Programmier-Paradigmen

Tutorium – Gruppe 4 & 8

Henning Dieterichs

Akka Aktoren / Design By Contract

Aufgabe 1.1: Einfacher Nachrichtenaustausch

- Erstelle einen Aktor *Kid*, die auf Schimpfwort-Nachrichten (Strings) eine genervte Antwort auf die Konsole schreibt und ab der vierten Nachricht nach der Mutter ruft.

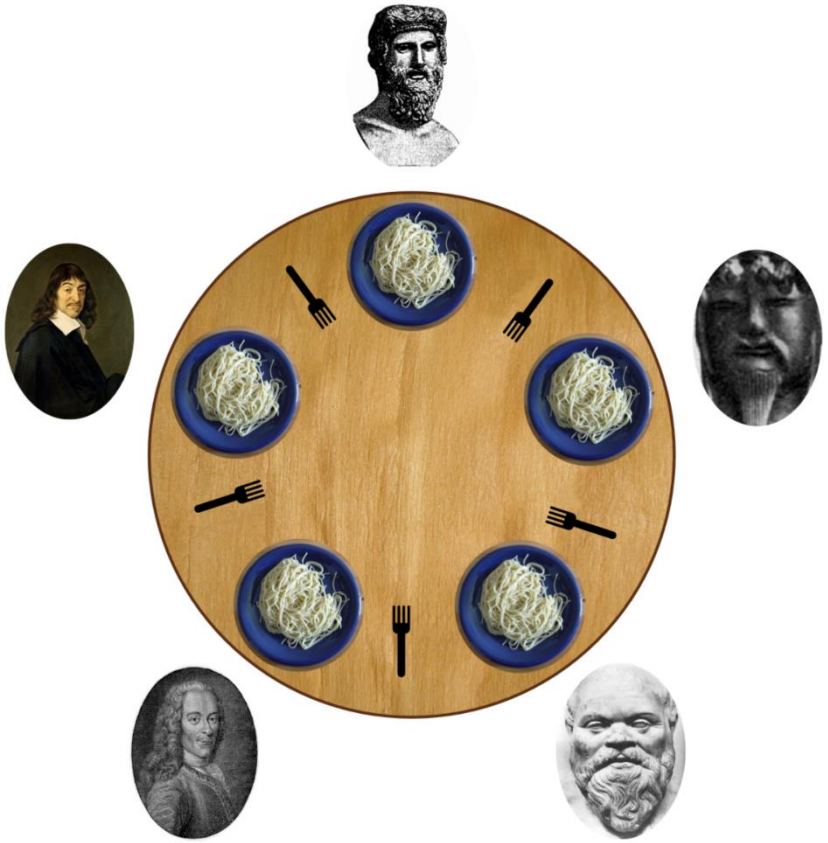
```
public class TeaseLisa {
    0 references
    public static void main(final String[] args) throws InterruptedException {
        ActorSystem actorSystem = ActorSystem.create("TeaseLisa");
        ActorRef lisa = actorSystem.actorOf(Props.create(Kid.class));
        lisa.tell("idiot", ActorRef.noSender());
        lisa.tell("muppet", ActorRef.noSender());
        lisa.tell("idiot", ActorRef.noSender());
        lisa.tell("muppet", ActorRef.noSender());
        Thread.sleep(1000);
        actorSystem.terminate();
    }
}
```

Aufgabe 1.2: Ping Pong

- Aktor gibt *„<name> received <wert>“* aus, sendet *wert+1* zurück
- Zwei Aktoren, einer sendet die erste Nachricht an den anderen

Aufgabe 2: Speisende Philosophen

- Erläutere das Aktorenkonzept
  - Vorteile gegenüber Threads?
  - Relevanz für das Dining-Philosophers Problem
- Implementiere Dining-Philosophers
  - Mit Java-Aktoren



Aufgabe 3: Design by Contract - Warmup

- Finde alle Contract-Verletzungen!

Aufgabe 4: Design by Contract

- Spezifiziere Kontrakte für *hire* und *fire*
  - Ein Mitarbeiter kann nur bei einer Firma angestellt sein.
  - Es darf nur versucht werden, einen Mitarbeiter einzustellen, wenn dieser noch nicht in der Firma angestellt ist.
  - Es darf nur versucht werden, einen Mitarbeiter zu feuern, wenn dieser in der Firma angestellt ist.
- Implementiere das Interface korrekt (mit assert)
- Diskussion: assert vs JML (Java Modeling Language)/OCL (Object Constraint Language)