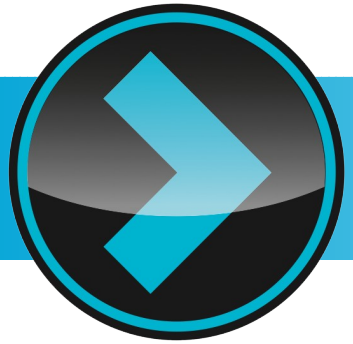
Threads

Java B3

ingésup
L' École d'Experts en Informatique
> Bordeaux/Toulouse

Course Goals

- Thread creation
- Thread use



Introduction

First example

- In this example, we have two attributes :
  - a memorized number, and a proposed number
- We can change proposed number.

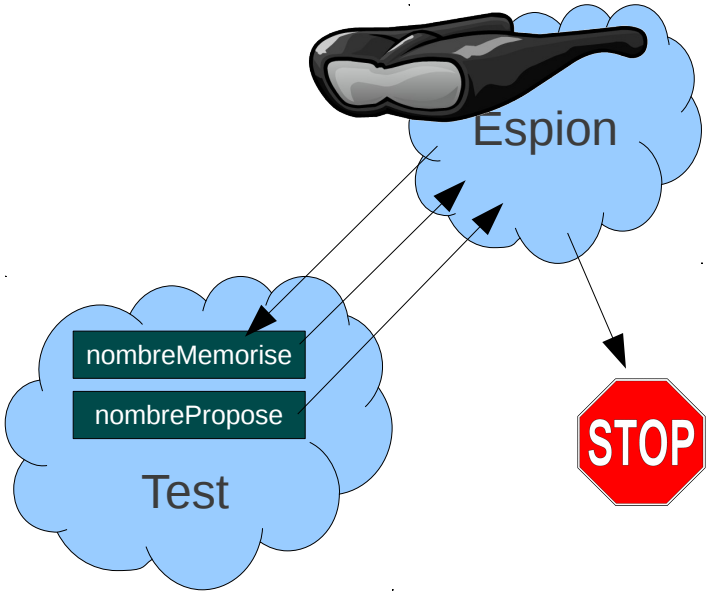
```
package com.ingesup.b3.threads;

import java.util.Scanner;

public class Test
{
	private int nombreMemorise,nombrePropose;
	public Test(){
		nombreMemorise=2;
		nombrePropose=2;
		Scanner in=new Scanner(System.in);
		while(true){
			afficheMemorise();
			System.out.print("Entrez une nouvelle valeur : ");
			nombrePropose=in.nextInt();
		}
	}
	public void afficheMemorise(){
		System.out.println("Le nombre actuellement mémorisé est "+nombreMemorise);
	}
	public static void main(String[] args){
		new Test();
	}
	public int getNombreMemorise()                          {          return nombreMemorise;  }
	public void setNombreMemorise(int nombreMemorise)       {          this.nombreMemorise = nombreMemorise;     }
	public int getNombrePropose()                           {          return nombrePropose;   }
}
```

First example

- We want to create a spy which compares memorized number with proposed number.
- This spy will run in parallel and store proposed number into memorized number, with a message. If stored number is 0, spy will force program exiting.
- First of all, we create a thread class inheriting Runnable
- Because we want thread to discuss with class, thread's constructor will memorize parent class
- run() method :
  - will have an infinite loop
  - will sleep 1 second (→ use Thread.sleep() with a try.. catch..)
  - will test if two numbers are different
    - if yes, stores proposed number into memorized number, and prints it. If number equals 0, exit



Thread class - result

```
package com.ingesup.b3.threads;

public class ThreadEspion implements Runnable
{
	private Test test;
	public ThreadEspion(Test test){
		this.test=test;
	}
	public void run(){
		while(true)	{
			try	{
			Thread.sleep(1000);
			} catch (InterruptedException e)  {}
			if(test.getNombreMemorise()!=test.getNombrePropose()){
				System.out.println("Le thread s'active ...");
				test.setNombreMemorise(test.getNombrePropose());
				test.afficheMemorise();
				if(test.getNombreMemorise()==0)
					break;
				System.out.println("Le thread se rendort....");
			}
		}
		System.exit(0);
	}
}
```

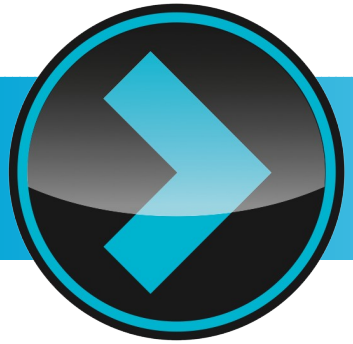
- First of all, we create a thread class inheriting Runnable
- Because we want thread to discuss with class, thread's constructor will memorize parent class
- run() method :
  - will have an infinite loop
  - will sleep 1 second (→ use Thread.sleep() with a try.. catch..)
  - will test if two numbers are different
    - if yes, stores proposed number into memorized number, and prints it. If number equals 0, exit

