

# Threads, thread-safe

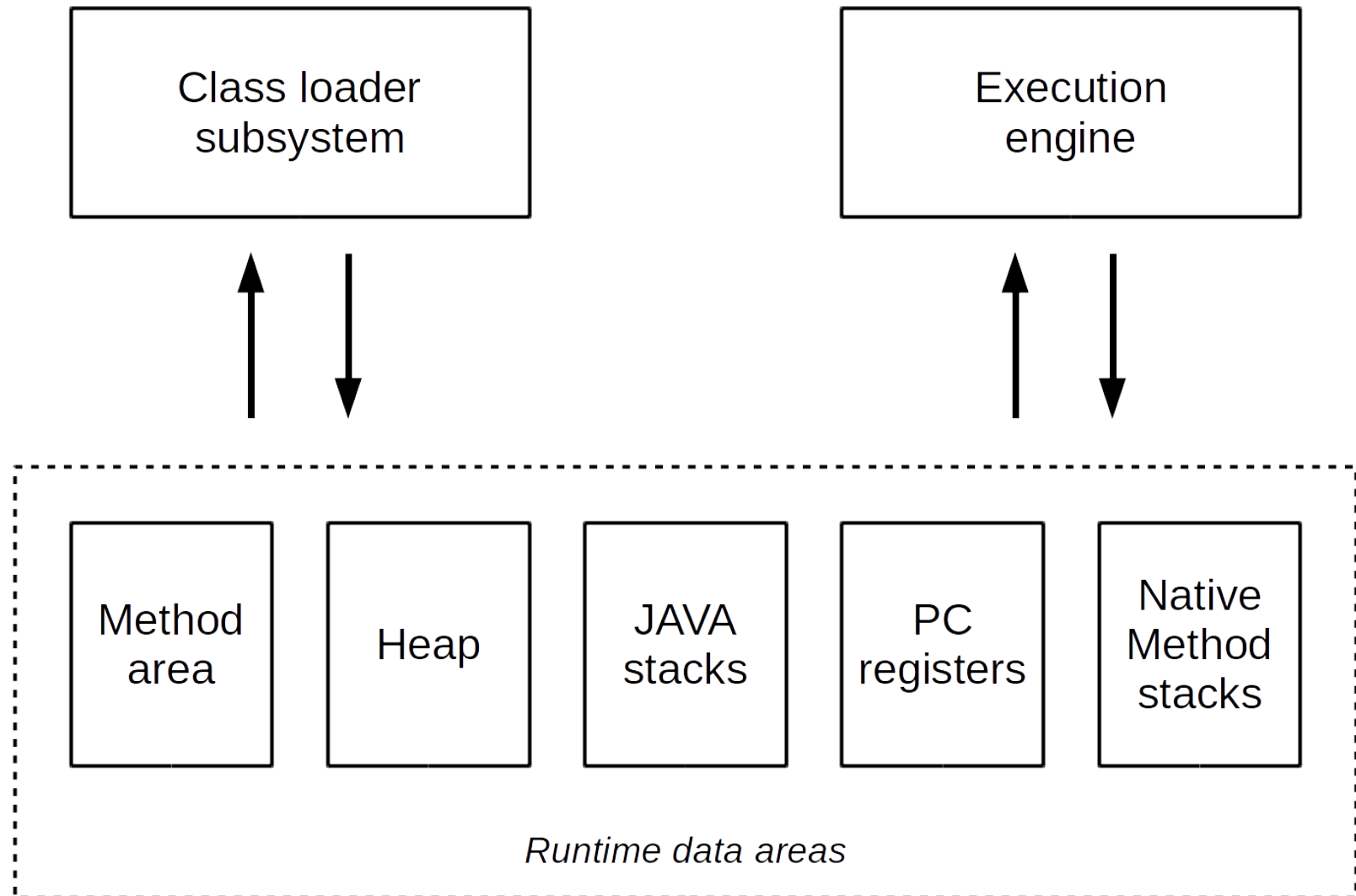
Leleux Laurent

2016 - 2017

# Thread ?

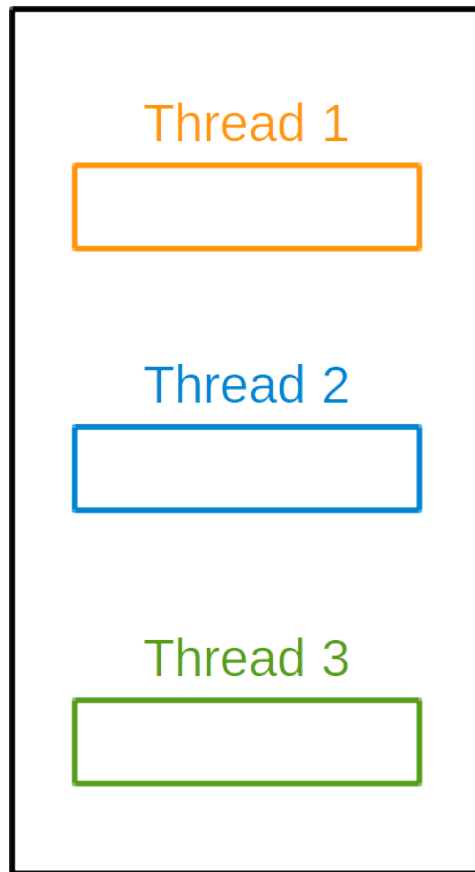
- Un fil d'exécution (processus ?)
- Partagé :
  - Method area
  - Heap
- Privé :
  - Stack
  - Pc-register

