



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**.



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.

Contents

I	Object Models	1
1	Introduction	3
1.1	Why Model?	6
1.2	Guiding Example: Study-Right University	10
1.3	From Concrete to Abstract to Objects	12
1.4	The High Art of Giving Examples	14
1.5	Scenarios and Stories	19
1.6	Exercises	21
1.6.1	Abstract vs. Concrete	21
1.6.1.1	Terms and Definitions	21
1.6.1.2	Chess	21
1.6.2	Giving Examples	22
1.6.2.1	Mau Mau	22
1.6.2.2	Battleships	23
1.6.2.3	Mancala	24
1.6.2.4	Mensch Ärgere Dich Nicht	26
1.6.2.5	Towers of Hanoi	27
1.6.2.6	ATM Money Withdrawal	27
1.6.2.7	Borrowing Electronic Books from a Library	28
1.6.2.8	Trouble Ticket System	28
1.6.2.9	Webshop	28
1.6.3	User Stories / Scenarios	29
1.6.3.1	Mau Mau	29
1.6.3.2	Battleships	29
1.6.3.3	Mancala	29
1.6.3.4	Mensch Ärgere Dich Nicht	30
1.6.3.5	Towers of Hanoi	30
1.6.3.6	ATM Money Withdrawal	30
1.6.3.7	Borrowing Electronic Books from Library	30
1.6.3.8	Trouble Ticket System	30

1.6.3.9	Webshop	30
2	Object Diagrams	31
2.1	Exercises	39
2.1.1	Mau Mau	39
2.1.2	Battleship	39
2.1.3	Mancala	40
2.1.4	Mensch Ärgere Dich Nicht	40
2.1.5	Towers of Hanoi	40
2.1.6	ATM Money Withdrawal	40
2.1.7	Borrowing Electronic Books from Library	40
2.1.8	Trouble Ticket System	40
2.1.9	Webshop	41
3	From Objects to Classes	43
3.1	Introduction	43
3.2	Deriving Class Diagrams from Object Diagrams	44
3.3	About Cardinalities	50
3.4	Correct Use of Inheritance	53
3.5	Exercises	54
3.5.1	Mau Mau	54
3.5.2	Battleships	54
3.5.3	Mancala	54
3.5.4	Mensch Ärgere Dich Nicht	54
3.5.5	Towers of Hanoi	55
3.5.6	ATM Money Withdrawal	55
3.5.7	Borrowing Electronic Books from Library	55
3.5.8	Trouble Ticket System	55
3.5.9	Webshop	55
3.5.10	Matching	55
3.5.11	Deriving Object- and Class Diagrams	57
3.5.12	Connect Four Game	57
4	Translating Diagrams to Java Code	59
4.1	Class Diagrams to Java Code	60
4.1.1	Bidirectional Associations: Managing Referential Integrity	64
4.1.2	Choosing the Right Container Class for To-Many Associations	67
4.1.3	Generated Code	68
4.2	Object Diagrams to Java Code	71
4.3	Generic Implementations of Object Diagrams	75
4.4	eDobs as Runtime Model Explorer	78
4.5	Exercises	80

4.5.1	Study-Right University	80
4.5.2	Mau Mau	81
4.5.3	Battleships	82
4.5.4	Mancala	84
4.5.5	Mensch Ärgere Dich Nicht	84
4.5.6	Towers of Hanoi	85
4.5.7	ATM Money Withdrawal	85
4.5.8	Borrowing Electronic Books from Library	86
4.5.9	Trouble Ticket System	86
4.5.10	Webshop	87
4.5.11	Connect Four Game	87
4.5.12	Derive Classes from Code	88
4.5.13	Referential Integrity Templates	88

II Behavior 91

5 Storyboards 93

5.1	Exercises	102
5.1.1	Mau Mau	102
5.1.2	Battleships	103
5.1.3	Mancala	103
5.1.4	Mensch Ärgere Dich Nicht	104
5.1.5	Towers of Hanoi	104
5.1.6	ATM Money Withdrawal	104
5.1.7	Borrowing Electronic Books from Library	105
5.1.8	Trouble Ticket System	105
5.1.9	Webshop	106
5.1.10	Connect Four Game	106

6 Algorithm Design 109

6.1	Model Exploration	110
6.2	Object Games	119
6.3	Exercises	124
6.3.1	Mau Mau	124
6.3.2	Battleships	124
6.3.3	Mancala	124
6.3.4	Mensch Ärgere Dich Nicht	124
6.3.5	Towers of Hanoi	125
6.3.6	ATM Money Withdrawal	125
6.3.7	Borrowing Electronic Books from Library	125
6.3.8	Trouble Ticket System	126

