**User Interfaces 1, HS 2010**

05.12.2010

**BookMasterPro**

**Entwicklungs- und Benutzerdokumentation**

**Christina Heidt & Lukas Elmer**

# Inhalt

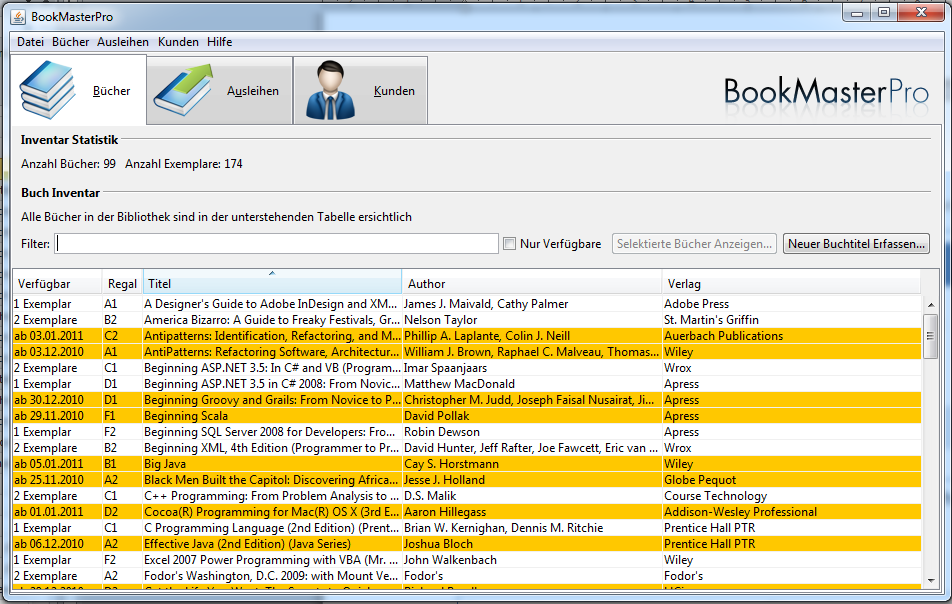
[1. Inhalt 1](#_Toc278219190)

[2. Todos 2](#_Toc278219191)

# Szenarien

Da die Szenarios das Zentrale am UI-Projekt sind, haben wir den Ablauf der einzelnen Szenarios kurz beschrieben und mit Bildern illustriert.

Der Augangspunkt für alle Szenarien ist der „BookMaster“ mit dem ausgewählten Tab „Bücher“. Der Cursor befindet sich im „Filter:“ Feld:



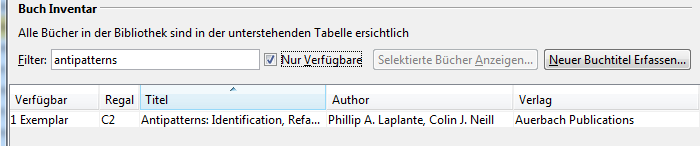
## «Muss» Funktionalität

### Szenario 1.1 (Buchverfügbarkeit prüfen & Buchstandort finden)

1. Eingabe des Buchtitels im Feld „Filter:“ (Bsp: „antipatterns“)
2. Buchtitel werden in der Tabelle angezeigt. Im Feld „Regal“ kann der Buchstandort herausgelesen werden, dem Feld „Verfügbar“ entnommen werden, ob sich zurzeit Exemplare in der Bibliothek befinden oder wann wieder Exemplare in der Bibliothek sein sollten

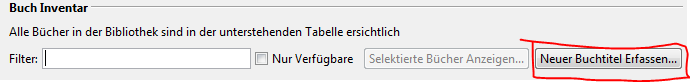


1. Sollen nur die verfügbaren Bücher angezeigt werden, so können diese durch die Check Box „Nur Verfügbare“ gefiltert werden:

