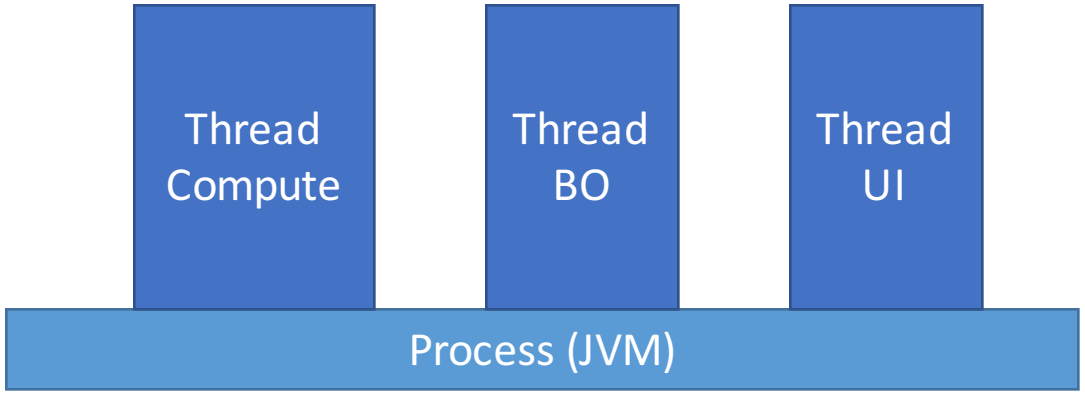
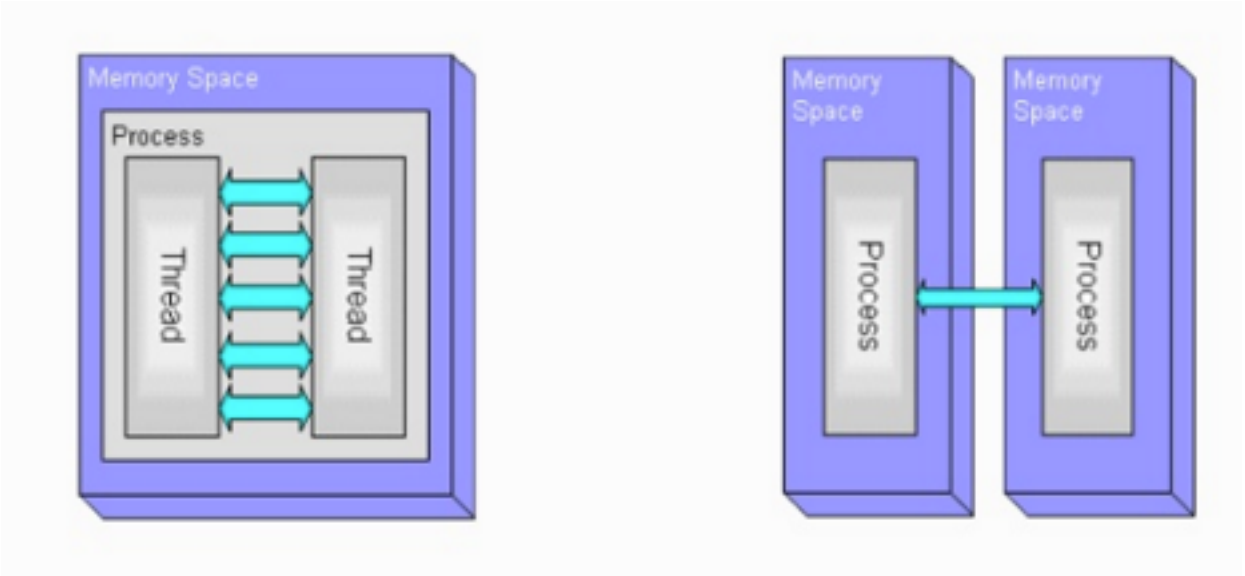
Process Vs Thread

