

## Profil

### Dipl.-Inform. Jens Fransson

Senior Softwarearchitekt und Softwareentwickler



Jens Fransson, Jahrgang 1972, konzipiert und entwickelt seit 1996 selbständig Java-Anwendungen. Besonderes Augenmerk legt er auf Clean Code, SOLID-Prinzipien, Refactoring und Test Driven Development. Er bevorzugt einen funktionalen Programmierstil.

## Technische Schwerpunkte

- Java Experte, Erfahrung seit 1996
- Clean Code, SOLID-Prinzipien, funktionaler Programmierstil
- Refactoring, Test-Driven Development
- Continuous Integration & Delivery

## Frameworks & Tools

- IntelliJ Idea
- Spring/Spring Boot, Google Guice
- Swing, SWT, Eclipse RCP, Tycho, Vaadin, CUBA, HTML/CSS, Bootstrap
- Maven, Gradle, Git Jira, Jenkins

## Themen

- Agile, Scrum
- Rapid Web Application Development
- Branchen: Logistik, Finanzen, Börse, Industrie, Verlage, Energieversorgung
- Geschäftsanwendungen, Handelssysteme, Quantitative Finance (Quant Developer)

*Good programmers write code that humans can understand.*

— Martin Fowler in *Refactoring*

## Zur Person

Name	Jens Fransson
Ausbildung	Universität, Hamburg Abschluss: Diplom-Informatiker
Berufserfahrung seit	1995
Fremdsprachen	<ul style="list-style-type: none"> <li>• Deutsch (Muttersprache)</li> <li>• Englisch (fließend)</li> <li>• Französisch (gut)</li> </ul>

## Kompetenzen

Schwerpunkte	Softwareanalyse, -design und -implementierung; Code-Reviews, Qualitätssicherung, Clean Code, Refactoring
Spezielle Kenntnisse	Java (seit 1996)

## IT-Kenntnisse

Java (Schwerpunkt, Erfahrung seit 1996)	Java-Versionen, JDK 1.0 bis Java 15, Java EE, J2EE, JSP (JavaServer Pages), Servlets, Spring Core, Spring MVC, Spring Security, Spring Social, Hibernate, JPA (Java Persistence API), AWT, Swing, SWT, JFace, Eclipse RCP, Tycho, JDBC, GWT (Web-GUI), Vaadin (Web-GUI + Backend), CUBA, Struts, Taglibs, JSF (Java Server Faces), Java Cryptography, GCJ (Gnu Compiler for Java), Excelsior Jet (Nativer Compiler für Java), JAXP (Java API for XML Processing), JDOM (Java Document Object Model), MDR (Meta Data Repository), Jakarta Log4J, JUnit, Java Regular Expressions, Java 2D, Piccolo, Jazz (ZGUI Rahmenwerke), JFreeChart (Visualisierung von Daten), JNI (Java Native Interface)
Weitere Programmiersprachen	Object Pascal, Borland Delphi, VCL Rahmenwerk, Hypertext Preprocessor (PHP), C/C++, STL, Win32 API, Mumps, Microsoft C#, .NET Rahmenwerk, JavaScript, HTML 5, CSS 3, jQuery
Datenbanken	SQL, Oracle, H2, HSQLDB, MariaDB, MySQL, DB2, Borland Database Engine (BDE), MS Access, db4o (Objektorientierte Datenbank)
Markup-Sprachen	XHTML, HTML, CSS, XML, XSL, XSLT (XML Stylesheets), SGML, Bootstrap
Kommunikation	TCP/IP, RMI, JMS (Java Message Service), Serielle Schnittstelle

## IT-Kenntnisse

Entwicklungsumgebungen	IntelliJ Idea, Eclipse, Netbeans, Borland Delphi, MS Visual Studio .NET, Oxygen
Modellierungswerkzeuge	Together, Visual Paradigm for UML, Rational Rose, XDE, MagicDraw
Betriebssysteme	Windows 10, 7, Vista, ..., 3.1, MS-DOS, Unix (Linux, Solaris)

