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | L | O | O | O |
--------------------------

File d'attente vide...
--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | L | O | O | O |
--------------------------

Client 6 par machine 2 terminé. Le client numéro 6 a été traité.

--------------------------
 Automate | 0 | 1 | 2 | 3 |
```

```
--------------------------
   État   | L | O | L | O |
--------------------------

File d'attente vide...
--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | L | O | L | O |
--------------------------

Client 9 traité par machine 0
Client 7 par machine 3 terminé. Le client numéro 7 a été traité.

--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | O | O | L | L |
--------------------------

File d'attente vide...
--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | O | O | L | L |
--------------------------

Client 8 par machine 1 terminé. Le client numéro 8 a été traité.

--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | O | L | L | L |
--------------------------

File d'attente vide...
--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | O | L | L | L |
--------------------------

Client 9 par machine 0 terminé. Le client numéro 9 a été traité.

--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
```

```
   État   | L | L | L | L |
--------------------------

File d'attente vide...
--------------------------
 Automate | 0 | 1 | 2 | 3 |
--------------------------
   État   | L | L | L | L |
--------------------------
```