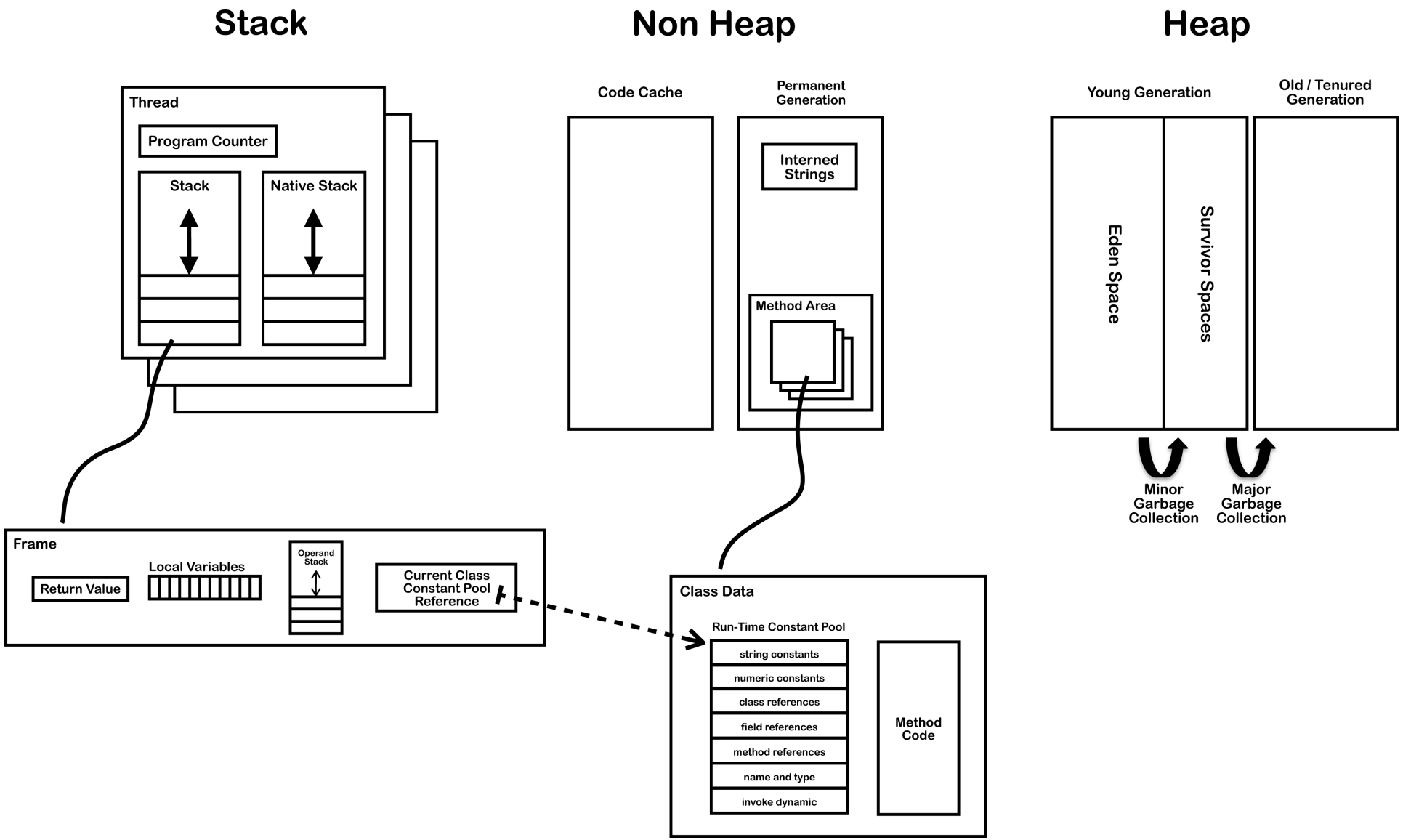
- Process :
  - Environnement d’exécution complet
  - Espace mémoire dédié
- Thread :
  - « Lightweight process »
  - Existe dans un process
  - Partage des ressources avec les autres Threads