## Projekte / Tätigkeiten

<b>Branche / Bereich</b>	<b>Logistik</b>
Zeitraum	08/18 – heute
Projektziel	Weiterentwicklung und Pflege einer Eclipse RCP Anwendung im Logistikumfeld. Die Anwendung dient als Plattform für per OSGi angebundene Kundenmodule (die wiederum eigene Anwendungen darstellen). Sie umfasst ca. 160.000 Codezeilen und läuft auf ca. 40.000 Client-Rechnern. Die Anwendung wird über eine weitere, eigens implementierte Service-Anwendung auf den Client-Rechnern ausgeliefert.
Aufgaben / Verantwortlichkeiten	<ul style="list-style-type: none"> <li>• Erfassung von Änderungswünschen und Fehlerberichten für die Anwendungen in Atlassian Jira.</li> <li>• Dokumentation der erforderlichen Prozesse im Wiki.</li> <li>• Implementation der Anforderungen in Java mit IntelliJ Idea.</li> <li>• Überwachung, Wartung und Weiterentwicklung der Continuous Integration auf Jenkins.</li> <li>• Unregelmäßige Erstellung und Veröffentlichung von Releases der RCP Anwendung.</li> <li>• Integration der Bibliothek JxBrowser zur Anbindung von Kundenmodulen, die als Web-Anwendung implementiert sind.</li> </ul>
Methoden / Technologien	Java SE 1.8, IntelliJ Idea, Spring 4, SWT, Eclipse RCP, OSGi, Maven, Tycho, Nexus, SonarQube, Jira, ClearQuest, Jenkins, Maven, Git/BitBucket

<b>Branche / Bereich</b>	<b>Logistik</b>
Zeitraum	08/16 – heute
Projektziel	Weiterentwicklung einer umfangreichen Java/Spring Swing-Anwendung im Logistikumfeld. Die Anwendung umfasst ca. 3,5 Millionen Codezeilen und läuft auf ca. 40.000 Client-Rechnern. Das Backend läuft auf dynamisch skalierenden Server-Instanzen.
Aufgaben / Verantwortlichkeiten	<ul style="list-style-type: none"> <li>• Die Anwendung ist in Unterprojekte gegliedert, für jedes Unterprojekt ist ein Team von ca. 10 Entwicklern zuständig. Insgesamt arbeiten ca. 100 Entwickler in dem Projekt.</li> <li>• Aufgaben: Full Stack – Entwicklung neuer Funktionen, sowohl im Frontend (UI: Swing, Angular) wie auch im Backend (Java Spring Services, JPA, Hibernate, Oracle, Tomcat 7), inklusive der erforderlichen Unit-Tests. Beheben von Fehlern.</li> </ul>
Methoden / Technologien	Java SE 1.8, IntelliJ Idea, Spring 4, Swing, Oracle, Hibernate, Maven, Nexus, Tomcat 7, SonarQube, Jira, ClearQuest, Jenkins, Maven, Git/BitBucket

## Projekte / Tätigkeiten

<b>Branche / Bereich</b>	<b>Börsenhandel</b>
Zeitraum	09/15 – heute
Projektziel	Entwicklung einer Java-Anwendung zur Simulation von Handelssystemen für Finanzinstrumente (Aktien, Futures, Forex).
Aufgaben / Verantwortlichkeiten	<ul style="list-style-type: none"> <li>• Clean Code – Funktionaler Programmierstil (fast alle Objektinstanzen sind unveränderlich)</li> <li>• Durchgängige Einhaltung der SOLID Prinzipien</li> <li>• Umfang der Anwendung: ca. 100.000 Lines of Code, ca. 1.800 Unit Tests. Laufzeit aller Tests ca. 12 sec.</li> <li>• Konfiguration der Anwendung mittels Spring Boot Profile.</li> <li>• Aufgaben: Entwicklung neuer Funktionen inklusive der erforderlichen Unit-Tests.</li> </ul>
Methoden / Technologien	Java SE 15, IntelliJ Idea, Spring Boot 2, Mockito, Git, Gradle 6, TeamCity