Thread use

- In main class, we will instanciate thread and run it. That's all !

```
public class Test
{
	private int nombreMemorise,nombrePropose;
	public Test()
	{
		nombreMemorise=2;
		nombrePropose=2;
		Scanner in=new Scanner(System.in);
		Thread espion=new Thread(new ThreadEspion(this));
		espion.start();
		while(true)
		{
			afficheMemorise();
			System.out.print("Entrez une nouvelle valeur : ");
			nombrePropose=in.nextInt();
		}
	}
	...
```

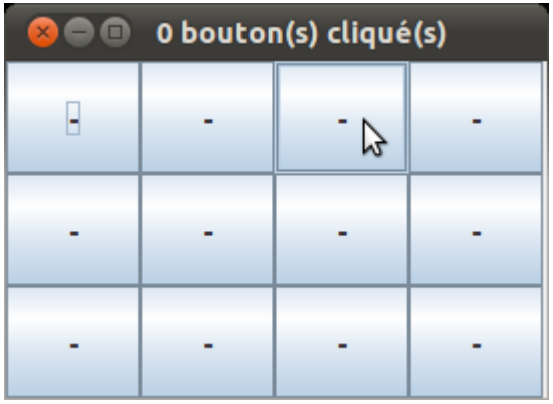
```
Le nombre actuellement mémorisé est 2
Entrez une nouvelle valeur : 2
Le nombre actuellement mémorisé est 2
Entrez une nouvelle valeur : 3
Le nombre actuellement mémorisé est 2
Entrez une nouvelle valeur : Le thread s'active : nombres différents...
Le nombre actuellement mémorisé est 3
Le thread se rendort....
2
Le nombre actuellement mémorisé est 3
Entrez une nouvelle valeur : Le thread s'active : nombres différents...
Le nombre actuellement mémorisé est 2
Le thread se rendort....
0
Le nombre actuellement mémorisé est 2
Entrez une nouvelle valeur : Le thread s'active : nombres différents...
Le nombre actuellement mémorisé est 0
```

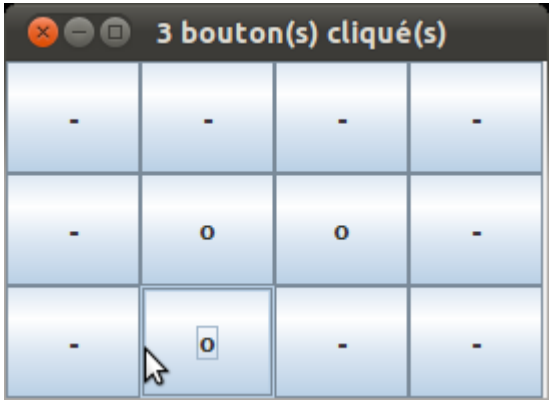


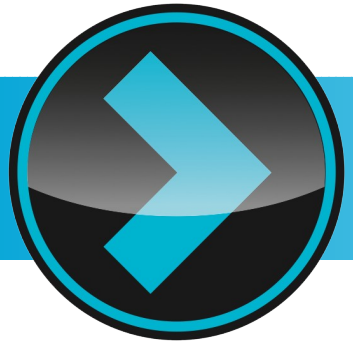
First job

Work

- Create a dialog box with an array of buttons. Their label is initialized to “-”.
- If we click on one of them, their label is toggled to “o” (program inverse toggle).
- Create a spy thread which counts buttons with “o”, modify frame title, and exit program if count is 5



