Threads

Programmation objets
et Java

(B. Sonntag)

Définition

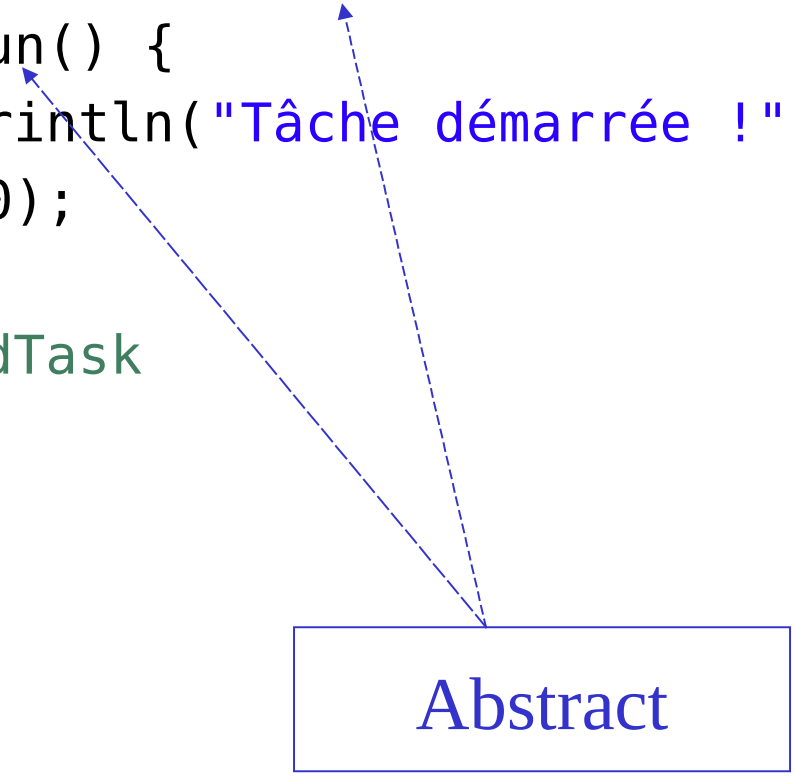
- *Processus léger*
- Execution d'une routine d'un programme plus gros, de manière indépendante
- Exemple, pour un **Navigateur** :
  - Téléchargement
  - Défilement
  - Animation
  - Son
  - Impression…

Exemple 1 (Timer)

```
<planifier une tâche pour un démarrage différé>
Démarrage de tâche dans 5 sec...
<plus tard…>
Tâche démarrée !
```

Exemple 1 (Timer)

```
class Tache extends TimerTask {
  public void run() {
   System.out.println("Tâche démarrée !");
   System.exit(0);
  }
}//classe RemindTask
```



Exemple 1 (Timer)

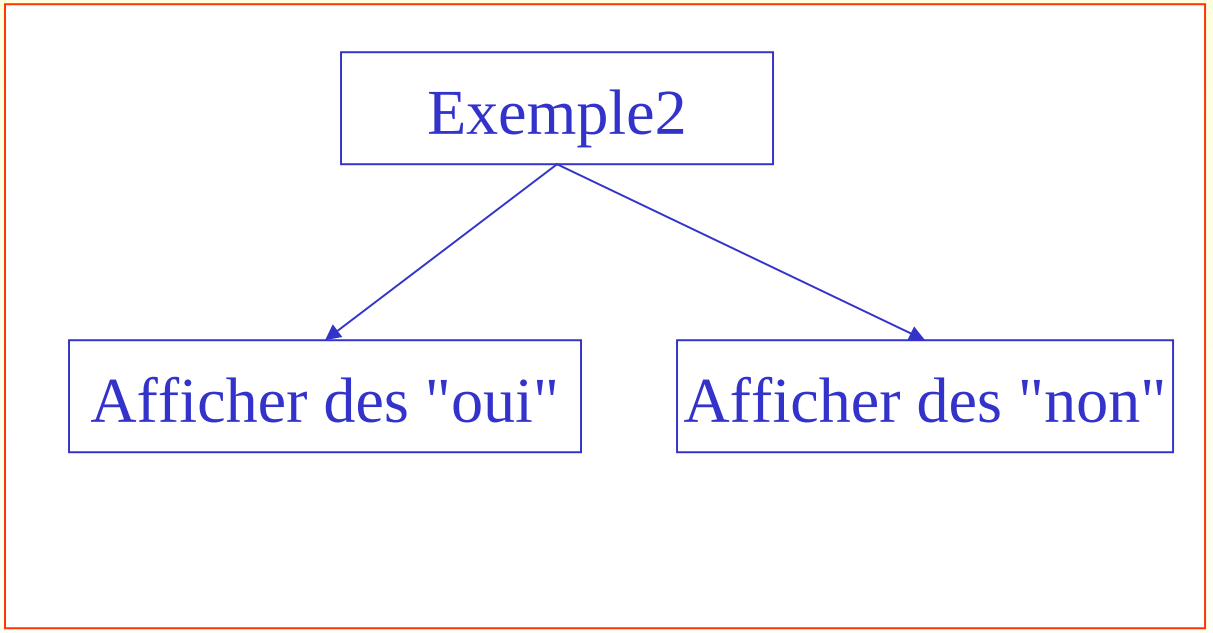
```
public class Exemple1 {
/**
 * Démarre une tâche en différé (de 5 sec.)
 */
 Timer timer;

 public Exemple1(int sec) {
  timer = new Timer();
  timer.schedule(new Tache(), sec*1000);
 }
//main …
}//classe Exemple1
```

