INTRODUCTION À LA SIMULATION

Pascal CANTOT

Ingénieur Principal des Études et Techniques d'Armement

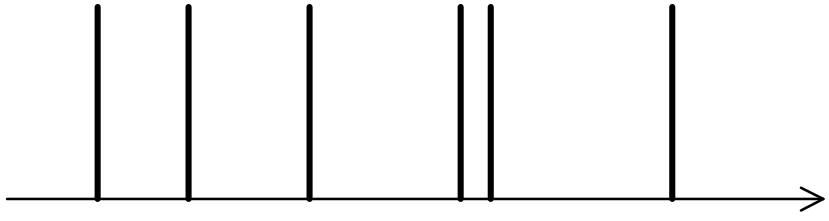
cantotp@wanadoo.fr

LA SIMULATION À ÉVÉNEMENTS DISCRETS

RAPPELS

Un **système à événements discrets** est un système dont l'état change de façon discontinue en un certain nombre d'instants pouvant être ou apparaître aléatoires.

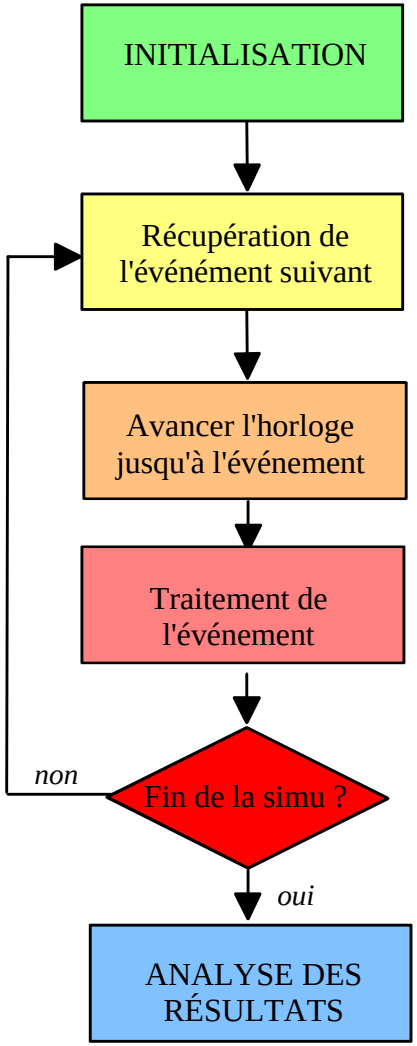
➔ La simulation par événements discrets (*discrete event simulation*, habituellement abrégée DES ou DEVS) traite de la modélisation de tels systèmes.



RAPPELS

- Dans une simulation à événements discrets, on rencontre fréquemment les éléments suivants :
  - **Entités**
  - **Ressources** : éléments du système fournissant un service
  - **Éléments de contrôles** : permettent de modifier la réponse du système à un événement, par l'intermédiaire de switches, compteurs, règles logiques ou arithmétiques
    Ex.: contrainte horaire (fermeture des guichets pendant la pause de midi)
  - **Opérations** : ce sont les manipulations effectuées par ou sur les entités évoluant au sein du système.
    Ex.: le traitement d'un client au guichet ou le départ de ce client

LA RÉPLIQUE



- Une **réplique** (*replication* ou *run* en anglais) est une exécution d'une simulation.

- Gestion des événements ?

GESTIONS DES ÉVÉNEMENTS

- En DEVS, les événements conditionnent le temps de la simulation
- Les événements sont déterminés dans le présent de la simulation pour une date future
- Utilisation d'une liste d'événements