Process Vs Threads



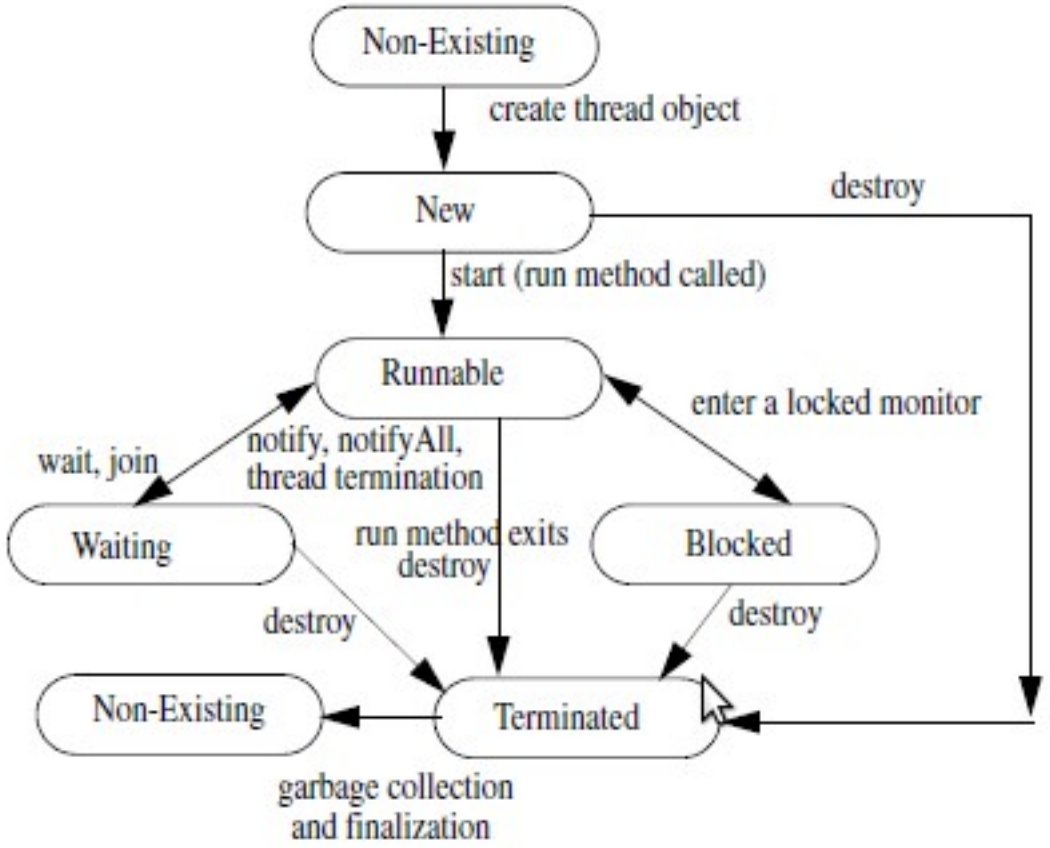
JVM


Stack
Non Heap
Heap
Thread
Program Counter
Stack
Native Stack
Code Cache
Permanent Generation
Interned Strings
Method Area
Young Generation
Old / Tenured Generation
Eden Space
Survivor Spaces
Minor Garbage Collection
Major Garbage Collection
Frame
Return Value
Local Variables
Operand Stack
Current Class Constant Pool Reference
Class Data
Run-Time Constant Pool
string constants
numeric constants
class references
field references
method references
name and type
invoke dynamic
Method Code

La stack

```
sun.misc.Unsafe.park(Native Method)
java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:226)
java.util.concurrent.LinkedTransferQueue.awaitMatch(LinkedTransferQueue.java:731)
java.util.concurrent.LinkedTransferQueue.xfer(LinkedTransferQueue.java:644)
java.util.concurrent.LinkedTransferQueue.poll(LinkedTransferQueue.java:1145)
java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1068)
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1130)
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
java.lang.Thread.run(Thread.java:724)
```

Les états


Non-Existing
create thread object
New
destroy
start (run method called)
Runnable
enter a locked monitor
wait, join
notify, notifyAll,
thread termination
Waiting
run method exits
destroy
Blocked
destroy
destroy
Non-Existing
Terminated
garbage collection
and finalization

Implémentation

- Héritage de Thread

```
public class SimpleThread extends Thread {
```

- Implémente Runnable

```
public class TestThread implements Runnable
```

Rendu

- Nom du dossier contenant le dossier du projet:
  - NOM_PRENOM
- Sujet du mail: AN_2_TP2
- Envoyer à mail@samimakki.fr

TP #3

- Créer une classe MemberThread qui va louer des livres de façon compulsive.
- Il doit implémenter Runnable et hériter de Member
- La fonction déclenche une boucle qui choisit aléatoirement entre 3 actions:
  - Emprunter un Book au hasard
  - Emprunter un Laptop au hasard
  - Rendre un Borrowable au hasard
- http://gitlab.samimakki.fr/cours/cours_tp2

Notation

- 1 – Lancer un thread
- 1 – Demo avec 3 threads en //
- 1 – Stopper les threads proprement lors de l'appui sur une touche
- 1 – Eviter les accès concurrents dans la bibliothèque
- 1 – Choisir aléatoirement un élément parmi une **collection** sans Math.random ou Random