Opdracht JSF31, week 6

We gaan deze week opnieuw uit van de opdracht van week 4/5, en gebruiken dit keer een thread pool om meerdere berekeningen tegelijk te kunnen laten doen.

Doelstellingen

- Thread pool gebruiken met `Callable`
- `CyclicBarrier` gebruiken

Opdracht

Zorg dat de VM weer 4 CPU's gebruikt.
Maak een versie van de fractal applicatie die gebruik maakt van een thread pool en iedere keer als de fractal berekend moet worden drie tasks aan de threadpool aanbiedt: `generateLeftEdge`, `generateBottomEdge`, en `generateRightEdge`. We gebruiken nu dus geen gewone threads voor de berekeningen, zoals in opdracht 5.

Elke task moet door middel van een van `Callable` afgeleide klasse doorgegeven worden aan de thread pool. Net als de `Runnable` in opdracht 5, moet nu elke `Callable` een eigen `KochFractal` object krijgen. De `Callable` is Observer van zijn `KochFractal` object, en heeft dus de `update` methode.
Geef echter in dit geval elke thread (`Callable`) zijn eigen lokale `private ArrayList`, waar de thread al zijn berekende edges in zet. Door middel van `Futures` moeten deze `ArrayLists` doorgegeven worden aan de Java FX Application Thread.

Gebruik nu geen gemeenschappelijke variabele `count`, maar een `CyclicBarrier` om te bepalen wanneer alle berekeningen klaar zijn, en er dus getekend kan worden. Let er ook nu weer op dat de Java FX Application Thread niet geblokkeerd wordt terwijl de threads aan het rekenen zijn.
De benodigde reken- en tekentijd moet nu ook weer weergegeven worden.

Toon het eindresultaat aan de docent.