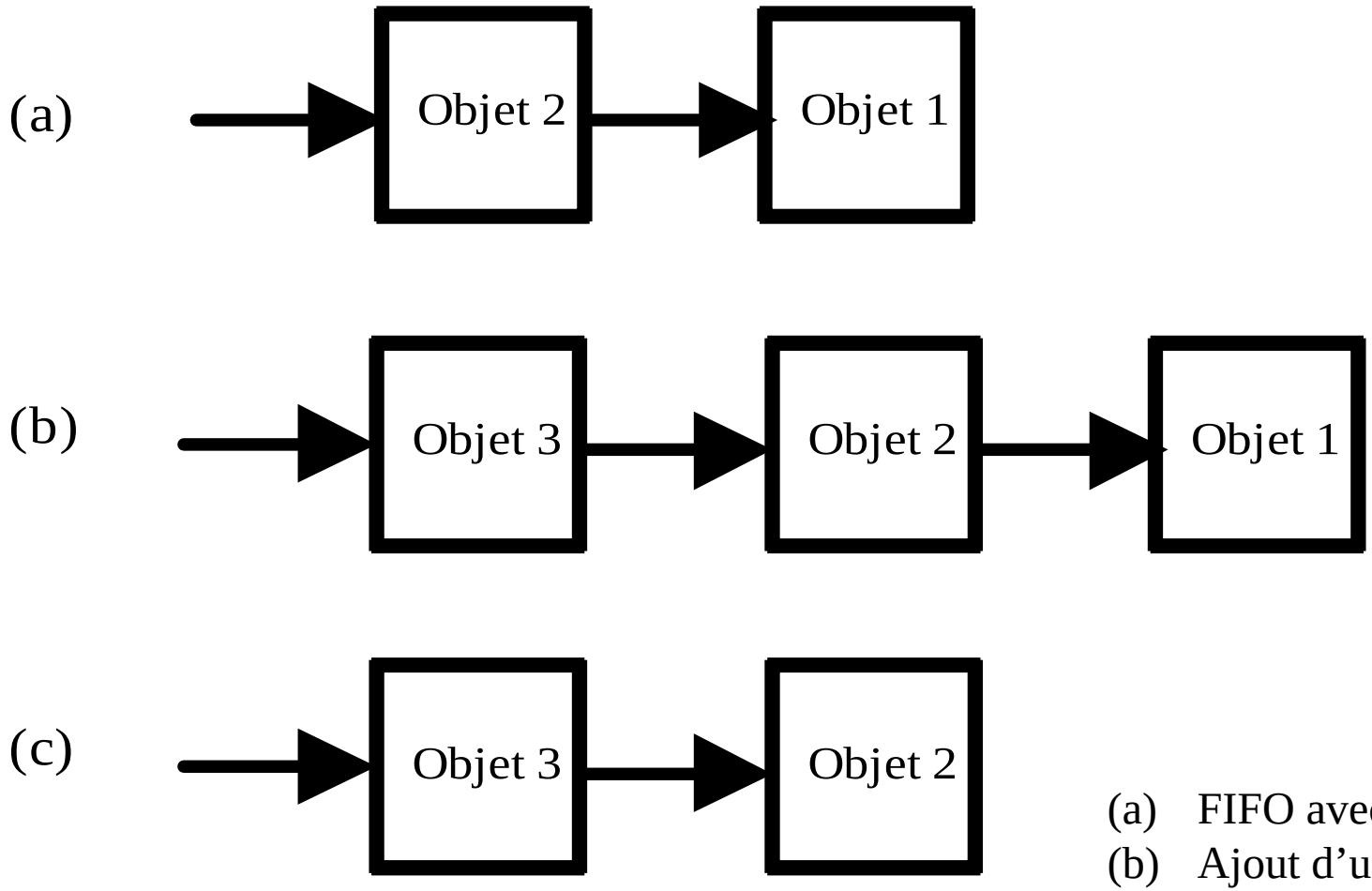
EXEMPLE DE LISTE D'ÉVÉNEMENTS

- **0.000 : ouverture guichets**
- **0.000 : arrivée client**
- **0.000 : arrivée client**
- **0.320 : arrivée client**
- **0.563 : arrivée client**
- **...**
- **3.500 : début pause déjeuner**
- **4.250 : fin pause déjeuner**
- **...**
- **6.155 : début panne ordinateur**
- **6.670 : fin panne ordinateur**
- **...**

