

AOP & Microservices

Für die Einsendeaufgabe „AOP & Microservices“ wurde in der Entwicklungsumgebung *IntelliJ* ein *AspectJ* Beispiel erstellt. Hierfür wurde die Funktionsweise des Java Raytracers skizziert und durch das Einweben einer Methode gezeigt.

```

1 public class RaytracingDemo {
2     public void traceRays() {
3         System.out.println("Rays werden verfolgt...");
4     }
5
6     public static void main(String[] args) {
7         RaytracingDemo raytracingDemo = new RaytracingDemo();
8         raytracingDemo.traceRays();
9     }
10 }

```

Abbildung 1: **RaytracingDemo** Klasse, welche einen `System.out.println` mit der Funktionsbeschreibung eines einfachen Raytracers enthält.

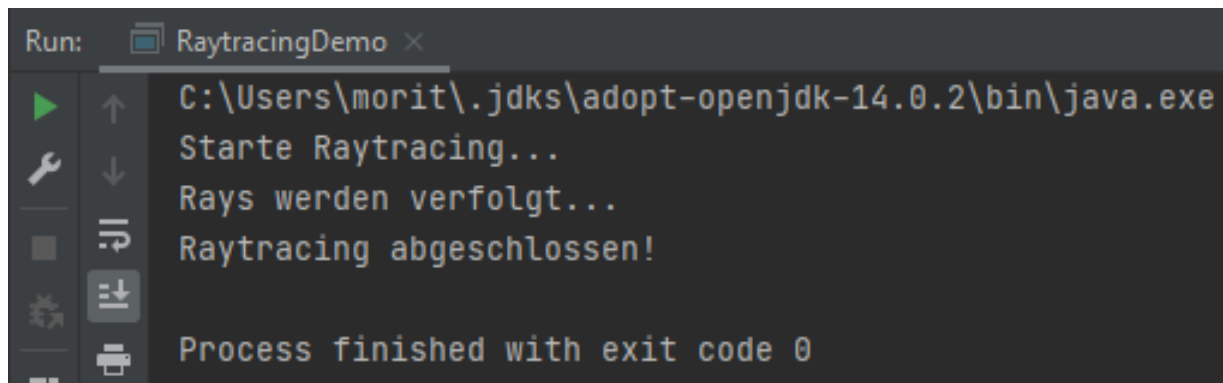
Zusätzlich wird eine *Aspect* .aj-Datei erzeugt, welche in den Programmablauf der **RaytracingDemo** verwoben wird. In dieser Datei werden durch zwei weitere `System.out.println` Befehle der Start und das Ende des Programmablaufs in der Konsole ausgegeben. Durch `void around` wird die *Aspect* Datei um die Methode **traceRays()** in der **RaytracingDemo** herumgewoben. `:call` bestimmt die Zielmethode. Die Funktion `proceed()` bestimmt den Zeitpunkt der Initiierung der Methode **traceRays()**.

```

1 public aspect AspectRaytracer {
2     void around():call(void RaytracingDemo.traceRays()){
3         System.out.println("Starte Raytracing...");
4         proceed();
5         System.out.println("Raytracing abgeschlossen!");
6     }
7 }

```

Abbildung 2: .aj Datei mit dem verwobenen Programmablauf der **traceRays()** Methode



```
Run: RaytracingDemo x
C:\Users\morit\.jdk\adopt-openjdk-14.0.2\bin\java.exe
Starte Raytracing...
Rays werden verfolgt...
Raytracing abgeschlossen!
Process finished with exit code 0
```

Abbildung 3: Ausgabe in der Konsole resultierend aus der **RatracingDemo** inklusive verwobener **AspectRaytracer.aj**