Examination
User Interaction (UI)

Prof. Dr.-Ing. Christian Heller <christian.heller@ba-leipzig.de>

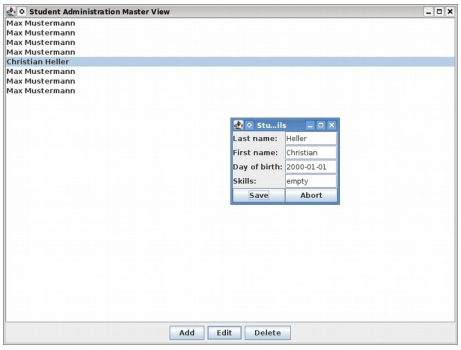
Personal Data	
First and Last Name	Ihre Daten werden von der Klausuraufsichtsperson schriftlich auf der Anwesenheitsliste festgehalten.
Matriculation Number	
Subject and Year	CS 2015
Login	Das Login "klaus..." muss unbedingt schriftlich auf der Anwesenheitsliste festgehalten werden, da sonst keine Zuordnung des Logins zu Ihrem Namen und damit keine Korrektur der Klausur möglich ist!

Examination Data	
Date	2017-03-27
Duration [min]	100
Permitted Study Aids	Dokumentation im lokalen Netzwerkverzeichnis (Intranet); NICHT gestattet sind Kommunikationsmöglichkeiten (Internet) oder Anmeldung via SSH auf dem Rechner "fileserv", wo Ihr Homeverzeichnis liegt, oder eine Anmeldung mit Ihrem Klausur-Login nach Ende der Prüfung. Dies kann leicht geprüft werden (last cs12*, Server-Log-Dateien). Bitte unterlassen Sie also Täuschungsversuche in Ihrem eigenen Interesse.
Remarks	Hinterlegen Sie alle Programme und Antworten in elektronischer Form! Es wird kein Papier angenommen. Möchten Sie Lösungen erläutern, so nutzen Sie Quelltext-Kommentare oder legen eine Text-Datei an. Speichern Sie sämtliche Daten im HOME-Verzeichnis des Nutzers, d.h. unter Windows auf Laufwerk H:\ (NICHT auf C:\ oder "Eigene Dateien")! Idealerweise legen Sie dort ein Unterverzeichnis namens "klausur" an. Lesen Sie die Aufgaben komplett durch, bevor Sie sie lösen! Die Reihenfolge der Lösung ist Ihnen überlassen. Probieren Sie immer, eine Aufgabe zu lösen, da auch auf richtige Teile nicht vollständiger Lösungen Punkte vergeben werden! Falls vom Prinzip her richtig, so werden auch alternative Lösungen akzeptiert. Sie dürfen beliebig viele Bildschirmausgaben von Werten in den Quelltext einbauen, um ein Programm besser nachvollziehen zu können. Diese Aufgabenstellung in Papierform können Sie nach dem Ende der Klausur behalten.

Evaluation						
Task	1	2	3	4	5	Summe
Points	20	10	30	30	10	100

Im Verlauf der Klausur soll eine kleine Anwendung zur Verwaltung von Studentendaten erstellt werden, welche ähnlich dem Master-Detail-Prinzip funktioniert. Dabei enthält das Hauptfenster eine namentliche Liste der Studenten. Die persönlichen Daten eines jeden Studenten können in einem Detaildialog bearbeitet werden. Siehe nachstehenden Bildschirmschnappschuss!

Auf die Kapselung wurde der Einfachheit und Übersichtlichkeit halber verzichtet. Sichtbarkeiten sowie Zugriffsmethoden können nach Belieben verwendet werden.



Task 1: Master View [20]

Zweck: Erstellen eines Hauptfensters.

a) Erstellen Sie eine Klasse namens "MasterView", welche von "JFrame" erbt! [2]

b) Geben Sie ihr eine Methode namens "initialise", in welcher Fenstertitel sowie Schließoperation (exit) zugeordnet werden! [2]

c) Weisen Sie dem Fenster außerdem eine Größe von 800 x 600 Pixeln zu und zeigen es an! [2]

d) Erzeugen Sie ein Objekt des Types "JList", welches aber nicht als lokale Variable, sondern als Instanzattribut zu speichern ist! [2]

e) Kapseln Sie es in einer "JScrollPane" und fügen es dem Rahmenfenster hinzu (CENTER)! [2]

f) Erzeugen Sie desweiteren ein Objekt des Types "JPanel", welches an eine neu zu erstellende Methode namens "initialisePanel" zu übergeben ist! [2]

g) Platzieren Sie das "JPanel"-Objekt im unteren Bereich des Rahmenfensters! [2]

h) Erzeugen Sie in der Methode "initialisePanel" drei "JButton"-Objekte mit je einer Aufschrift "Add", "Edit" und "Delete"! [2]

i) Fügen Sie die drei Knöpfe dem als Parameter übergebenen "JPanel"-Objekt hinzu! [2]

j) Erzeugen Sie das Rahmenfenster in der "main"-Methode einer neuen Startklasse namens "Launcher"! Rufen Sie seine "initialise"-Methode auf! [2]

Ergebnis: Nach dem Programmstart ist das Hauptfenster zu sehen und kann beendet werden.