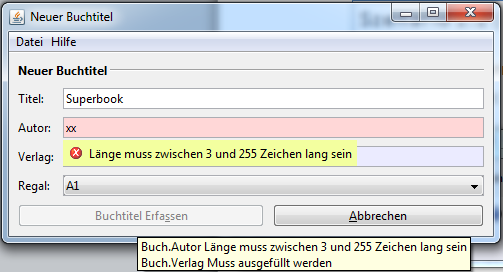


### Szenario 1.2 (Buchtitel hinzufügen)

1. Um einen neuen Buchtitel zu erfassen muss auf den Button „Neuer Buchtitel Erfassen…“ geklickt werden:



1. Ein neues Fenster „Neuer Buchtitel“ erscheint
2. Formular muss ausgefüllt werden (automatische Validierung, falls Eingabe fehlerhaft)
   1. Anzeige der Fehlermeldung direkt unter dem Textfeld
   2. Anzeige der gesamten Fehlermeldung als ToolTip beim Button „Buchtitel erfassen“

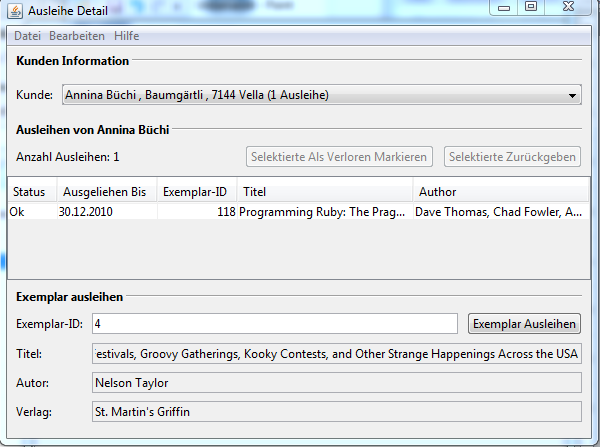


1. Sobald alle Felder richtig ausgefüllt wurden, wird der Button „Buchtitel erfassen“ aktiviert und der neue Buchtitel kann gespeichert werden.

### Szenario 2.1 (Buch ausleihen)

Da der Kunde in der Bibliothek ein Exemplar auswählt, das mit einer Exemplar-ID angeschrieben ist, ist die Exemplar-ID bekannt. Wenn der Kunde noch nicht vorhanden ist, so muss er zuerst erstellt werden (siehe „Master-Detail View für Kundenverwaltung“). Dieses Szenario geht davon aus, dass der Kunde bereits erstellt wurde.

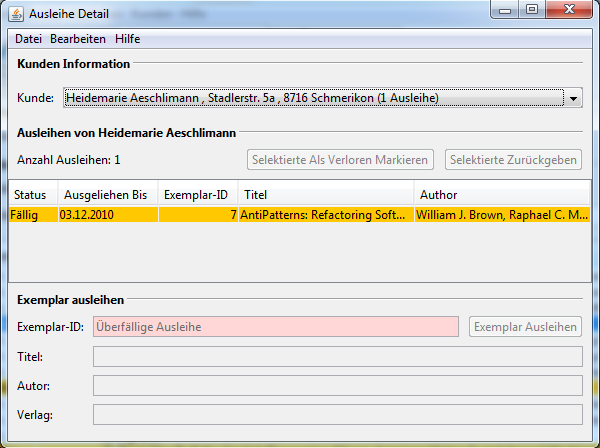
1. Wechseln zum Tab „Kunden“
2. Kunde Filtern (z.B. nach „büchi“)
3. Kunde „Büchi Annina“ auswählen
4. Auf Button „Ausleihe Für Kunden Erfassen…“ klicken
5. Neues Fenster „Ausleihe Detail“ wird geöffnet
6. Pendente Ausleihen Kontrollieren, falls vorhanden 🡪 im Fehlerfall am Kunden Feedback geben (Benutzer wird durch GUI unterstützt)
   1. ob Kunde weniger als 3 Ausleihen hat
   2. ob Kunde keine überfälligen Ausleihen hat
7. Exemplar-ID in Text Feld eingeben
8. Unter der Exemplar-ID werden weiter Buchinformationen angezeigt
   1. Kontrollieren, ob richtige Exemplar-ID eingegeben wurde
9. Auf Button „Exemplar Ausleihen“ klicken, um das Buch auszuleihen



