# TP 1

# Threads en Java : premiers pas

# Threads et exceptions

1. Dans un nouveau projet, créer une classe Activite qui implémente l’interface Runnable et qui a pour comportement (méthode run) d’afficher l’identifiant du *thread* sur lequel elle s’exécute.
2. Créer une classe Application ayant une méthode main qui lance 5 *threads* chacun exécutant une instance de Activité, et exécuter le programme.
3. Modifier la méthode run de Activite pour qu’elle fasse appel à une autre méthode m(), qui a son tour appelle une méthode n(), qui « simule » une erreur donnant lieu à une exception (division par zéro ou accès à un pointeur null) . Essayez le programme sans traiter l’exception dans la fonction run.
4. L’exception sera maintenant traitée par un bloc try/catch dans la méthode run en affichant l’identifiant du *thread* courant et la pile d’appel au moment de l’erreur. Essayez le programme à nouveau.
5. Créer une classe MonException qui dérive de java.lang.Exception. Modifier la méthode run de Activite pour qu’elle fasse appel à une autre méthode f() qui soulève une exception de type MonException aléatoirement avec une probabilité de 0.2 (voir java.lang.Math).   
   Comme vous le verrez, le compilateur Java vous obligera alors de déclarer que la fonction f peut soulever l’exception (clause throws) et de traiter l’exception dans run.

# Création de threads : BallWorld

Nous allons commencer à partir d'un programme qui ouvre une fenêtre dans laquelle 4 balles colorées doivent bouger en rebondissant contre les murs.

## Programme initial

[Télécharger](http://webetud.iut-blagnac.fr/mod/resource/view.php?id=5197) BallWorld.zip dans votre répertoire de travail et décompressez le fichier. Vous y trouverez trois classes :

* Ball.java  
  Une instance de cette classe est une balle, qui vit dans un BallWorld. La balle a la capacité de se dessiner (avec la méthode draw) dans un contexte graphique. En outre, à la création (dans son constructeur), la balle s'enregistre elle-même dans le monde où elle habite en appelant la méthode BallWorld.addBall.
* BallWorld.java   
  Une instance de cette classe est un monde, qui peut contenir plusieurs balles, stockés dans un ArrayList. Un monde est une sous-classe de JPanel, la classe Swing utilisée comme surface de dessin. Par conséquent, le monde peut se dessiner (méthode paintComponent) en demandant à toutes ses balles de se dessiner.
* BallsApp.java   
  Cette classe contient la méthode main, qui crée un monde et y insère 4 balles.

Compiler et exécuter le programme:

$ javac \*.java

$ java BallsApp

**Regardez bien le code des classes et assurez-vous que vous le comprenez.**

Vous pouvez créer un projet dans l’IDE de votre choix (e.g., Eclipse) pour la suite de cet exercice.

## Exercice : Faire bouger les balles

Comme vous constatez, les balles ne bougent pas encore. Votre première tâche est de les faire bouger indépendamment les unes des autres. Pour cela, vous allez devoir convertir la classe Ball en une sous-classe de Thread et définir sa méthode run comme une boucle infinie où le ballon met à jour sa position puis se met en sommeil pendant une période égale a period.

# Propagation des opérations sur la mémoire

*Lire attentivement*, *puis compiler et exécuter* le programme ThreadTest1.java joint.

1. Répéter l’exécution plusieurs fois. Que constatez vous ?
2. Ajouter le mot clé volatile à la déclaration de running. Que constatez vous ?

# Exclusion mutuelle

*Lire attentivement*, *puis compiler et exécuter* le programme ThreadTest2.java joint.

1. Répéter l’exécution plusieurs fois. Que constatez vous ?
2. Ajouter volatile à la déclaration de i. Que constatez vous ?
3. Trouver un moyen de corriger ce programme en utilisant la primitive synchronized.