Universität Zürich
Institut für Informatik

s.e.a.l. software evolution & architecture lab
RERG

Software Engineering Übung 1

Programmverstehen, Dokumentation

1 Allgemeines

1.1 Wichtige Daten

- Ausgabe Di 15.09.2009
- Abgabe So 27.09.2009 bis 23:55 Uhr
- Besprechung am Di 6.10.2009, 12:15 Uhr

1.2 Formales

Die Lösungen sollen als PDF Datei mit dem Namen **Ex[*n*]_[*NameA_NameB_NameC*].pdf** abgegeben werden, wobei [*n*] die Nummer der Übung ist und [*NameA_NameB_NameC*] die Nachnamen der Gruppenmitglieder sind.

Mailen sie Ihre Lösungen vor dem Abgabetermin an `wueest@ifi.uzh.ch`. Der Betreff der E-mail sollte mit **[SE EX HS09]** beginnen. Mailen Sie bitte ein Archiv (.zip-File), welches Ihre ergänzte Klasse `report.visitor.HTMLExporter` (nur die *.java* Datei) und Ihr PDF enthält. Benennen sie das Archiv anhand der oben erwähnten Konventionen.

Denken Sie daran, dass verspätete Abgaben zwar korrigiert, die erreichten Punkte aber nicht angerechnet werden. Die Übungen sollen in 3er Gruppen gelöst werden. Jedes Gruppenmitglied muss über alle Teile der Lösungen Auskunft geben können.

1.3 ReportingSystem Installation

Diese Übung basiert auf einem existierenden Java Programm, *ReportingSystem*. *ReportingSystem* ist ein Teil eines grösseren Systems. Es ist verantwortlich für das Speichern von Berichten in verschiedenen Formaten, wie z.B. RTF (Rich Text Format) oder LaTeX.

ReportingSystem kann als gepacktes Archiv von der Webseite der Übungen heruntergeladen werden. Das Archiv beinhaltet:

- den Source Code von *ReportingSystem* (im `src/main` Ordner).
- einige Testfälle für *ReportingSystem* (im `src/test` Ordner). Der Source Code umfasst *JUnit* Testfälle und einen Bericht im RTF, HTML sowie LaTeX Format.
- alle benötigten Dateien, so dass das Archiv als Projekt in Eclipse importiert werden kann.

Gehen Sie in Eclipse ins *File* Menü und klicken Sie auf *Import...*. Wählen Sie dann *Existing project into the workspace* und geben Sie bei *Select archive file* das *ReportingSystem* Archiv an. Klicken Sie auf *Finish*. Eclipse erstellt im Workspace ein neues Projekt (namens *ReportingSystem* wie in der Abbildung 1).

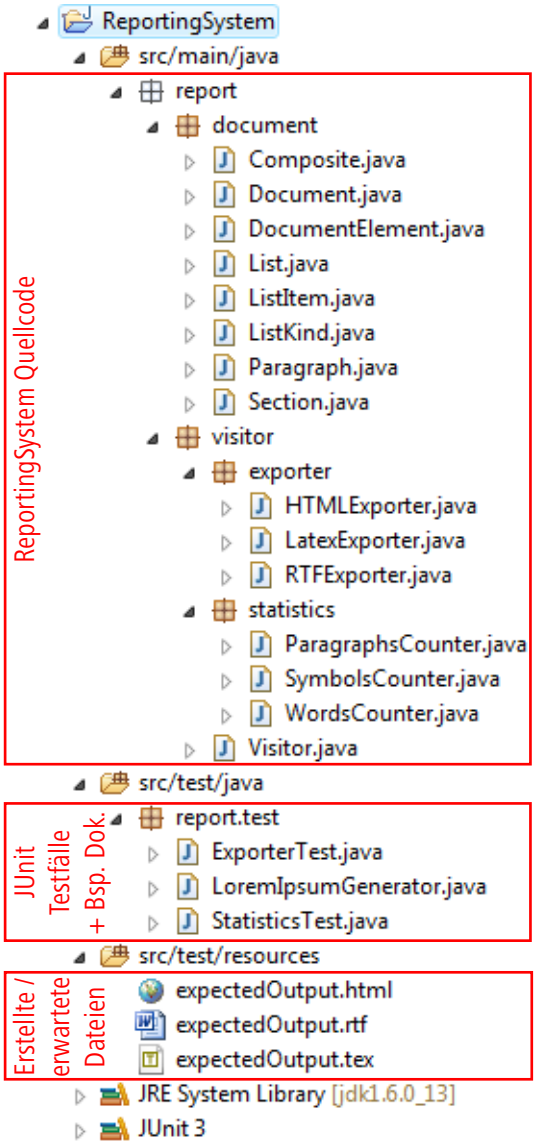

Abbildung 1: ReportingSystem Projekt in Eclipse

2 Verstehen des Quellcodes (12 Punkte)

2.1 Struktur von Dokumenten (5 Punkte)

Betrachten Sie das Package *report.document*. Dieses Package enthält Klassen, die benützt werden, um ein Dokument abzubilden. Die Klasse `report.test.LoremIpsumGenerator` illustriert, wie ein Dokument im *ReportingSystem* gebaut wird.