### Szenario 2.2 (Bücher ausleihen mit überfälligem Buch)

Ähnliches Szenario wie 2.1:

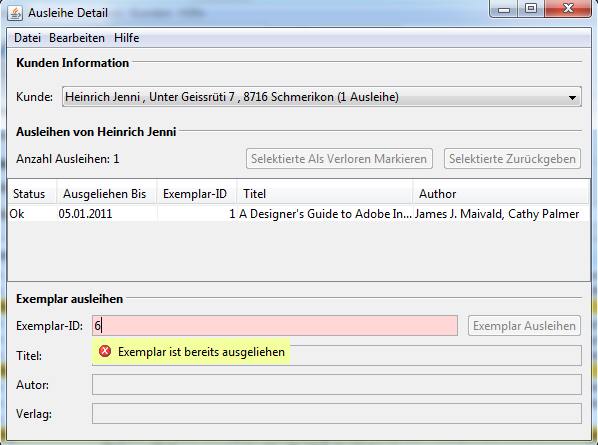
1. Wechseln zum Tab „Kunden“
2. Kunde Filtern (z.B. nach „büchi“)
3. Kunde „Büchi Annina“ auswählen
4. Auf Button „Ausleihe Für Kunden Erfassen…“ klicken
5. Neues Fenster „Ausleihe Detail“ wird geöffnet
6. Pendente Ausleihen Kontrollieren, falls vorhanden 🡪 im Fehlerfall am Kunden Feedback geben (Benutzer wird durch GUI unterstützt)
7. Kunde hat überfällige Ausleihe



### Szenario 2.3 (Buch ausleihen welches schon ausgeliehen ist)

Ähnliches Szenario wie 2.1:

1. Wechseln zum Tab „Kunden“
2. Kunde Filtern (z.B. nach „büchi“)
3. Kunde „Büchi Annina“ auswählen
4. Auf Button „Ausleihe Für Kunden Erfassen…“ klicken
5. Neues Fenster „Ausleihe Detail“ wird geöffnet
6. Pendente Ausleihen Kontrollieren, falls vorhanden 🡪 im Fehlerfall am Kunden Feedback geben (Benutzer wird durch GUI unterstützt)
   1. ob Kunde weniger als 3 Ausleihen hat
   2. ob Kunde keine überfälligen Ausleihen hat
7. Exemplar-ID in Text Feld eingeben
8. Meldung „Exemplar ist bereits ausgeliehen“ wird angezeigt
9. Button „Exemplar Ausleihen“ ist deaktiviert



### Szenario 2.4 (Mehrere Bücher ausleihen)

Ähnliches Szenario wie 2.1

1. Wechseln zum Tab „Kunden“
2. Kunde Filtern (z.B. nach „büchi“)
3. Kunde „Büchi Annina“ auswählen
4. Auf Button „Ausleihe Für Kunden Erfassen…“ klicken
5. Neues Fenster „Ausleihe Detail“ wird geöffnet
6. Pendente Ausleihen Kontrollieren, falls vorhanden 🡪 im Fehlerfall am Kunden Feedback geben (Benutzer wird durch GUI unterstützt)
   1. ob Kunde weniger als 3 Ausleihen hat
   2. ob Kunde keine überfälligen Ausleihen hat
7. Exemplar-ID in Text Feld eingeben
8. Unter der Exemplar-ID werden weiter Buchinformationen angezeigt
   1. Kontrollieren, ob richtige Exemplar-ID eingegeben wurde
9. Auf Button „Exemplar Ausleihen“ klicken, um das Buch auszuleihen
10. Falls weitere Exemplare an den Kunden ausgeliehen werden sollen, weiter bei 6

