Höhere Programmierkonzepte
Praktikum I

Prof. Dr. Nikolaus Wulff

27. Okt – 05. Nov 2020

1 *Axela* und *Iris*

Im Rahmen des HPK Praktikums werden Sie „*An eXtendable Expression Language Abstraction*“ entwickeln und Stück für Stück inkrementell mit jedem Praktikum weiter ausbauen. *Axela* soll beliebige Probleme lösen können – was auch immer das im konkretem Kontext heißen mag und wie auch immer dies geschieht. *Axela* ist quasi ein kleines Wunder. Vollbracht wird dieses Wunder mit Hilfe von vielen „*Intelligent Rulebased Inference Solver*-Instanzen“, von denen sich beliebig viele bei *Axela* registrieren, verwalten und passend zur Aufgabenstellung heranziehen lassen können.

```
/**
 * Intelligent Rulebased Inference Solver (IRIS).
 */
public interface Iris<Problem, Facts, Solution> {
  /**
   * Solve a problem using the given facts and return the solution.
   * @param problem to solve
   * @param facts to use
   * @return solution
   */
  Solution solve(Problem problem, Facts facts);
}
```

Listing 1: Iris.java: Die generische *Iris* Schnittstelle.

Iris fungiert ähnlich wie ein Micro-Service in der Cloud, nur hier in einer *Axela* - Ablaufumgebung. Technisch verwaltet *Axela* intern die einzelnen *Iris* Instanzen mit Hilfe eines assoziativen Containers[1] vom Typ *java.util.Map*$<K,V>$. Was genau Key (K) und Value (V) sein werden gilt es in den folgenden Übungen und Praktika zu erarbeiten.

[1] Wem diese Container Klassen nichts mehr sagen eine kurze Übersicht ist in meiner Informatik II Vorlesung vom 16.05.2019 unter „Programmieren in Java“ auf dem Lab4Inf Server zu finden. Java Kenntnisse auf diesem Niveau werden für HPK vorausgesetzt.

Iris ist eine einfache generische Schnittstelle, die innerhalb einer `solve`-Methode zu gegebenem Problem und Fakten eine Lösung findet. Die Schnittstelle ist mittels Generics des JDK1.5 formuliert, welches 2004 veröffentlich wurde. Die Verwendung und Codierung von Generics wird im Rahmen der HPK Vorlesung behandelt, für Praktikum I nehmen Sie die drei Platzhalter `Problem`, `Facts` und `Solution` erst einmal so hin – es gibt keine Klassen oder Schnittstellen mit diesen Namen. Auch ist es möglich die *Iris* Schnittstelle mittels λ-Ausdrücken zu implementieren, ein Konzept des JDK1.8 (erschienen 2014), das Sie ebenfalls im Rahmen der Vorlesung behandeln werden.

Vermutlich klingen Ihnen bereits jetzt von dem vielen Fachjargon die Ohren, die wenigen Zeilen des Listing (1) haben es also „in sich“ und bieten viel Stoff für Diskussionen – Sie werden sich langsam innerhalb der Vorlesung an den aktuellen Stand der Java Sprache heranarbeiten dürfen.

Aufgabe

Genug der Vorrede, fangen wir mit Praktikum I an. *Axela* soll als ein Eclipse-Gradle Multiprojekt mit vielen Kindprojekten aufgesetzt sein, ähnlich zu meinem bereitgestellten Übungsvideos zum „gproject“. Jedes der Kindprojekte gehört zu einem anderem Praktikum und löst einen anderen Problembereich per passendem *Iris* Service. Einen solchen einfachen Service gilt es in diesem Praktikum zu entwickeln, bevor Sie sich an *Axela* wagen, d.h. in diesem Praktikum I ist *Axela* nur ein Name und wird noch nicht implementiert. Wichtiges Ziel dieses Praktikums ist es die (Multi)Projektstruktur hinzubekommen und erfolgreich im Git zu versionieren und zu bauen.