<b>Branche / Bereich</b>	<b>Logistik</b>
Zeitraum	02/15 – 06/16
Projektziel	Weiterentwicklung einer umfangreichen Java/Java EE Web-Anwendung im Logistikumfeld.
Aufgaben / Verantwortlichkeiten	<ul style="list-style-type: none"> <li>• Die Anwendung ist in Unterprojekte gegliedert, für jedes Unterprojekt ist ein Team von ca. 5 – 10 Entwicklern, 2 – 5 Requirement Engineers und Business Stakeholdern zuständig. Insgesamt arbeiten ca. 100 Entwickler in dem Projekt.</li> <li>• Aufgaben: Full Stack – Entwicklung neuer Funktionen, sowohl im Frontend (UI: JSF, HTML, CSS) wie auch im Backend (Java EE Services, EJBs, JPA, EclipseLink, Oracle, GlassFish 3), inklusive der erforderlichen Unit-Tests. Beheben von Fehlern. Dokumentation der Implementierung.</li> </ul>
Methoden / Technologien	Java SE 1.7, Java EE 6, JSF, HTML, CSS, Oracle, EclipseLink, GlassFish 3, Google Chrome, SonarQube, ClearQuest, Jenkins, Gradle, Git/Gerrit, IntelliJ Idea, Eclipse

## Projekte / Tätigkeiten

<b>Branche / Bereich</b>	<b>Energieversorger</b>
Zeitraum	11/12 – 06/14
Projektziel	Weiterentwicklung einer Client/Server Enterprise-Anwendung mit Swing GUI
Aufgaben / Verantwortlichkeiten	<ul style="list-style-type: none"> <li>• Die bestehende Enterprise-Anwendung zur Planung und Optimierung von Kraftwerkseinsätzen wird um ein neues Modul zur Planung von Stillständen (Revisionen, Störfälle etc.) erweitert</li> <li>• Hierfür werden neue GUI-Ansichten (GUI-Schicht), Service-Methoden (Geschäftslogik) und Entitäten (Persistenzschicht) entwickelt</li> <li>• Die Entwicklung der Anwendung erfolgt vor Ort in enger Abstimmung mit dem Anwender</li> <li>• Die Unterstützung des Anwenders im laufenden Betrieb gehört ebenfalls zum Aufgabenfeld</li> </ul>
Methoden / Technologien	Java SE 1.6, Swing, Jide, Oracle Weblogic 12 Application-Server, Oracle, H2 SQL, Spring, iBatis, MyBatis Persistenz-Framework, RMI, JMS, Subversion V, Jenkins Build-Server, BoFiT, Gurobi, IntelliJ Idea, Eclipse

<b>Branche / Bereich</b>	<b>Mobile</b>
Zeitraum	09/14 – 12/14
Projektziel	Evaluierung der Eignung aktueller Java-basierter Web-Rahmenwerke zur Realisierung einer mobilen Single Page App (SPA)
Aufgaben / Verantwortlichkeiten	Bei dieser Untersuchung aktueller Java-basierter Web-Rahmenwerke wurden die derzeitigen Möglichkeiten zur Implementierung einer Single Page App in Java bewertet
Methoden / Technologien	Java SE 1.7, 1.8, GWT, Vaadin, Spring MVC, Spring Security, Spring Social, OAuth2, Gradle, Git, IntelliJ Idea

## Projekte / Tätigkeiten

<b>Branche / Bereich</b>	<b>Interbankenhandel</b>
Zeitraum	04/09 – 11/12
Projektziel	Entwicklung und Ausführung eines Handelssystems zur Abwicklung von Devisengeschäften
Aufgaben / Verantwortlichkeiten	<ul style="list-style-type: none"> <li>• Ein Prototyp des Handelssystems wird auf der Plattform MultiCharts in der Programmiersprache PowerLanguage entworfen. Auf dieser Plattform findet auch die Überprüfung mittels historischer Kursdaten statt.</li> <li>• Die Validität des Handelsmodells und die fachliche Funktionalität wird im Prototypen sichergestellt. Zur Ausführung von Handelsgeschäften ist MultiCharts jedoch nicht ausreichend. Daher wird das Handelssystem auf der Zielplattform OpenQuant in der Programmiersprache C# neu entwickelt. Die Ausführung der Handelsgeschäfte wird in Echtzeit überwacht.</li> <li>• Zum Abruf der historischen Kursdaten wird ein Java-Programm entwickelt. Die Entwicklung erfolgt mit der Entwicklungsumgebung IntelliJ Idea. Die Kursdaten werden als persistente Beans in der SQL-Datenbank H2 gespeichert.</li> <li>• Zur Vereinfachung des Umganges mit der Datenbank kommt das Apache Commons Framework DBUtils zum Einsatz. Die Daten liegen in sekundengenaue Auflösung vor. Aufgrund der hieraus resultierenden großen Mengen an Daten sind zahlreiche Maßnahmen zur Sicherstellung einer hohen Performance nötig. Das Datenmodell ist besonders platzsparend und effizient ausgelegt. Die Daten werden nicht nur über verschiedene Tabellen verteilt, sondern auch in nach Zeiträumen unterteilte Datenbanken abgelegt.</li> </ul>
Methoden / Technologien	MultiCharts (PowerLanguage), OpenQuant (C#), IntelliJ Idea, H2 SQL, DBUtils (Apache Commons Framework)