### Szenario 3.1 (Buch zurückgeben)

### Szenario 3.2 (Buch zurückgeben welches überfällig ist)

## «Kann» Funktionalität

### Szenario 3.3 (Mehrere Bücher zurückgeben)

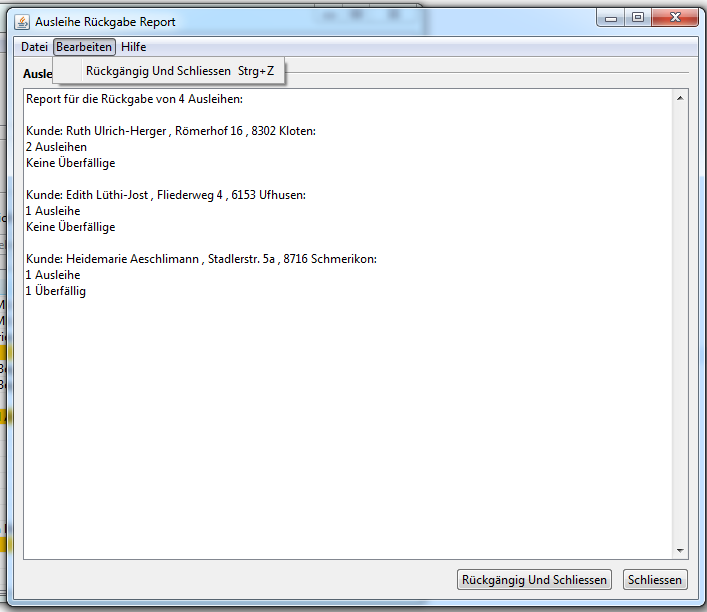
### Szenario 3.4 (Buch zurückgeben welches nicht der Ausleihe entspricht)

## «Optionale» Funktionalität

### Commands & Undo/Redo

Für gewisse Szenarien ist es sehr schwierig, ein gutes Undo/Redo zu implementieren, das später auch wirklich gebraucht wird. Da der Aufwand dafür sehr gross sein kann, haben wir nur an einer Stelle im Miniprojekt ein Undo implementiert.

Da die Wahrscheinlichkeit für einen Missgriff beim Ausleihe zurückgeben eher gross ist, haben wir dort ein Undo implementiert:



### Help

Um zu demonstrieren, wie die Hilfe aufgerufen werden kann, haben wir eine Action implementier (F1) und ein Menu Eintrag hinzugefügt. Es wurde aber keine Benutzerhilfe geschrieben, deshalb wird, statt dass die Hilfe angezeigt wird, ein Popup angezeigt.

### CellRenderer

Für den “Book Master” hatten wir zuerst gedacht, dass wir einen CellRenderer brauchen, um bestimmte Zeilen einzufärben. Allerdings haben wir dafür einen anderen Weg gefunden, ohne einen CellRenderer zu implementieren:



Bei dieser Lösung hat man den Vorteil, dass nicht für jede einzelne Spalte ein CellRenderer hinzugefügt werden muss. Da in unseren Tabelle sich eine ganze Zeile immer auf ein bestimmtes Objekt bezieht (z.B. auf ein Buch), haben wir diese Lösung bevorzugt.

Um zu zeigen, dass wir auch den CellRenderer beherrschen, haben wir eine Demoversion dafür gebaut, die in der Klasse „LoanDetailWithCellRenderer“ gefunden werden kann:



In der Klasse „BookMasterUiManager“ kann dieses Fenster aktiviert werden, indem man die folgenden Zeilen ändert:





### CellEditor

#### Cell Editor im „BookMaster“

Um zu zeigen, dass wir den CellRenderer und den CellEditor beherrschen, haben wir als Demo einen CellEditor im BookMaster implementiert:



Wir empfehlen aber explizit, dass dieser CellEditor wieder entfernt wird. Denn es soll vermieden werden, dass ein Benutzer aus Versehen einen Fehler macht und ein Buch bearbeitet (Missgriff), obwohl er es nicht will.