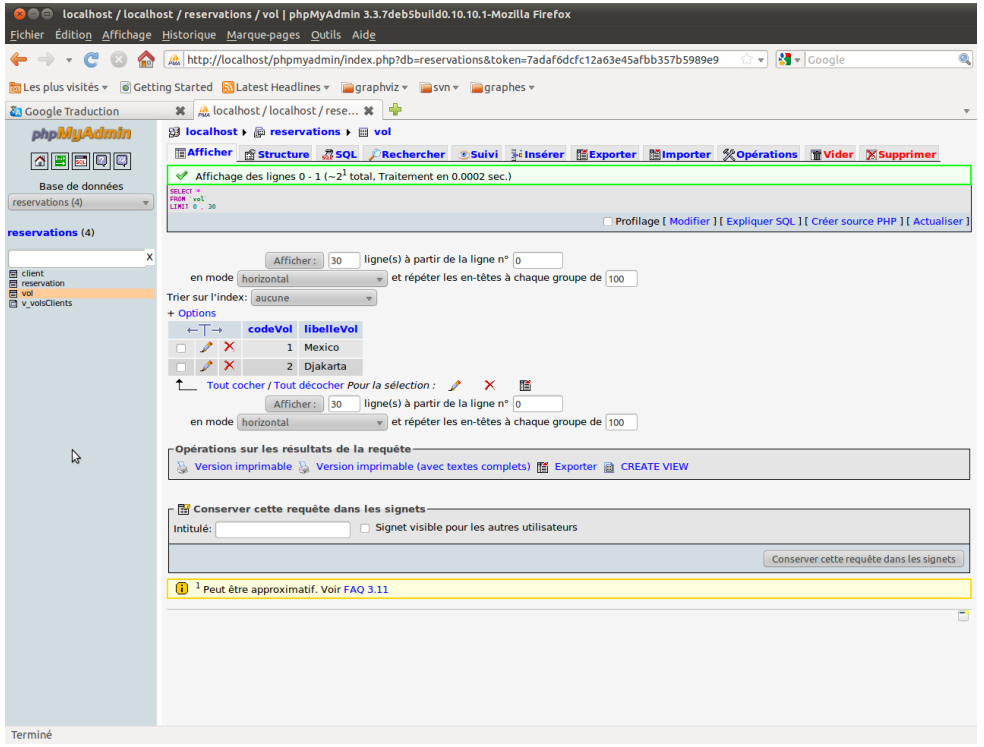




Hard job

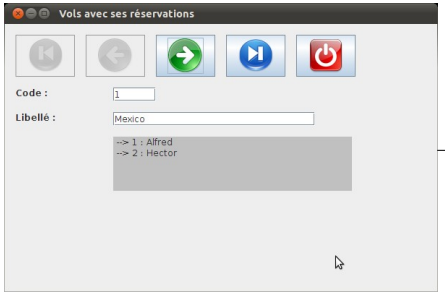
Work

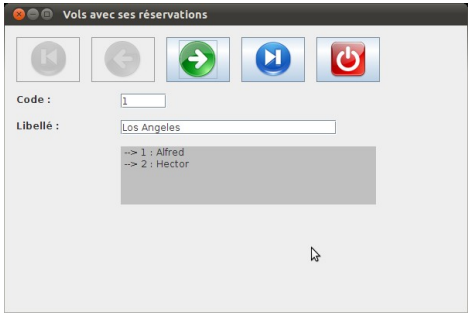
- Import 08_VolsAvantThreads.zip and reservations.sql database
- We have a dialog box which shows fligth names with its clients.
- Data are retrieved through MySql

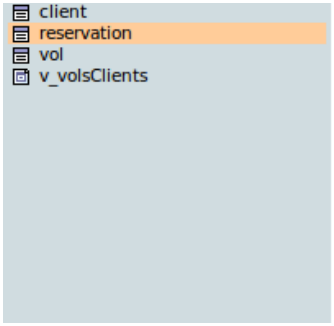


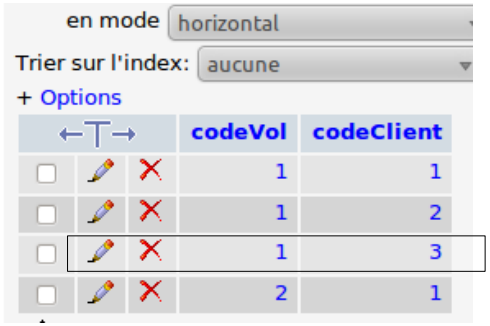
Work

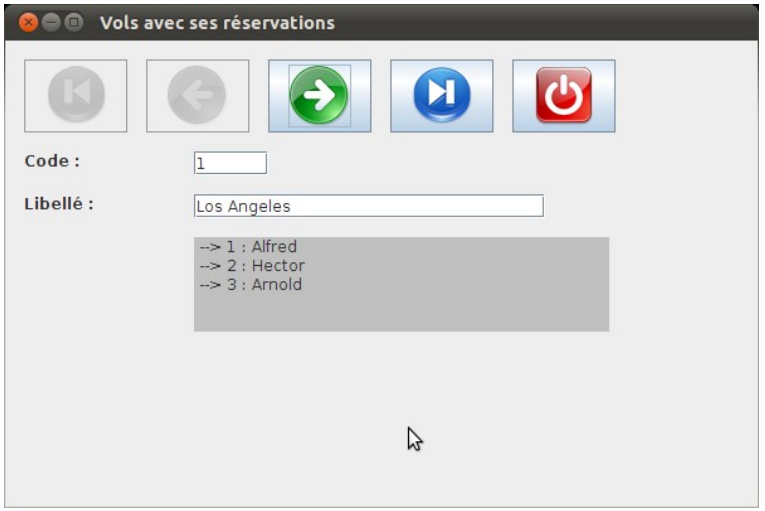
- We want to create a spy which has two jobs :
  - compare and change current flight label with database
  - compare current flight clients count and change them











Work

- Thread must :
  - retrieve actual dialog box flight code and label, and compare it to database
  - retrieve actual dialog box flight clients count and change them
- Modify VolsInterface accessors, add a refresh method to both arraylist classes
- Thread algorithm :
  - Constructor : memorize panel
  - run method :
    - infinite loop
    - sleep 3 seconds
    - retrieve flight code and label
    - connect to database and compare flight labels
      - if different, we have to refresh vols ArrayList and re-show data
    - compare flight clients count
      - if different, we refresh...

UML Proposition