<b>Branche / Bereich</b>	<b>Finanzdienstleistungen</b>
Zeitraum	10/08 – 03/09
Projektziel	Automatisierter Herstellertest für eine Web-Anwendung
Aufgaben / Verantwortlichkeiten	<ul style="list-style-type: none"> <li>• Neuentwicklung von automatisierten Herstellertests mit Java SE 5 für eine J2EE-Web-Anwendung (Plattform: IBM WebSphere, Rapid Application Developer (RAD)). Die Anwendung wird von Banken genutzt und dient der Bonitätsbewertung von Finanzierungen (Rating).</li> <li>• Zur Prüfung der Funktionalität wird das Rahmenwerk HttpUnit verwendet. Es werden Dialogabläufe sowie Feldinhalte und -zustände überprüft. Die Testdaten werden für den Herstellertest in XML angelegt.</li> <li>• Außerdem findet noch eine weitere Form des Tests statt, in dem die Ergebnisse von Berechnungen der Anwendung mit vom Kunden gelieferten Testdaten abgeglichen werden. Hierbei werden die Testdaten über eine Microsoft Access-Datenbank eingelesen und automatisch gegen die Rechenergebnisse der GUI geprüft.</li> </ul>
Methoden / Technologien	Java SE 5, HttpUnit, JUnit, DBUnit, JavaDoc, Mercury Quality Center, Log4J, Apache Commons, Microsoft Access, Eclipse, Oxygen, CVS, Ant, Wiki

<b>Branche / Bereich</b>	<b>Logistik</b>
Zeitraum	02/08 – 08/08
Projektziel	Architektur & Integration für eine Logistikanwendung
Aufgaben / Verantwortlichkeiten	<ul style="list-style-type: none"> <li>• Entwicklung einer Komponente zur Messung der Performance einer performancekritischen Logistikanwendung für Containerhäfen. Die Anwendung ist stark heterogen und verteilt. Die Kommunikation erfolgt über JMS (Java Message Service).</li> <li>• Zur Überwachung der Performance werden die JMS-Nachrichten zusammen mit ihren Sendezeiten mittels JPA (Java Persistence API) und Hibernate in einer Oracle-Datenbank gespeichert. Die primär eingesetzten Programmiersprachen sind Java 6 und C#. Das Projekt wird mit Maven 2 gebaut, geringfügig werden Ant-Skripte eingesetzt.</li> <li>• Das Projekt ist in Komponenten (Bundles, Services) im Sinne einer OSGi-Anwendung aufgeteilt. Die Komponenten werden mit Spring konfiguriert.</li> <li>• Die Builds werden mit CruiseControl gesteuert. Programmteile werden aus mit MagicDraw 15 erstellten UML-Modellen in Verbindung mit OpenArchitectureWare 4 generiert. Die OSGi-Komponenten werden mit Spring konfiguriert.</li> </ul>
Methoden / Technologien	Maven 2, Java SE 6, MagicDraw 15, OpenArchitectureWare 4, Spring, Oracle, Hibernate, JPA Java Persistence API, JMS Java Message Service, Subversion, Junit, Log4J, Apache Commons, IntelliJ Idea 7, Eclipse 3.4, CruiseControl, Ant, Jira, Test Track Pro