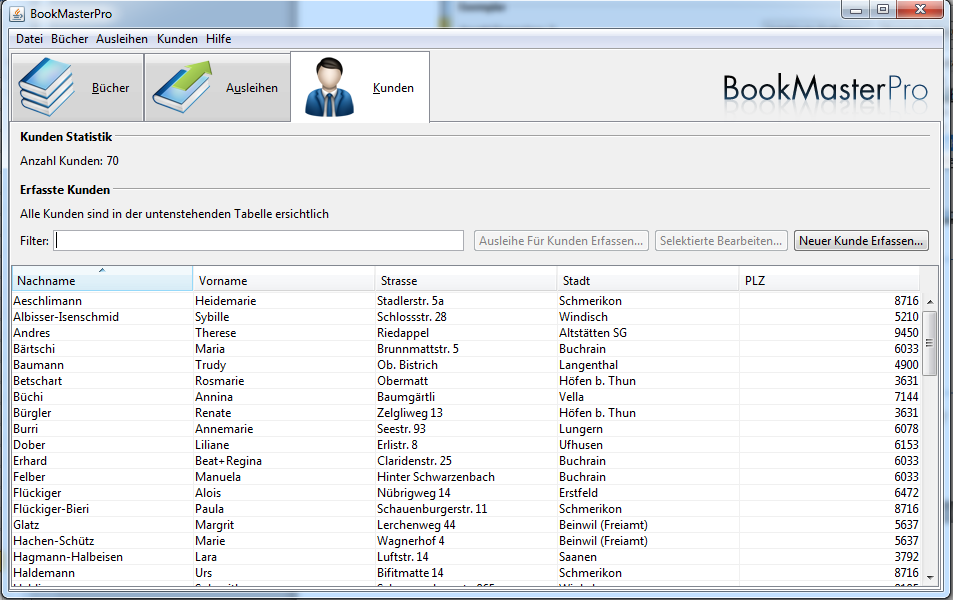
#### Cell Editor im „Buch Detail“, um Exemplar zu bearbeiten

Da das Exemplar nur ein Feld hat, das bearbeitet werden kann, haben wir uns entschieden, dafür einen CellEditor zu verwenden, damit das Exemplar direkt in der Tabelle bearbeitet werden kann:

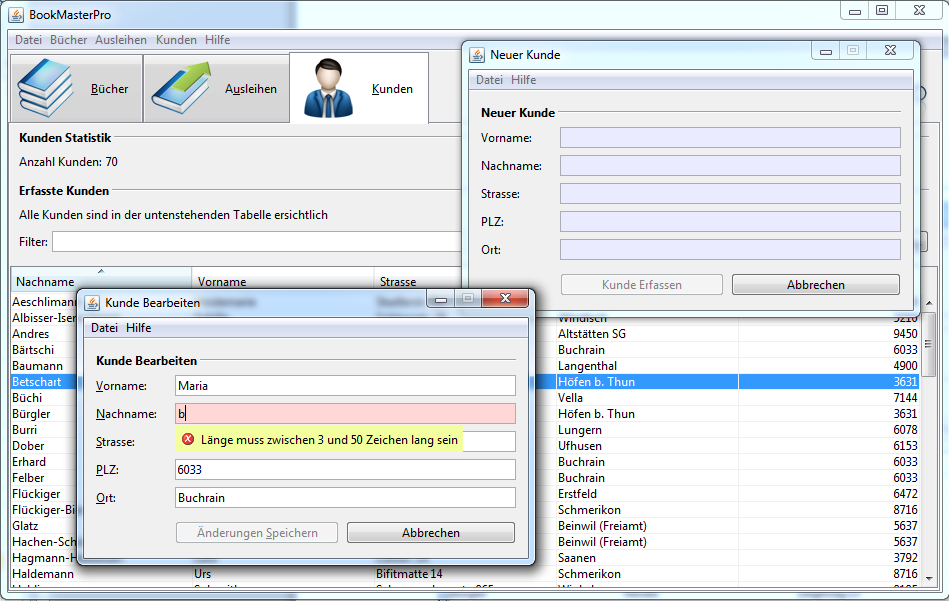


### Master-Detail View für Kundenverwaltung

Die Master View für die Kundenverwaltung sieht wie folgt aus:



Die Detail View für die Kunden sieht so aus (oben rechts: „Neuer Kunde“, unten links: „Kunde Bearbeiten“):



### Printing

Wegen zu viel Aufwand nicht implementiert.

### Internationalisierung / Mehrsprachigkeit

Wegen zu viel Aufwand nicht implementiert.