# JVM

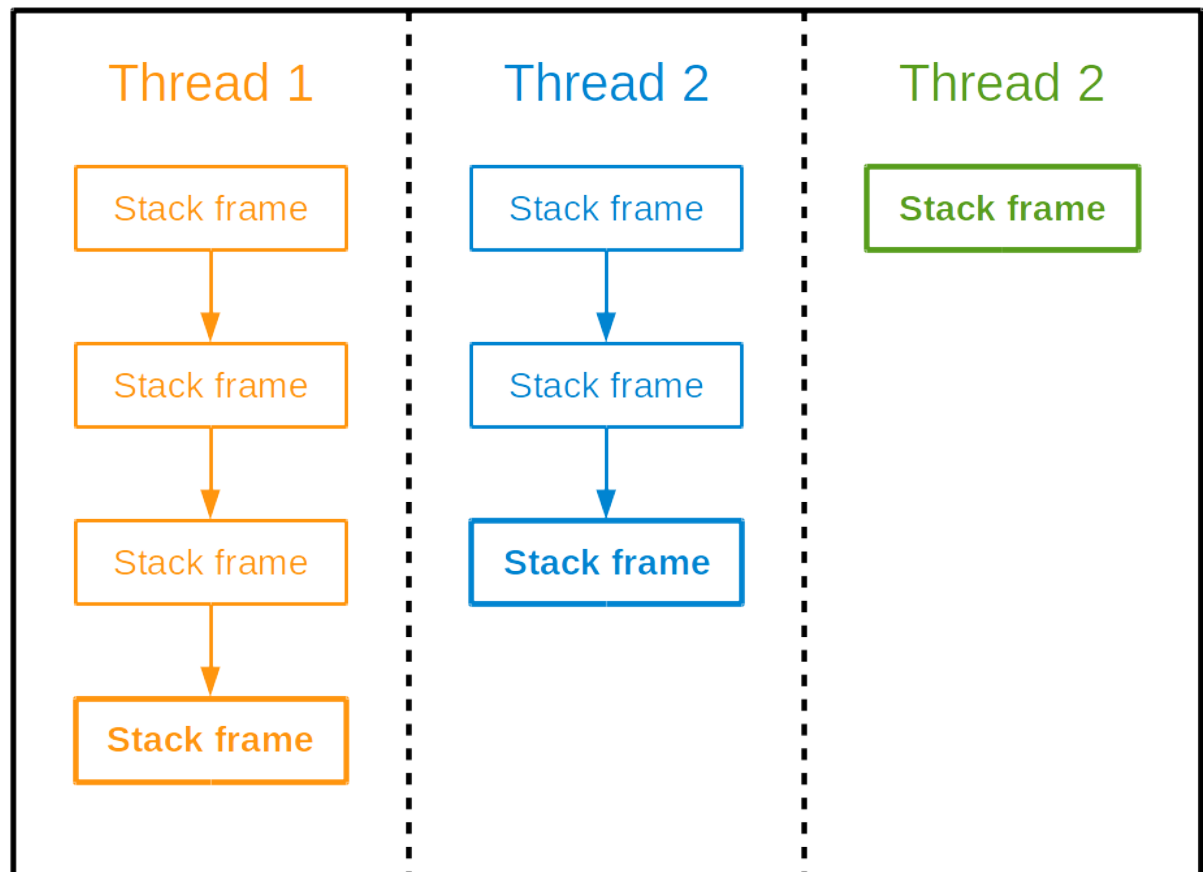


# JVM memory

## PC registers



## JAVA Stacks



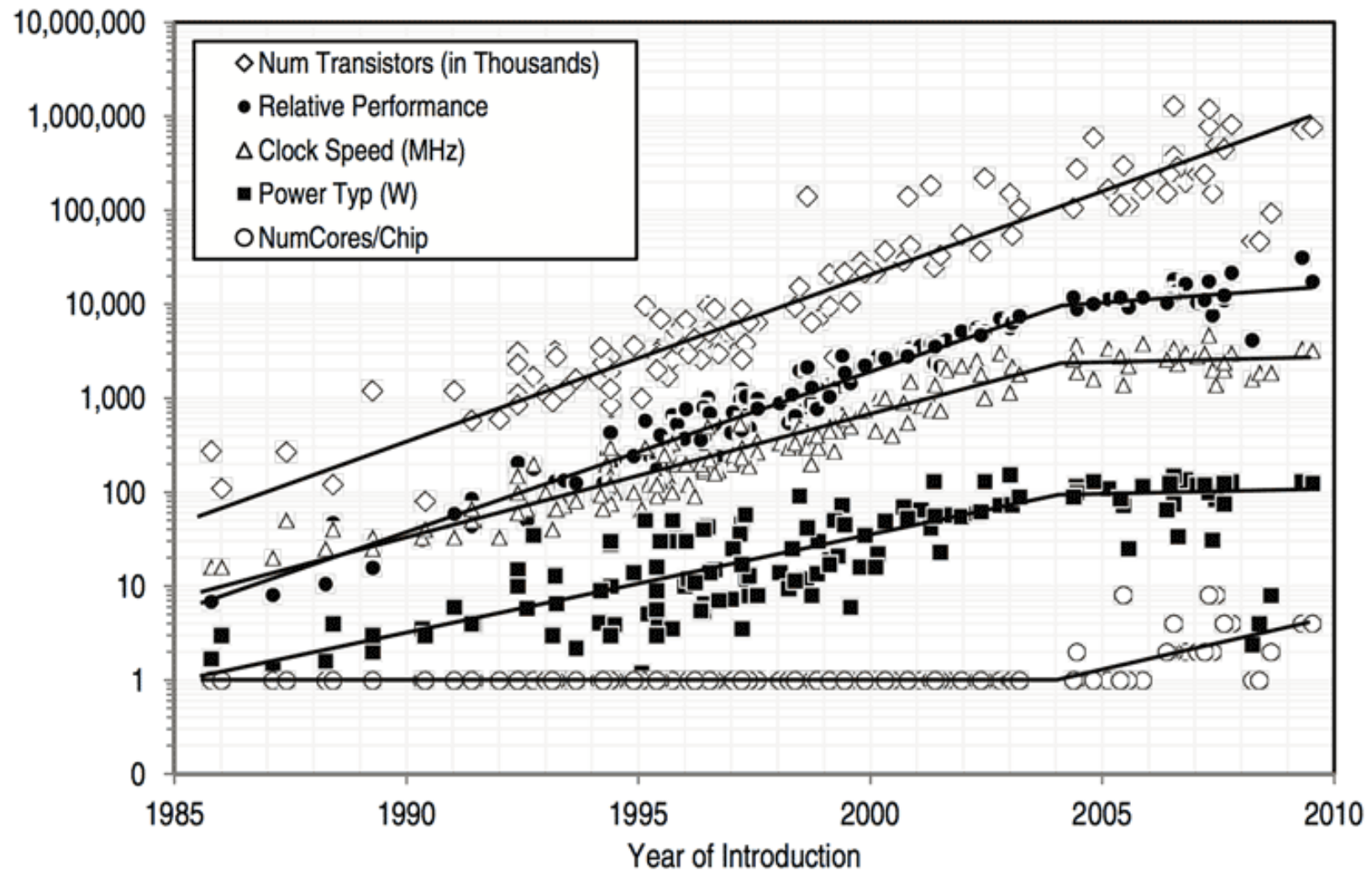
# Exemples

- Navigateur
- Serveur Web/FTP
- Accès DB
- Lecture de fichiers
- Garbage collector

# Avantages

- Parallélisme
  - I/O non bloquants
  - Performances
- Processeurs multi-cœurs

# Loi de Moore



# Partage de ressources

- Performances
- Synchronisation
- /!\ JVM multiples /!\



# Thread-safe

- Non partage
  - Thread-local
- Partage
  - Exclusion mutuelle
  - Opérations atomiques
  - Objets immuables

# Exclusion mutuelle

- Cadenas
- Pour classes et instances
- « synchronized » ferme le cadenas
- Deux écritures (method, statement)
- /\ Deadlocks /\

# Singleton

```
public final class Singleton {  
  
    private static Singleton instance = null;  
  
    private Singleton() {  
        super();  
    }  
  
    public final static Singleton getInstance() {  
        if (Singleton.instance == null) {  
            synchronized(Singleton.class) {  
                if (Singleton.instance == null) {  
                    Singleton.instance = new Singleton();  
                }  
            }  
        }  
        return Singleton.instance;  
    }  
}
```

# Immutable Objects

« Objets dont l'état ne peut plus changer après leur création. »

- Pas de setter, même indirects
- Attributs « final » et « private »
- Classe « final »
- Ne pas transmettre ni recevoir des références vers des objets mutables (copies si nécessaire)