FILE D'ATTENTE

- **Queue** ou **FIFO** : First In, First Out
- Ordonnée **chronologiquement** suivant la date (*timestamp*) des événements postés dans la file
- TSO = *Time Stamp Order*

FONCTIONNEMENT D'UNE FIFO



(a) FIFO avec 2 items
(b) Ajout d'un item
(c) Retrait d'un item

EXEMPLE D'IMPLÉMENTATION

- B.E. ENSIETA 1999-2000 : gestion de stocks
  - Méthode par calcul itératif (EXCEL)
  - Méthode par simulation à événements discrets
- Il existe de nombreux outils "sur étagère"
  Ex.: DEVS, Matlab/Simulink, GPSS/H, Arena/Siman...
- Utilisation d'une librairie "maison"

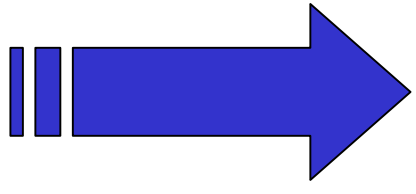
MODULES

- **`magasin.java`**
  Programme principal, avec paramètres de la simulation, moteur simulation, statistiques…
- **`MoreRandom.java`**
  Classe pour génération de nombres aléatoires
- **`Event.java`**
  Classe structure pour décrire/stocker des événements
- **`EventQueue.java`**
  Classe de file d'événements datés ou non

Note : utilisation de Javadoc

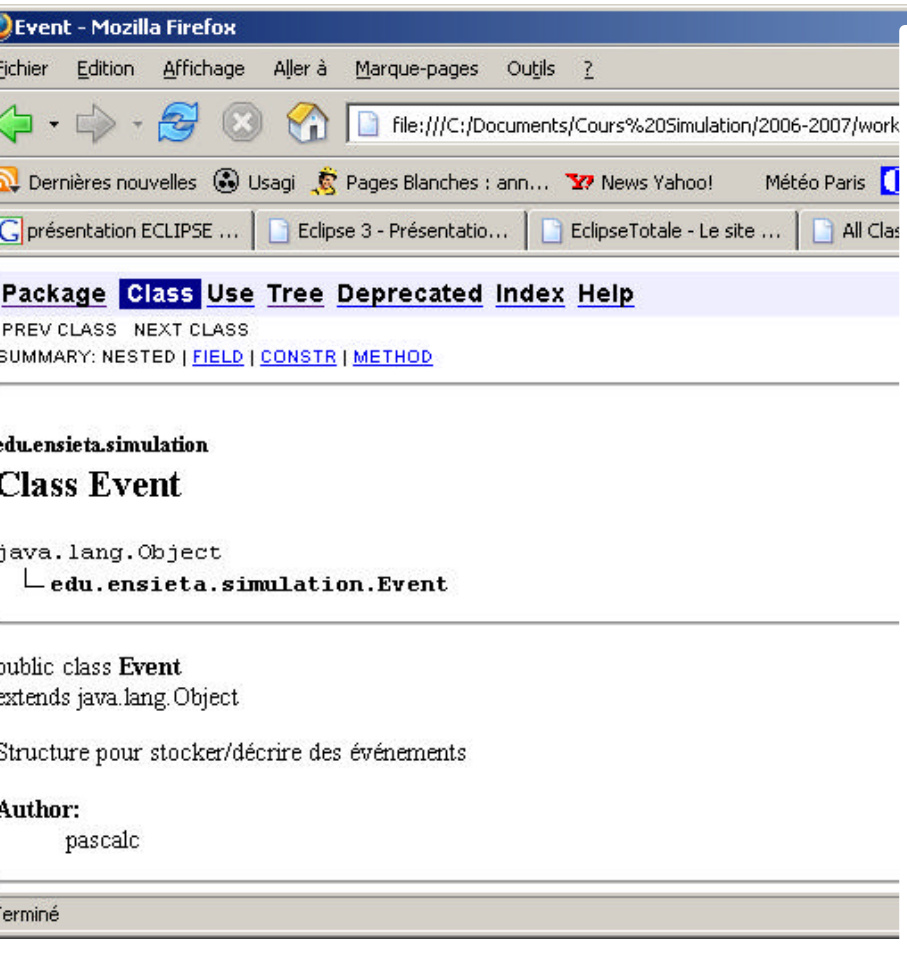
- Le code Java peut être autodocumenté moyennant le respect de certaines conventions
- Exemple extrait de la classe Event :