### Breadcrumbs

Nicht implementiert, da Breadcrumbs nicht zu unserem UI passen.

### Trashcan für gelöschte Objekte

Wegen zu viel Aufwand nicht implementiert. Die Library (Model) würde gewisse weitere Funktionalität bereitstellen, wie z.B. abgeschlossene Ausleihen ausgeben.

### Java 2D eingesetzt

#### Splash Screen

Für den SplashScreen haben wir zwar nicht direkt Java 2D eingesetzt, aber indirekt durch die JGoodies.

#### Logo BookMasterPro

Um das Logo zu zeichnen haben wir Java 2D eingesetzt. Als Demo haben wir auch eine Wolke gezeichnet. Die Demo kann einkommentiert werden:

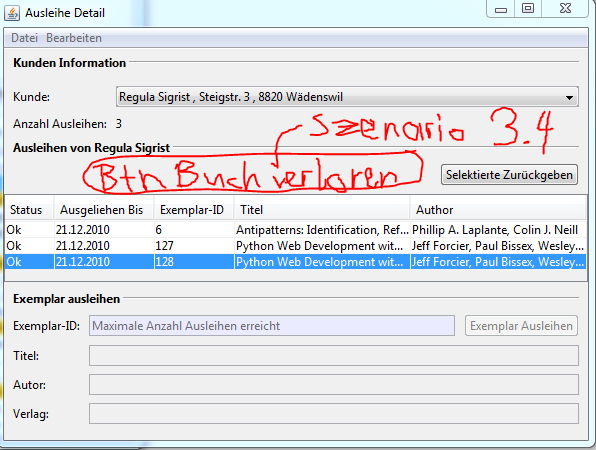


# Todos

## Prio 1

## Dones

* Ausleihe entfernen 🡪 Tabelle updaten
* Accelerators / Hotkeys im dokumentieren
* Default Actions, Bsp. <Enter> zum Speichern
* F1 + Help Menu
* label.setLabelFor(field) überall implementieren
* Default Selection nach dem Öffnen eines Fensters
* Wenn LoanDetail offen ist und ein Buchtitel im anderen Fenster bearbeitet wird (bsp. Author), dann soll das Buch auch im LoanDetail aktualisiert werden 🡪 Observer Pattern
* Wenn LoanDetail offen ist und ein neuer Kunde hinzugefügt wird im BookMaster, dann soll der neue Kunde auch in der Kundenliste von LoanDetail angezeigt werden 🡪 Observer Pattern
* Wenn LoanDetail offen ist und Kunde im anderen Fenster bearbeitet wird (bsp. Name), dann soll der Name des Kunden auch im LoanDetail aktualisiert werden 🡪 Observer Pattern
* Wenn BookDetail offen ist und eine Copy im anderen Fenster zurückgegeben oder ausgeliehen wird, dann soll sich die Tabelle mit den Loans auch im BookDetail aktualisieren 🡪 Observer Pattern
* Wenn Copy im BookDetail entfernt wird, müssen auch der Loan entfernt werden, sofern Loan vorhanden ist
* Ausleihe Detail 🡪 falls Buch ausgeliehen 🡪 neue Ausleihe nicht möglich (Button deaktiviert)
* Menus implementieren
* Neue Ausleihe erfassen…
* Doppelklick Table
* UIManager
  + Überall implementieren
  + Subframes!
* Kunden 🡪 Ausleihen anzeigen
* BookMaster Loans 🡪 Ausgewählte zurückgeben
* Wizard? Mehrere Bücher zurückgeben 🡪 Nein, besser im AusleiheMaster
* Buch zurückgeben welches nicht der Ausleihe entspricht?
  + Buch verloren 🡪 Liste „lostCopies“ in der Library
  + Idee: Button „Buch verloren“ im Ausleihe Detail entfernt Kopie



## Fragen, ob entwickeln

* Business Logik auch in der Library prüfen und bei Gui-Fehlverhalten Exceptions werfen
  + Muss nicht implementiert werden
* Loan / Buchtitel / Kunde Löschen?
  + Muss nicht implementiert werden

# Accelerators und Mnemonics

# CellEditor und CellRenderer