Exemple 1 (Timer)

```
public static void main(String[] args) {
  new Exemple1(5);
  System.out.println("Démarrage de tâche dans 5s...");
}
```

Exemple2 (Parallèle)



Exemple 2 (Parallèle)

```
public class Oui implements Runnable{
/**
 * Affiche oui 10x
 */
 public void run() {
  System.out.println("Début OUI");
  for (int i = 0; i < 10; i++) {
   System.out.println("OUI: "+i);;
   try {
    Thread.sleep(1000);
   } catch (InterruptedException e) {//ignorer}
  }
 }
}
```

Exemple 2 (Parallèle)

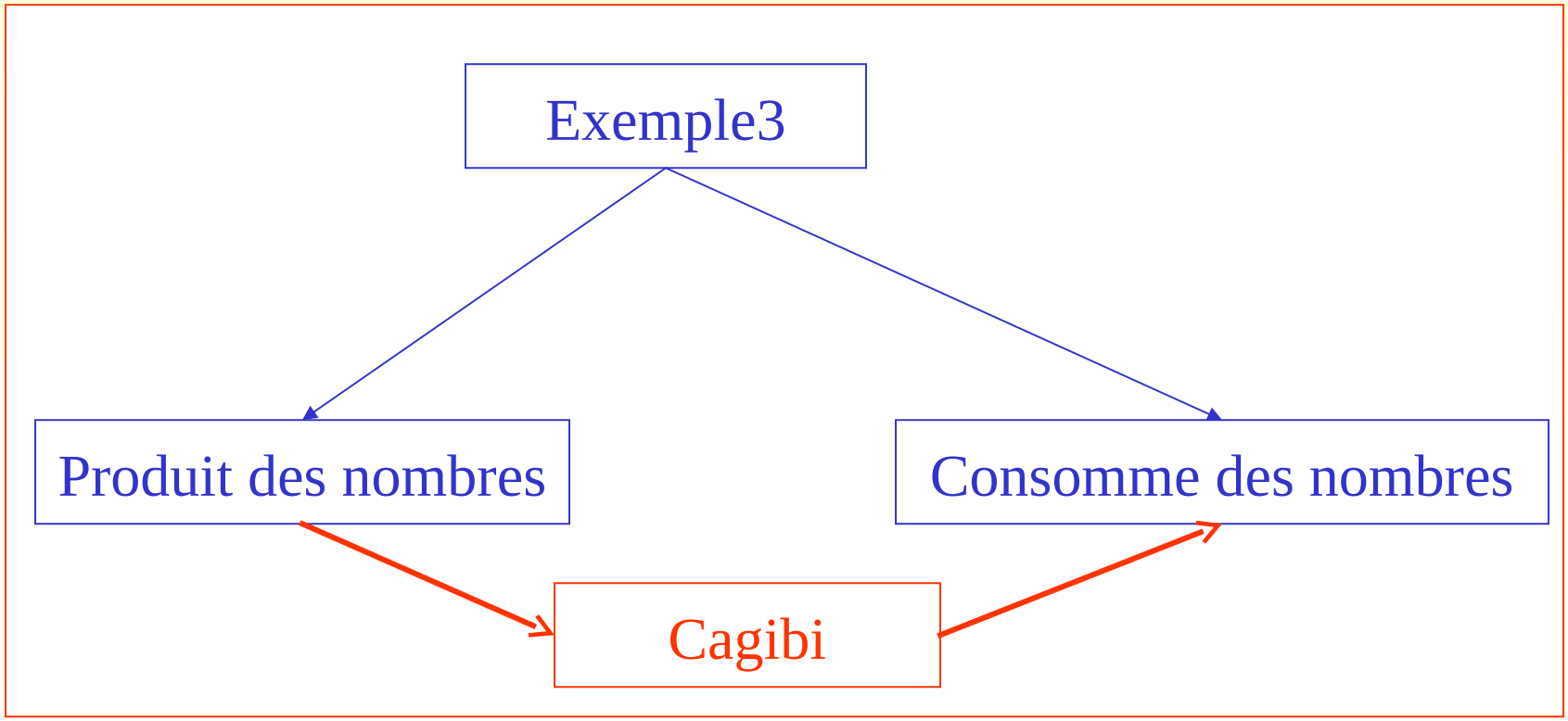
```
public class Non implements Runnable{
 /**
  * Affiche non 5x
  */
 public void run() {
  System.out.println("Début NON");
  for (int i = 0; i < 5; i++) {
   System.out.println("NON: "+i);
   try {
   Thread.sleep(2000);
   } catch (InterruptedException e) {//ignorer}
  }
 }
}
```