Zeichnen Sie ein UML Klassendiagramm, welches folgendes zeigt:

- alle zum Package *report.document* gehörenden Java Klassen
- ihre Attribute und Methoden
- die Beziehungen zwischen ihnen (Vererbung und Assoziation mit Kardinalität)

Schreiben Sie einen kurzen Text (maximal 5 Sätze), welcher den Inhalt des Diagramms zusammenfasst.

2.2 Speichern eines Berichts (5 Punkte)

Die Klasse *report.visitor.exporter.LatexExporter* ist dafür verantwortlich, ein Dokument als LaTeX Datei zu speichern. Diese Klasse implementiert das sogenannte *Visitor* Entwurfsmuster[1]. Es dient zum Kapseln von Operationen, die auf Elementen einer Objektstruktur (hier: das Dokument) ausgeführt werden. Daher bringt es die Möglichkeit, neue Handlungen über Dokumenten zu definieren, wie z.B. die Unterstützung eines neuen Dateiformates für das Speichern von Berichten.

Benutzen Sie ein UML Sequenzdiagramm, um die Kollaboration zu beschreiben, welche durch den Aufruf der Methode *testLatex()* der Klasse `report.test.ExporterTest` initialisiert wird. Sie müssen nur die Interaktionen zwischen den Objekten der folgenden Klassen darstellen (und können die restlichen Interaktionen ignorieren):

- `report.test.ExporterTest`
- `report.visitor.exporter.LatexExporter`
- `report.document.Dokument` und
- `java.io.PrintWriter`

Schreiben Sie einen kurzen Text (maximal 5 Sätze), welcher den Inhalt des Diagramms zusammenfasst.

2.3 Statistiken über einen Bericht (2 Punkte)

Im Package `report.visitor.statistics` sind 3 Visitoren, welche die Wörter, die Zeichen und die Abschnitte in einem Dokument zählen.

Erklären Sie die Begriffe[2] *Wort* und *Zeichen*, die benützt werden, um diese Klassen zu implementieren (1 Punkt).

Um diese Klassen zu testen,[3] werden die Ergebnisse des *ReportingSystem* aus dem Dokument, welches in der Klasse `report.test.LoremIpsumGenerator` generiert wird, mit den Ergebnissen der Datei *expectedOutput.rtf* von Microsoft Word verglichen. Rechts-klicken Sie auf die Klasse `report.test.StatisticsTests`, und klicken Sie auf *Run as* und *JUnit Test*. Die Abbildung 2 zeigt, dass Microsoft Word 35 und *ReportingSystem* nur 8 Abschnitte gezählt hat. Warum? (1 Punkt)

[1] Entwurfsmuster werden später in der Vorlesung (Kapitel 5) und in der Übung 4 behandelt.

[2] In der Vorlesung (Kapitel 4) und in der nächsten Übung werden Sie entdecken, warum es wichtig ist, ein Glossar anzulegen.

[3] Testen wird später in der Vorlesung (Kapitel 8) und in der Übung 6 behandelt.

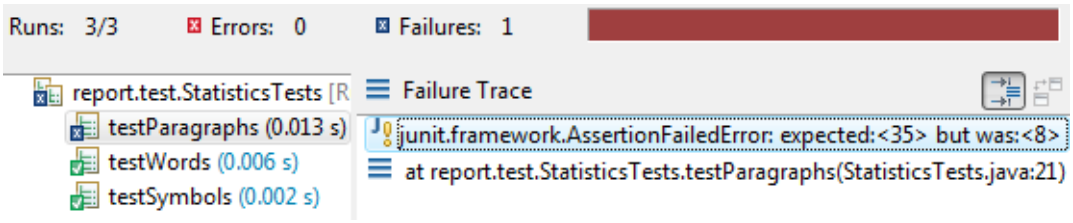

Abbildung 2: Ergebnisse von `report.test.StatisticsTests`

3 Verbesserung und Erweiterung des Quellcodes (8 Punkte)

3.1 Unterstützung des HTMLFormats (5 Punkte)

Ergänzen Sie die Klasse `report.visitor.HTMLExporter`, so dass der Test `report.test.ExporterTest` ohne Fehler ausgeführt werden kann. Wenn der Test ausgeführt wird, wird eine Datei *actualOutput.html* im Ordner `src/test/resources` generiert. Falls Sie diese Datei nicht sehen, rechts-klicken Sie auf dem Ordner `src/test/resources` und klicken Sie auf *Refresh*. Die Datei *expectedOutput.html* ist das genaue Ergebnis, das ihre Implementierung ausgeben soll.

3.2 Dokumentation (3 Punkte)

Dokumentieren Sie ihre Klasse `report.visitor.HTMLExporter`. Schreiben Sie Klassen-, Feld- und Methodenkommentare. Folgen Sie dabei den Richtlinien des Java Style Guide (erhältlich auf der Webseite der Übungen). Ihre Antworten zu den vorherigen Fragen werden für diese Aufgabe nützlich sein.