6.3.9	Webshop	126
6.3.10	Connect Four Game	126
7	Simple Java with Object Models	127
7.1	Exercises	140
7.1.1	Mau Mau	140
7.1.2	Battleships	140
7.1.3	Mancala	140
7.1.4	Mensch Ärgere Dich Nicht	141
7.1.5	Towers of Hanoi	141
7.1.6	ATM Money Withdrawal	141
7.1.7	Borrowing Electronic Books from Library	141
7.1.8	Trouble Ticket System	142
7.1.9	Webshop	142
7.1.10	Connect Four Game	142
8	Testing	143
8.1	Testing in Java with JUnit	145
8.2	Creating our First Test	146
8.3	JUnit Assertions	149
8.4	Testing for Errors	151
8.5	Test Coverage	153
8.6	Exercises	154
8.6.1	Mau Mau	154
8.6.2	Battleships	154
8.6.3	Mancala	155
8.6.4	Mensch Ärgere Dich Nicht	155
8.6.5	Towers of Hanoi	156
8.6.6	ATM Money Withdrawal	156
8.6.7	Borrowing Electronic Books from Library	156
8.6.8	Trouble Ticket System	157
8.6.9	Webshop	157
8.6.10	Connect Four Game	158
9	Story Diagram Idioms	159
9.1	Story Diagram Control Flow Idioms	159
9.1.1	Empty Story Diagram	159
9.1.2	Java Statements in Story Diagrams	160
9.1.3	Sequences of activities	161
9.1.4	If-Then-Else Control Flow	161
9.1.5	Loop Control Flow	164
9.1.6	Return Values	167
9.1.7	Recursion	168

9.1.8	Exception Handling	168
9.2	Story Pattern Elements	170
9.2.1	Pattern Variables and Pattern Objects	170
9.2.2	Matching Objects and Links	172
9.2.3	Attribute Conditions	173
9.2.4	Java Statements in Story Patterns, a Complex Example for the FAMous Fujaba Abstract Machine	174
9.2.5	Attribute Assignments, Object and Link Creation and Deletion	179
9.2.6	Using Marker Nodes as Collections	183
9.2.7	Handling Multiple Matches: For-Each Activities	186
9.2.8	Some Special Pattern Elements: Type Casts	189
9.2.9	Some Special Pattern Elements: Paths	192
9.2.10	Handling Multiple Matches: Set Valued Pattern Objects	195
9.2.11	Important Pattern Elements We Did Not Yet Use: Op- tional Objects, Negative Objects, Complex Constructors	196
9.3	Exercises	199
9.3.1	Mau Mau	199
9.3.2	Battleships	199
9.3.3	Mancala	200
9.3.4	Mensch Ärgere Dich Nicht	200
9.3.5	Towers of Hanoi	200
9.3.6	ATM Money Withdrawal	200
9.3.7	Borrowing Electronic Books from Library	201
9.3.8	Trouble Ticket System	201
9.3.9	Webshop	201
9.3.10	Connect Four Game	201
10	Simple Story Driven Modeling	203
10.1	The Study-Right University doAssignments	203
10.2	The Study-Right University findPath	212
10.3	Exercises	213
11	SDM of GUI Applications	219
11.1	Storyboard and Behavior Modeling	235
11.2	Using a GUI Construction Toolkit	243
11.3	Testing Graphical User Interfaces	251
11.4	Exercises	254
11.4.1	Mau Mau	254
11.4.2	Battleships	255
11.4.3	Mancala	255
11.4.4	Mensch Ärgere Dich Nicht	255

11.4.5 Towers of Hanoi	255
11.4.6 ATM Money Withdrawal	255
11.4.7 Borrowing Electronic Books from Library	255
11.4.8 Trouble Ticket System	255
11.4.9 Webshop	255
11.4.10 Connect Four Game	255

III Scaling Up 257

12 SDM for concurrent applications 259

12.1 Multiple Threads to Keep Your GUI Responsive	261
12.2 A Basic Chat Program	271
12.3 Exercise	281
12.3.1 Collaborative Whiteboard	281
12.3.2 Client Server Chat	283
12.3.3 Multi Player Versions of Our Exercise Applications	284

13 Story Driven Modeling 285

13.1 Iterative Refinement	285
13.2 SDM with Graphical User Interfaces	287
13.3 SDM with Customers	289
13.3.1 Requirements	293
13.3.2 Writing business software for customers	298
13.4 Exercises	300
13.4.1 Kanban Board	300
13.4.2 Assignment Management System	300
13.4.3 Key Administrative App	300

14 Yes, It Scales 303

14.1 More Objects	304
14.2 More Classes, More Algorithms	306
14.3 Design Pattern	309
14.4 More Installations	310
14.5 Exercises	312

15 SDM of Embedded Applications 315

15.1 What is an Embedded System?	316
15.2 Example: LEGO Mindstorms NXT Robotics System	319
15.3 Example: Android Platform (for Robotics)	320
15.4 Developing for Arduino	321
15.5 Exercises	322

<i>CONTENTS</i>	13
Teaching Guidelines	323



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, 21, 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 [10].

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 offers a very interactive and agile approach to modern software design. In 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 software 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 the biggest software project you have been involved in? How many lines of code did it have? How 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. How 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 O. 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 term “model” (in the context of software) are related to a system. Such as given by Bezivin et al. in

- [18]: “A model is a simplification of a system built with an intended goal in mind. The 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

- [31]: “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