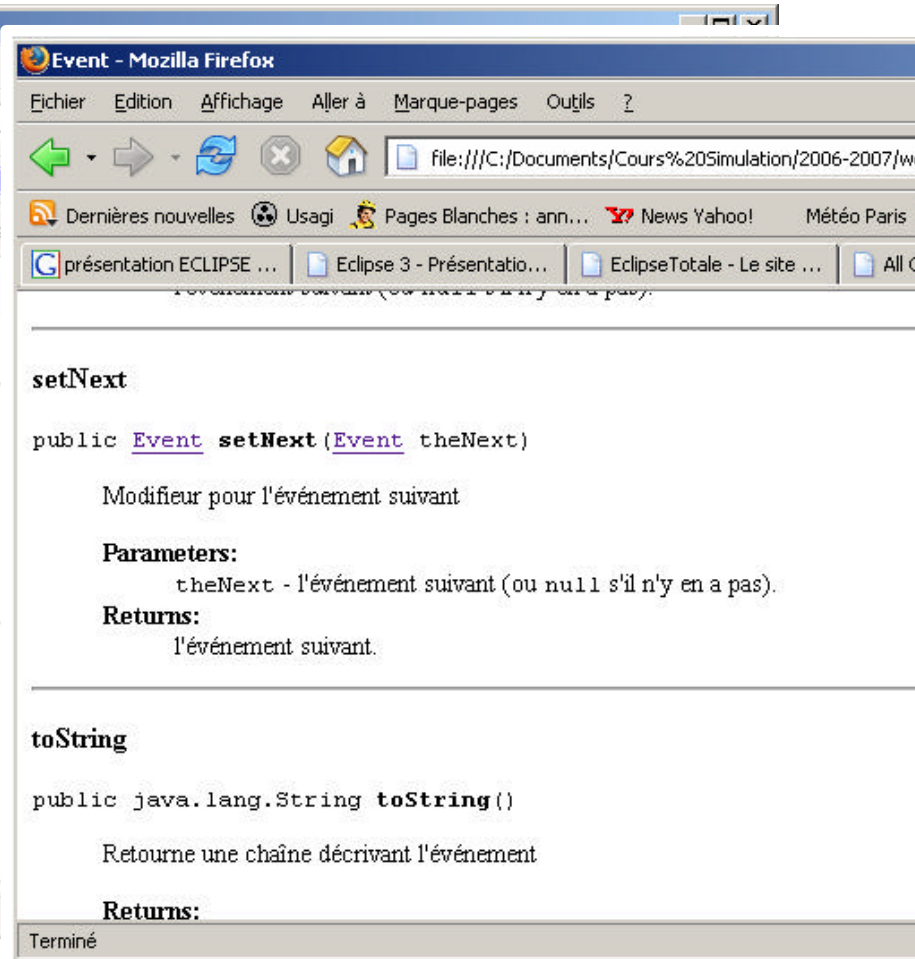
```
/**
 * Modifieur pour l'événement suivant
 * @param theNextl'événement suivant
 *     (ou <code>null</code> s'il n'y en a pas).
 * @return l'événement suivant.
 */
public Event setNext(Event theNext)
{
   . . .
}
```



Note : utilisation de Javadoc

➔ Générateur de documention `javadoc` :





Class Event

```
protected int id;
protected double timeStamp;
protected int type;
protected String description;
public Object parameter;

public Event(double theTime, int theType)
public Event(double theTime, int theType, String theDescription, Object
            theParameter)
public double getId()
public double getTimeStamp()
public int getType()
public String getDescription()

public Event getNext()
public Event setNext(Event theNext)
public String toString()
private static void doTest()
```