Exemple 2 (Parallèle)

```
public class Exemple2 {

 public static void main(String[] args) {
  Oui oui = new Oui();
  Non non = new Non();
  Thread th1 = new Thread(oui);
  Thread th2 = new Thread(non);
  th1.start();
  th2.start();
 }
}
```

Exemple3 (P/C)



Exemple3 (P/C)

```
public class Exemple3 {
/**
 * Teste producteur/consommateur
 */
 public static void main(String[] args) {
  Cagibi ca = new Cagibi();
  Producteur pr = new Producteur(ca);
  Consommateur co = new Consommateur(ca);
  Thread th1 = new Thread(pr);
  Thread th2 = new Thread(co);
  th1.start();
  th2.start();
 }
}
```

Exemple3 (P/C)

```
public class Producteur implements Runnable{
 private Cagibi cagibi;
 public Producteur(Cagibi c) {
  cagibi = c;
 }
 public void run() {
  for (int i = 0; i < 10; i++) {
   cagibi.put(i);
   System.out.println("Produit: " + i);
   try {
    Thread.sleep((int)(Math.random() * 1000));
   } catch (InterruptedException e) { }
  }
 }
}
```

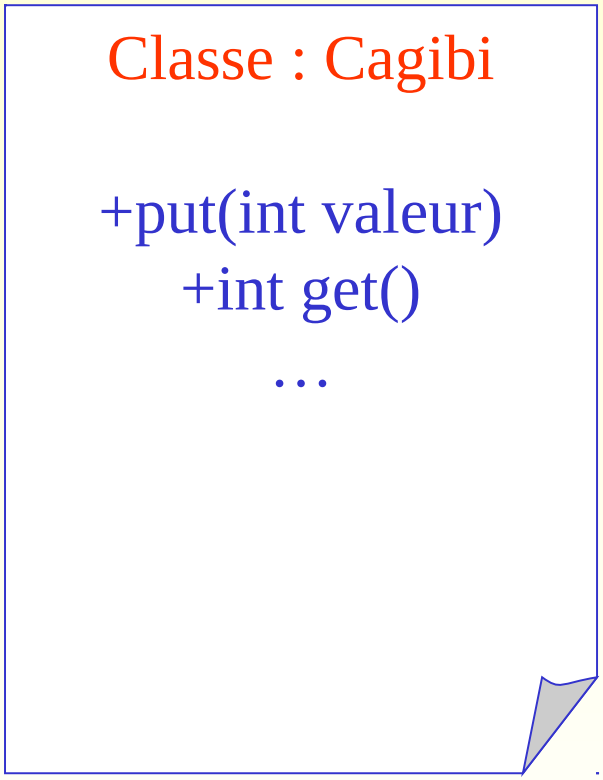
Exemple3 (P/C)

```
public class Consommateur implements Runnable{
 private Cagibi cagibi;

 public Consommateur(Cagibi c) {
  cagibi = c;
 }

 public void run() {
  int value = 0;
  for (int i = 0; i < 10; i++) {
   int value = cagibi.get();
   System.out.println("Consomme: " + value);
  }
 }
}
```

Exemple3 (P/C)



Verrou

- L'objet Cagibi partagé par le producteur et consommateur risque d'être utilisé simultanément par les 2 threads
- Java offre la possibilité de vérrouiller certaines méthodes d'une instance
- Il suffit de définir les méthodes sensibles comme *synchronized*

Exemple3 (P/C)

```
public class Cagibi {

 private int val;
 private boolean dispo = false;

 public synchronized int get() {
  …
 }

 public synchronized void put(int x) {
   …
 }
}
```

Exemple3 (P/C)

```
public synchronized int get() {
 while (dispo == false) {
  try {
   wait();
  } catch (InterruptedException e) {}
 }
 dispo = false;
 notifyAll();
 return val;
}
```

réveille tous les threads qui sont en attente sur le moniteur d'objet

Exemple3 (P/C)

```
public synchronized void put(int x) {
 while (dispo == true) {
  try {
   wait();
  } catch (InterruptedException e) {}
 }
 val = x;
 dispo = true;
 notifyAll();
}
```

Exemple3 (P/C)

```
Produit: 0
Consomme: 0
Consomme: 1
Produit: 1
Consomme: 2
Produit: 2
Produit: 3
Consomme: 3
Consomme: 4
Produit: 4
Consomme: 5
Produit: 5
Produit: 6
Consomme: 6
Consomme: 7
Produit: 7
Consomme: 8
Produit: 8
Produit: 9
Consomme: 9
```