Design Pattern SS 2013
Hausaufgabe 2

UNIKASSEL VERSITÄT

Die Aufgaben müssen einzeln bearbeitet und abgegeben werden. Die Abgabe muss **bis spätestens Montag 29.04.2013 um 23:59 Uhr** über unser Hausaufgabenabgabesystem **http://seblog.cs.uni-kassel.de/dpss13/** erfolgen. Die Abgabe ist nur als einzelne *.zip oder *.jar-Datei möglich. Daher müssen alle für eine Abgabe relevanten Daten zu einem solchen Archiv kombiniert werden.

Hinweise zur Abgabe:

- Die Hausaufgabe als exportiertes Eclipse Projekt (*.zip, nicht den gesamten Workspace) abgeben. Das kann mit Hilfe der Eclipse Export Funktion durchgeführt werden. Ist das Projekt nicht korrekt exportiert, kann es bei der Korrektur nicht berücksichtigt werden (es bietet sich also an, den Import des exportierten Projektes auszuprobieren).

  WICHTIG Benennen Sie ihr Projekt für diese Abgabe nach folgendem Schema:

  DPSS13_HA<a>_<Matrikelnummer>,

  wobei <a> für die aktuelle Hausaufgabe steht.
  Beispiel:
  DPSS13_HA10_12345678.

Allgemeines
Orientieren Sie sich für die Lösung der Aufgaben an den zugehörigen Übungen und Vorlesungen:
http://seblog.cs.uni-kassel.de/category/currentterm/design-patterns/

Die Benotung beruht auf den Hausaufgaben.

Hierfür werden die gesamten Hausaufgaben minus zwei vom Studenten abgegebenen Hausaufgaben addiert, die mit mehr als 50 % bewertet wurden. Nicht abgegebene Hausaufgaben oder Betrugsversuche bekommen 0 Punkte.

Die Prüfung gilt als nicht bestanden, wenn alle Hausaufgabenpunkte weniger als **50 %** ergeben oder der Student mehr als **zwei Hausaufgaben nicht abgibt**.

Fragen bitte an Stefan.Lindel@cs.uni-kassel.de oder george@uni-kassel.de

Aufgabe 1

- Implementiert ein Datenmodell für ein Buchinventarisierungssystem mit dem Composite Pattern
- Hierbei sollen die einzelnen Buchteile ermittelt werden
- Ein Book besteht aus: Parts, Chapters, Sections, Paragraphs, Sentences
- Schreibt einen Junit-Test
    - in welcher eine Klasse Book existiert
- die Methode sumBook(Book book) : Integer, welche die ermittelte Anzahl der Sätze in Textpassagen (Sentences) summiert und zurückgibt.

Fragen bitte an Stefan.Lindel@cs.uni-kassel.de oder george@uni-kassel.de

Contents

Chapter 1

Introduction

If you would ask us. whether this is just another book about modeling, we would probably feel inclined to say: yes and no. Yes. it is a lot about modeling, but no. it is also about programming, methodological software design. and rapid prototyping via methods from model driven engineering. It will also be one of the first complete references and teaching guides for Story Driven Modeling. Story Driven Modeling is an agile software development method using objects and scenarios and special modeling steps to facilitate system analysis and design. [34, 2 l, 20]. Most parts of this book can be done with pencil and paper and with standard UML tools and standard software development environments. However. some steps are best supported by the rapid prototyping tool Fujaba [5] or the Story Driven Modeling library SDMLib [I0].

The title of this book does not include Object Oriented Modeling on purpose. Object Orientation. Object Oriented Design. Object Oriented Analysis. and other object oriented methods all refer somehow to class diagrams and inheritance. Instead of this. we will actively use objects for modeling. analysis. and design. We will learn to think in objects.

This book is foremost planned to be a textbook for software modeling courses. It olfers a very interactive and agile approach to modern software design. ln this book. we introduce the Objects First principle which is the foundation of the Story Driven Modeling development method. This is not to be mistaken with an object oriented development method. You will see that the Object First method slightly differs from traditional object oriented methods. With this book. we address a majority of readers dealing with or wanting to learn sofiware development. This includes teachers and students for introduction to Object Orientation. Systems Modeling, Object Oriented Design, or Model Driven Engineering. This book should also be insightful for people interested in modeling and program design and beginning programmers. We expect from

1.1 Why Model?

Think for a minute, what was tl1e biggest software project you have been involved in‘! l-low many lines of code did it have‘! l-low long have people been working on it? If things took long, why did they take this much time?‘

Usually. people thinking about these questions will recall some communication problems in the process of developing a specific program. These communication problems occur talking to the customer and between the programmers and managers. Also, it should be observed that these problems are growing the bigger the project is. As the title of this book includes “Modeling”, modeling might be an answer to some of the problems we just discovered.

However, why should we model, when we are all great programmers? This is also a good question to ask to students to get them thinking. I-low would you answer this question?

> “If you were supposed to understand it. we wouldn’t call it code.”
> - from a Federal Express promotion. reported by IS Survivalist Matthew 0. Persico

Writing code already means that we create an artifact. which is not easy to understand, which is encrypted in a way that makes it less accessible for human understanding. It is obvious that showing the problem in a particular context, omitting some of the machine specific details. will help to understand this view better and therefore foster communication. Models can show connections, relations, and context at once. This can help us to see and analyze risks and costs.

Many definitions of the temi “model” (in the context of software) are related to a system. Such as given by Bezivin et al. in

- [I8]: “A model is a simplification of a system built with an intended goal in mind. ‘Hie model should be able to answer questions in place of the actual system.” or given by Seidewitz in
- [33]: ..A model is a set of statements about some system under study [SUS]. Here, statement means expression about the SUS.“

A definition focusing on aspects of a reality rather than systems is provided by Pohl in

- [3l]: “A model is an abstracting image of an existing or fictitious reality.”

Aufgabe 2

- stellt das Composite Pattern für Book vom letzten Mal auf ein Visitor Pattern um
- Baut einen Visitor zur Summierung der Konsonanten aller Textbausteine:
  „B, C, D, F, G, H, J, K, L, M, N, P, Q, R, S, ß, T, V, W, X, Z“
- Baut einen Visitor der alle Texte zählt, auf denen ein mitgelieferter regulärer Ausdruck matcht

Fragen bitte an Stefan.Lindel@cs.uni-kassel.de oder george@uni-kassel.de