Class EventQueue

```
public EventQueue(boolean isTimeStampOrdered)
public EventQueue()
public int getLength()
public void postEvent(Event evt)
public Event getNextEvent()
public Event readNextEvent()
void flush()
private static void doTest()
```

Class MoreRandom (extends Random)

```
public MoreRandom()
public MoreRandom(long seed)
public void setSeed(long seed)
public long getSeed()
public long randomize()
public double nextUniform()
public double nextUniform(double a, double b)
public double nextTriangle(double a, double b, double c)
public double nextExp(double lambda)
private static void doTest()
```

Magasin.java
Entête

```
import edu.ensieta.simulation.*;

/**
 * Simule le fonctionnement d'un magasin (très simple!)
 * @author cantotp@wanadoo.fr
 */

public class magasin
{
   // Déclaration des valeurs des événements
   final static int RIEN = 0;
   final static int CLIENT= 1;
   final static int LIVRAISON= 2;
   final static int FIN= 99;
```

Magasin.java
Paramètres de la simulation

```
// *** PARAMETRES DE LA SIMULATION ***
//
// Paramètre loi arrivée clients
final static double lambda = 1/2.0;

// Paramètres loi livraisons
final static double a=5.0;
final static double b=7.0;
final static double c=9.0;

// Paramètres de la simulation
final static double tmax= 360.0; // Unité de temps = jour
final static int stockInitial = 15;
final static int seuilCommande = 3;
final static int qteLivraison = 12;
```

Magasin.java
Variables de la simulation

```
private static void doSimulation()
{
    // *** VARIABLES DE LA SIMULATION ***

    // Variables statistiques
    int nbreClients = 0;
    int nbreInsatisfaits= 0;
    int nbreCommandes= 0;

    // Variables d'état et entités
    EventQueue fileEvt = new EventQueue(true);
    int stock = stockInitial;
    double tempsSimu = 0.0;

    // Générateur aléatoire
    MoreRandom alea = new MoreRandom();

    // Afficher les paramètres
    // ...
```

Magasin.java
Initialisation de la simulation

```
// *** INITIALISATION DE LA SIMULATION ***

// Pour démo
fileEvt.postEvent(new Event(0.0, RIEN, "Pour démo", null));

// Poster l'événement FIN à tmax
fileEvt.postEvent(new Event(tmax, FIN, "Fin de la simulation après " + tmax
    + " jours", null));

// Poster les événements client jusqu'à tmax
double t = alea.nextExp(lambda);

while (t < tmax)
{
    nbreClients++;
    fileEvt.postEvent(new Event(t, CLIENT, "Arrivée client #" + nbreClients,
          nbreClients));
    t += alea.nextExp(lambda);
}
```

Magasin.java
Boucle de simulation

```
// *** BOUCLE DE SIMULATION ***
// Lit les évenements dans le scheduler et les traite

Event evt; // Variable de travail pour l'événement courant
boolean finalizeSimulation = false; // Flag pour sortir de la boucle
do
{
    // Récupère l'évenement suivant
    evt = fileEvt.getNextEvent();
    // Avance le temps de la simulation à la date de l'événement
    tempsSimu = evt.getTimeStamp();
    // Affiche une trace sur la console
    System.out.print(evt.toString());

    // Traite l'événement
    switch(evt.getType())
    {
          . . .
    }
} while (!finalizeSimulation);
```

Magasin.java
Traitement d'un événement

```
case CLIENT:
    if (stock > 0)
    {
          // Si stock, alors client satisfait et une vente réalisée
          stock--;
          System.out.print(" ==> Servi");
    }
    else
    {
          // Sinon, client insatisfait!
          nbreInsatisfaits++;
          System.out.print(" ==> INSATISFAIT!");
    }
    // Vérifie si besoin commande
    if (stock == seuilCommande)
    {
          nbreCommandes++;
          System.out.print(" | COMMANDE");
          // Calcule la date de livraison
          fileEvt.postEvent(new Event(tempsSimu + alea.nextTriangle(a, b, c),
                    LIVRAISON, "Livraison de " + qteLivraison + " unités", null));
    }
    System.out.println();
    break;
```

