1.Using threads, create a simple Simulation for the Dining Philosophers Problem.

Ich erstelle eine möglichst einfache implementation in der alle philosophen gleichzeitig essen wollen, um die situation zu simulieren in der ein deadlock eintstehen könnte, dies wird in einem bestimmten Intervall immer weiter simuliert.

Hinschreiben wie ich mit Runnable implementiert hab – Korrektur – nicht mit Thread extenden

2.For each Philosopher, create a Thread within this Philosopher dines. The implementation should just create some outputs like:

Ich habe es so zum funktionieren gebracht und die threads nacheinander gestartet, bei jedem durchlauf des Programms schafft es ein anderer Philosoph die Gabeln zu bekommen, die Threads werden also nicht hinterenander ausgeführt.

```
Platon cant take the leftFork
Erasmus aus Rotterdamm cant take the right F
Thomas von Aquin eats
Seneca cant take the leftFork
Seneca cant take the right Fork
Platon cant take the right Fork
```

```
Platon cant take the leftFork
Seneca cant take the leftFork
Thomas von Aquin eats
Seneca cant take the right Fork
Erasmus aus Rotterdamm cant take the leftFork
Erasmus aus Rotterdamm cant take the right Fork
```

```
Seneca eats
Pythagoras cant take the right Fork
Thomas von Aquin cant take the leftFork
Erasmus aus Rotterdamm cant take the right Fork
Platon cant take the leftFork
Platon cant take the right Fork
```

```
Thomas von Aquin eats
Platon cant take the leftFork
Pythagoras cant take the right Fork
Seneca cant take the right Fork
Erasmus aus Rotterdamm cant take the leftFork
s aus Rotterdamm cant take the right Fork
```

```
Thomas von Aquin cant take the leftFork
Thomas von Aquin cant take the right Fork
Erasmus aus Rotterdamm cant take the leftFork
Platon cant take the leftFork
Erasmus aus Rotterdamm cant take the right Fork
Platon cant take the right Fork
```

deadlock

Provide several implementations with a sensible way to switch between them; at least one blocking one and one using a probabilistic solution to avoid the deadlock.

Um sicherzugehen, dass der array, welcher den Gabelstatus speichert, immer aktuell und korrekt ist, erstelle ich eine getter und Setter methode dieses Arrays, welche ich beide mit dem synchronized keyword schreibe – so kann ich sichergehen dass die Threads die einzelnen gabelpositionen nicht durcheinanderbringen

1. Idee :

alle Threads starten – danach müssen alle ausser dem ersten warten – wenn der erste fertig ist, kommt dann der 2 usw.