INFO-H-200 : Programmation orientée objet
TP 7 - Threads

Professeur : Hugues Bersini
http://cs.ulb.ac.be/public/teaching/infoh200

Année académique 2015-2016

Exercice 7.1

Implémentez la classe `MyTimer` présentée dans le rappel et testez là avec plusieurs valeurs.

Exercice 7.2

Implémentez la classe `MySync` présentée dans le rappel et testez là sans et avec l'ajout de la fonction 'join'.

Exercice 7.3

Modifier le code suivant pour que l'affichage se fasse de façon ordonnée.

```
public class Synchronization{
  public static void main(String args[]){
    Table obj = new Table();//only one object
    MyThread1 t1=new MyThread1(obj);
    MyThread2 t2=new MyThread2(obj);
    t1.start();
    t2.start();
  }
}
```

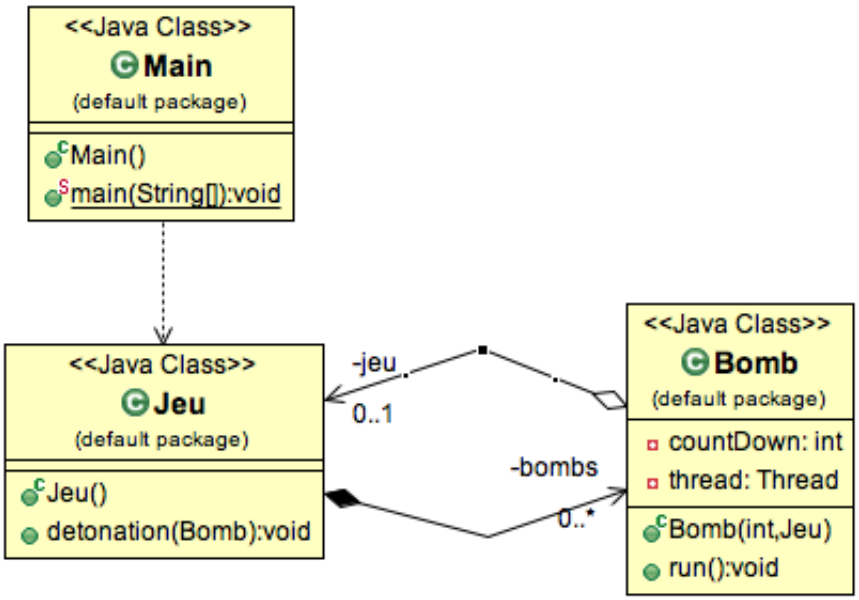
```
public class Table{
  void printTable(int n){
    for(int i=1;i<=5;i++){
      System.out.println(n*i);
      try{
        Thread.sleep(400);
      }catch(Exception e){System.out.println(e);}
    }
  }
}

public class MyThread1 extends Thread{
  Table t;

  MyThread1(Table t){
    this.t=t;
  }

  public void run(){
    t.printTable(5);
  }
}

public class MyThread2 extends Thread{
  Table t;

  MyThread2(Table t){
    this.t=t;
  }

  public void run(){
    t.printTable(100);
  }
}
```

Exercice 7.4

Considérez le scénario suivant : Dans un jeu de Bomberman, une fonctionnalité supplémentaire est implémentée. Des bombes vont être générées aléatoirement et doivent exploser après un temps donné. L'explosion d'une bombe provoque l'envoi d'un message de la bombe qui explose au jeu afin que celui-ci puisse déclencer les méthodes nécessaires.

Réduit au strict minimum, le problème fait intervenir deux classes : Jeu et Bombe. Le jeu génère quelque bombes avec des timers différents et celles-ci, lors de leur explosion, préviennent le jeu en appelant la fonction 'detonation()'.

On vous demande donc d'implémenter la structure de programme Java suivante en faisant usage de Thread pour gérer le compte à rebour des bombes :



INFO-H-200 - Programmation orientée objet
TP 7

Corrections

Exercice 7.1

```java
// Fichier MyTimer.java
public class MyTimer implements Runnable {

  private String string;
  private int waitTime;
  private Thread thread;

  public MyTimer(String string, int waitTime){
    this.string = string;
    this.waitTime = waitTime;
    this.thread = new Thread(this);
    thread.start();
  }

  public void run() {
    try{
      while(true){
        System.out.println(string);
        Thread.sleep(waitTime);
      }
    } catch (Exception e) {};
  }

  public static void main(String[] args) {
    MyTimer first = new MyTimer("first",2000);
    MyTimer second = new MyTimer("second",3000);
  }
}
```

Exercice 7.4

Fichier Main.java

```java
public class Main {

  public static void main(String[] args) {
    Jeu jeu = new Jeu();
  }
}
```

Fichier Jeu.java

```java
import java.util.ArrayList;

public class Jeu {

  private ArrayList<Bomb> bombs = new ArrayList<Bomb>();

  public Jeu(){
    this.bombs.add(new Bomb(2000, this));
    this.bombs.add(new Bomb(4000, this));
    this.bombs.add(new Bomb(1000, this));
  }

  public void detonation(Bomb bomb){
    System.out.println(bombs.size());
    System.out.println("Jeu : a bomb exploded");
    this.bombs.remove(bomb);
    System.out.println(bombs.size());
  }
}
```

Fichier Bomb.java

```java
public class Bomb implements Runnable{
  private int countDown = 2000;
  private Jeu jeu;
  private Thread thread;

  public Bomb(int countDown, Jeu jeu){
    this.countDown = countDown;
    this.jeu = jeu;
    this.thread = new Thread(this);
    this.thread.start();
  }

  public void run(){
    try {
      Thread.sleep(this.countDown);
      System.out.println("Detonation...");
      this.jeu.detonation(this);
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }
}
```