Magasin.java
Sortie des résultats

```
    // *** RESULTATS DE LA SIMULATION ***

    // Affiche les variables statistiques
    System.out.println("Nombre de clients     = " + nbreClients);
    System.out.print  ("Nombre d'insatisfaits = " + nbreInsatisfaits);
    System.out.println(" (" + String.format("%2.2f",
         100.0*nbreInsatisfaits/nbreClients) + "%)");
    System.out.println("Nombre de commandes   = " + nbreCommandes);
}
```

MAGASIN.EXE
Exécution

```
0,000000: 00 - Pour démo [id=0] ==> Rien
0,592477: 01 - Arrivée client #1 [id=2] ==> Servi
0,932704: 01 - Arrivée client #2 [id=3] ==> Servi
2,161226: 01 - Arrivée client #3 [id=4] ==> Servi
. . .
325,898545: 01 - Arrivée client #163 [id=164] ==> Servi
329,535388: 01 - Arrivée client #164 [id=165] ==> INSATISFAIT!
331,026962: 02 - Livraison de 12 unités [id=189] ==> Stock = 12
333,481225: 01 - Arrivée client #165 [id=166] ==> Servi
336,267650: 01 - Arrivée client #166 [id=167] ==> Servi
342,552094: 01 - Arrivée client #167 [id=168] ==> Servi
344,453833: 01 - Arrivée client #168 [id=169] ==> Servi
346,355175: 01 - Arrivée client #169 [id=170] ==> Servi
348,404009: 01 - Arrivée client #170 [id=171] ==> Servi
348,806816: 01 - Arrivée client #171 [id=172] ==> Servi
348,916989: 01 - Arrivée client #172 [id=173] ==> Servi
349,142341: 01 - Arrivée client #173 [id=174] ==> Servi | COMMANDE
351,492392: 01 - Arrivée client #174 [id=175] ==> Servi
355,370972: 01 - Arrivée client #175 [id=176] ==> Servi
356,225528: 02 - Livraison de 12 unités [id=190] ==> Stock = 13
357,106014: 01 - Arrivée client #176 [id=177] ==> Servi
360,000000: 99 - Fin de la simulation après 360.0 jours [id=1] ==> Finalize
Nombre de clients     = 176
Nombre d'insatisfaits = 17 (9,66%)
Nombre de commandes   = 13
```

On a pour chaque événement:

- date
- paramètre(s)
- compte-rendu de l'action

(peut être génération d'un autre événement)

Note : utilisation d'ECLIPSE

- IDE : Environnement de Développement Intégré
- Framework pour la réalisation d'IDE
  - Noyau : « runtime »
  - Plug-ins ➔ Java, C++, C#, PHP, Cobol, UML, Tomcat …
- Multi plateformes Win+Linux (Java SWT)
- Origine IBM (base de Websphere Studio)
  mais logiciel libre
- Aujourd'hui leader en plate-forme de développement

