#Appunti_22-11-22

Thread Pools sono risorse particolarmente pregiate che sono quelle che ci permettono di attivare i processi in concorrenza.

Un esecutore è una visione astratta di una risorsa di calcolo in grado di interpretare i programmi, si occupa principalmente di eseguire i Task, nel prima possibile.
Classe Executors ha due scopi:

- Punto d'accesso per tutte le attività che coinvolgono gli esecutori(façade)
- Versione astratta che costruisce oggetti (Factory)

Nelle interfacce Executor, ricevono un Runnable che deve essere eseguito, e con una loro politica interna arbitraria, decidono quando attivare questi task (in questa classe si chiamano command).
ExecutorService è un interfaccia che estende Executor, ci serve per implementare shutdown(); serve per chiedere di chiudersi in modo ordinato al threadpools e quindi ogni Task che viene inserita dopo lo shutdown() verrà rifiutata.

Partiamo dalla interfaccia EXECUTOR.JAVA

```
package it.unipr.informatica.concurrent;

public interface Executor{
        public void execute(Runnable command);
}
```

Appunto è un'interfaccia con un unico metodo Execute che riceve un Runnable, e una volta ricevuto lo mette in coda e prima o poi verrà eseguito, questo executor lancia anche un'eccezione *RejectedExecutionException* non viene dichiarata perchè è una RunTimeException, allora andiamo a dichiarare in RejectedExecutionException.JAVA dove estende RunTimeException.

```
package it.unipr.informatica.concurrent;

public class RejectedExecutionException extends RuntimeException{
        public RejectedExecutionException(String message){
                super(message);
        }
}
```

EXECUTORSERVICE.JAVA

```java
package it.unipr.informatica.concurrent;

public interface ExecutorService extends Executor{
	public void shutdown();
}
```

Come detto prima executorservice è un interfaccia che estende Executor, serve per implementare shutdown().

EXECUTOR.JAVA

```java
package it.unipr.informatica.concurrent;

public class Executors {
	public static ExecutorService newFixedThreadPool(int count) {
		return new SimpleThreadPoolExecutorService(count);
	}
	private Executors() {
		//Blank
	}
}
```

Sovrascriviamo il costruttore di default con uno vuoto, a questo punto chi può fare new? solamente il codice all'interno della classe stessa, se noi decidiamo di non mettere codice all'interno di questa classe che fa new allora non potrà essere istanziata e quindi non ci saranno mai oggetti Executors in giro per il sistema, a cosa ci serve? Ci serve perchè ci sono i metodi statici, e noi ne mettiamo uno che ci serve per costruire un ThreadPool di dimensione fissata *newFixedThreadPool* questo si limita a costruire una istanza si SimpleThreadPoolService che realizza un pool di Thread.

SIMPLETHREADPOOLEXECUTORSERVICE.JAVA

L'idea è abbastanza semplice quando costruiamo questo *SIMPLETHREADPOOLEXECUTORSERVICE* andiamo a costruire un numero di thread pari al argomento che ci viene passato, a questo punto attiviamo tutti i thread, mettiamo una coda bloccante, che contiene i task che vogliamo che vengano eseguiti, nel momento in cui i thread partono incominciano a fare get o take da questa coda bloccante, man mano che arrivano runnable i thread li prendono e li eseguono e ne chiedono un altro; il tutto va avanti finchè non viene attivato shutdown(), che quando questo viene attivato le richieste di accodamento non verranno accettate e terminerà solamente le task accodate prima che sia stata attivata.

Nel nostro programma creiamo una coda bloccante chiamata tasks cioè una coda di task da eseguire.