1. Legen Sie mit gradle init das Elternprojekt „*Axela* “ an. Bei der gradle Auswahl geben Sie als Projekt Typ 1: basic an, da dieses Projekt nur die äußere Klammer ohne eigene Java Verzeichnisse sein soll.

2. Innerhalb des Elternprojekts werden zwei Kindprojekte „*Axela.Core*“ und „*Axela.Client*“ als Projekt Typ 3: library angelegt. Das Client Projekt dient im Wesentlichem zum White-Box Test der *Axela* Anwendung und wird daher davon abhängig sein.

3. In das *Axela.Core* Projekt werden nun die zur Verfügung gestellten Sourcen mit den wichtigen Abstraktionen im Paket *de.lab4inf.axela* hinterlegt. Machen Sie sich mit der richtigen Einstellung des Classpath in der Entwicklungsumgebung vertraut.

4. Erstellen Sie nach dem Anlegen des Pakets *de.lab4inf.axela.simple* die Klasse `Concatinator` als Implementierung eines *Iris* - Service mit der Bindung Concatinator implements Iris<String,String[],String>. Das „Problem“ dieser Implementierung ist die Zeichenkette „CONCAT“ und als „Fakten“ wird ein beliebiges Feld von Zeichenketten an die

`solve` Methode übergeben, zurückgeliefert wird die aus den einzelnen Zeichenketten zusammengesetzte – d.h. concatinierte Zeichenkette – als Lösung.

5. Gradle wird Ihnen auch ein test Verzeichnis generiert haben mit einem einfachen Library Test, wie in meinem Video zu sehen. Hier sollen Sie Ihre eigenen JUnit Tests[2] ¸hinterlegen, um Black-Box Testszenarien zu ermöglichen. Für `Concatinator` ist bereits ein kleiner rudimentärer JUnit Test in den Sourcen enthalten, der Ihnen den Einstieg erleichtern und am Ende des Praktikums funktionieren soll. Er stellt sozusagen das Missing Link zu meiner dürftigen Spezifikation der Implementierung aus Teilaufgabe 4 dar. Sie entwickeln den `Concatinator` per *test driven development* (TDD) nach der Test-First Methode.

6. Entwickeln Sie ihren `Concatinator` bis der `ConcatinatorTest` „grün“ wird. Glückwunsch: Sie haben Ihren ersten *Iris* Service entwickelt!

7. Versionieren Sie ihr *Axela* Projekt innerhalb des FH Gits passend zu Ihrer Gruppeneinteilung und stellen Sie sicher, dass beide Projektpartner sich diese Sourcen auf Ihren jeweiligen Arbeitsrechnern teilen können, d.h. beide erfolgreich Ein- und Ausschecken können.

8. Versuchen Sie im Git-Lab das Projekt automatisch bauen zu lassen. Sobald auch dies gelingt sind Sie so weit, dass CI/CD (continious integration und delivery) für Ihr Projekt erfolgreich läuft.

Hinweise

Verzweifeln Sie nicht, falls Ihnen nicht gleich alle Aufgabenstellungen auf Anhieb klar werden oder Ihre Lösungen nicht funktionieren. Wäre dem so hätten Sie ja nichts Neues in HPK lernen brauchen. Daher: Fragen Sie rechtzeitig im Forum und ZOOM und kontaktieren Sie die Tutoren oder mich! Dies ist 100-mal besser als fremden Quellcode „zusammen zu googlen“. Sourcen die Sie selber nicht verstehen und die Ihnen im Studium nicht weiter helfen werden.

Eine Vorlage für *Axela* und *Iris* finden Sie als Startgrundlagen in der zum Praktikum I gehörenden ZIP-Datei auf dem Lab4Inf-Server. In einem Übungsvideo wird die Entstehung dieser Abstraktionen erläutert.

[2] Eine sehr rudimentäre Einführung zu JUnit finden Sie in meiner Informatik II Vorlesung vom 12.04.2019. Diese Testframework hat sich mittlerweile erheblich weiterentwickelt und wir werden sicherlich die ein oder andere Übungsstunde zu diesem Thema verwenden.