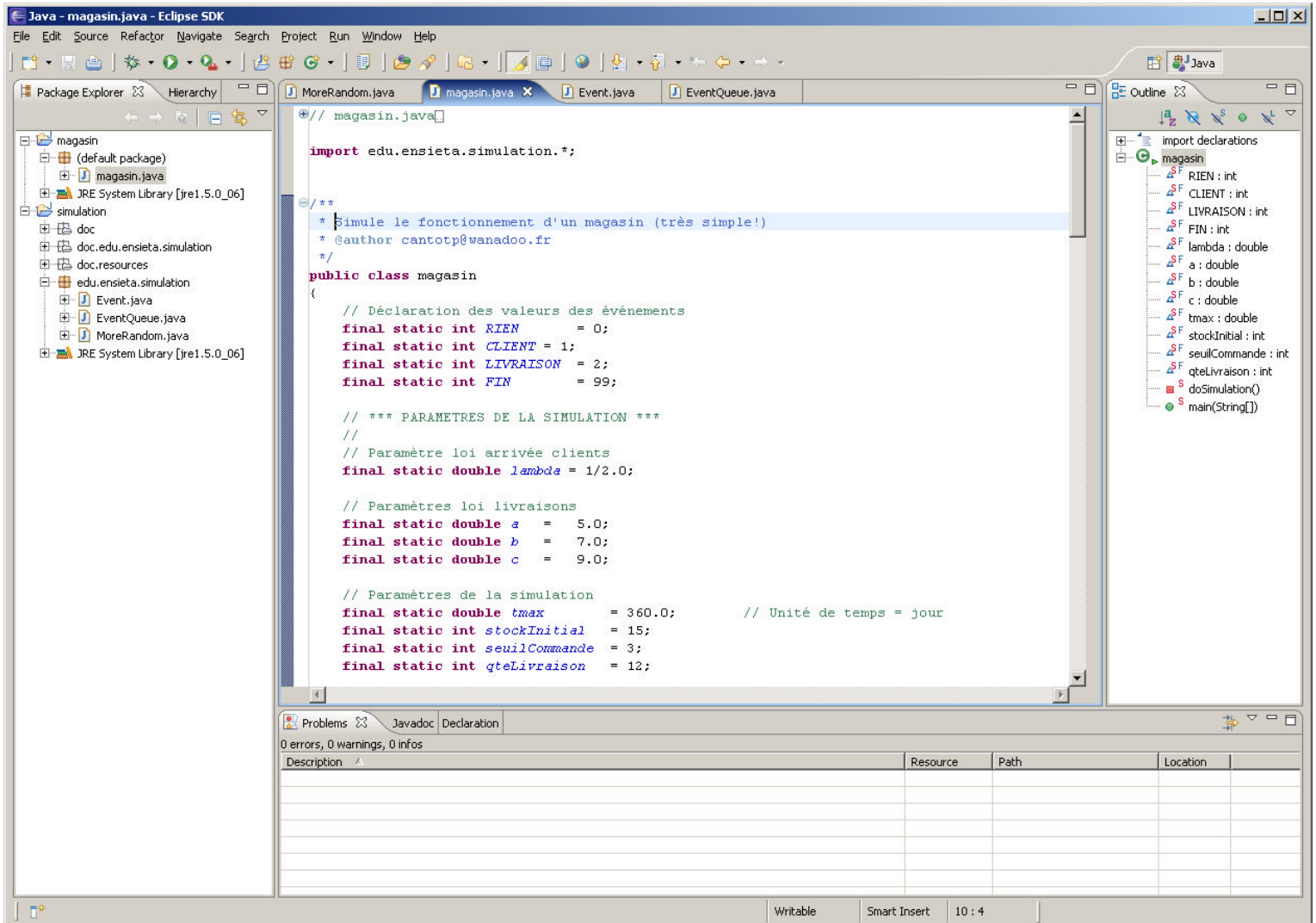
Quelques liens de perfectionnement

- Site officiel d'ECLIPSE
  http://www.eclipse.org/

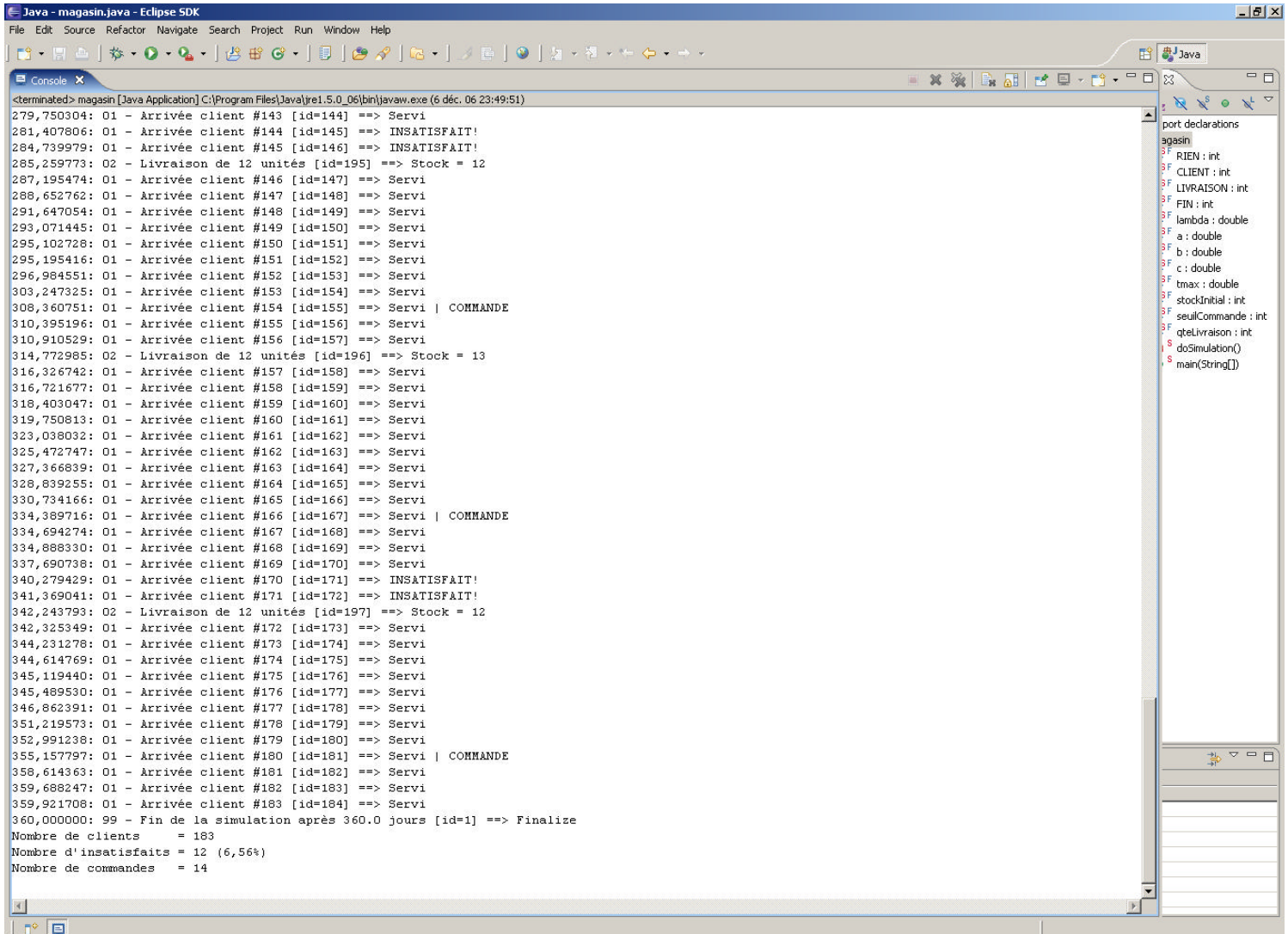
- Site francophone consacré au projet Eclipse et aux outils WebSphere Studio et Rational d'IBM
  http://www.eclipsetotale.com/

- Développons en Java avec Eclipse
  http://www.jmdoudoux.fr/java/dejae/titre.htm

Interface ECLIPSE



Console d'exécution



Et maintenant…

À VOUS DE JOUER !

Installation des outils DEVS

- Aller sur le site **simucentre.free.fr**
  http://simucentre.free.fr/cours.php
  Récupérer le sujet et les bibliothèques
  Les stocker dans un répertoire dédié (ex.: **`$HOME/tdsimu`**)

- Créer un projet sous ECLIPSE si vous l'utilisez
  (mais on peut le faire en ligne de commande : `javac`…)

- Vous pouvez aussi utiliser C++