Task 2: Data Model [10]

Zweck: Bereitstellen einer Entitätsklasse sowie einer Liste als Datenspeicher.

a) Erzeugen Sie in der "main"-Methode ein Objekt des Types "DefaultListModel"! Übergeben Sie es an die "initialise"-Methode der Klasse "MasterView"! [2]

b) Weisen Sie es dort dem Instanzattribut vom Typ "JList" zu! [2]

Hinweis: Nutzen Sie dessen "setModel"-Methode!

c) Erstellen Sie eine neue Klasse namens "Student" mit vier Attributen vom Typ "String", namens: "lastName", "firstName", "dayOfBirth", "skills"! [2]

d) Versehen Sie diese Instanzattribute in einer parameterlosen "initialise"-Methode mit den Anfangswerten: "Mustermann", "Max", "2000-01-01", "empty"! [2]

e) Überschreiben Sie die geerbte "toString"-Methode und lassen Sie sie eine Zeichenkette bestehend aus dem Vor- und Nachnamen des Studenten zurückliefern! [2]

Ergebnis: Keine sichtbare Veränderung. Das Datenmodell wird durch das System gereicht.

Task 3: Event Controller [30]

Zweck: Reagieren auf im Hauptfenster getätigte Mausklicks.

a) Nutzen Sie eine Klasse mit dem Namen "Controller" zum Verarbeiten von Ereignissen des Types "ActionEvent"! [2]

b) Definieren Sie darin drei "String"-Konstanten für die Aktionen bzw. Kommandos: "add", "edit", "delete"! [2]

c) Definieren Sie außerdem zwei Instanzattribute: "model" vom Typ "DefaultListModel" sowie "view" vom Typ "MasterView", deren Anfangswerte mittels an eine "initialise"-Methode übergebener Parameter zuzuweisen sind! [2]

d) Filtern Sie in der Ereignisverarbeitungsmethode alle drei Ereignisse heraus! Geben Sie eine Fehlermeldung auf der Konsole aus, wenn keines bekannt ist! [2]

e) Erstellen und initialisieren Sie im "add"-Zweig ein Objekt vom Typ "Student"! [2]

f) Fügen Sie es dem "model"-Attribut mittels passender Methode "addElement" zu! [2]

g) Greifen Sie via "view"-Attribut auf die Liste zu! Verwenden Sie eine ihrer Methoden, um den letzten Eintrag zu selektieren! [2]

Hinweis 1: Nutzen Sie (this.model.getSize() - 1) als Index, da neue Elemente immer am Ende angehangen werden!

Hinweis 2: Das Selektieren ist wichtig, da alle anderen Ereignisverarbeitungsmethoden den aktuellen Index verwenden.

h) Bestimmen Sie im "edit"-Zweig via "view" und "list" den Index des aktuell selektierten Studenten! [2]

i) Fangen Sie ungültige Rückgabewerte wie z. B. "-1" ab und bringen eine passende Fehlermeldung auf der Konsole! [2]

j) Holen Sie sich mittels des gültigen Indexes anschließend das passende Studenten-Objekt aus dem "model"-Attribut! [2]

Hinweis: Eine Typumwandlung auf "Student" ist nötig.

k) Bestimmen Sie im "delete"-Zweig wiederum zunächst den Index des aktuell selektierten Studenten und fangen ungültige Werte ab! Löschen Sie dann via "model"-Attribut das Studenten-Objekt mit dem zuvor ermittelten Index! [2]

l) Erzeugen Sie das "Controller"-Objekt in der "main"-Methode der "Launcher"-Klasse! Rufen Sie seine "initialise"-Methode auf, wobei die passenden Argumente mitzugeben sind! [2]

m) Ergänzen Sie die "initialise"-Methode der Klasse "MasterView" durch einen zweiten Parameter vom Typ "Controller"! Delegieren Sie ihn weiter an die Methode "initialisePanel"! [2]

n) Weisen Sie das "Controller"-Objekt den dortigen drei Knöpfen als Ereignislauscher zu! [2]

o) Weisen Sie den drei Knöpfen außerdem die jeweils passende Aktion bzw. Kommando zu, wofür die Konstanten der Klasse "Controller" zu verwenden sind! [2]

Ergebnis: Veränderungen am Datenmodell (Hinzufügen, Löschen) sind möglich.

Task 4: Details Dialog [30]

Zweck: Bearbeiten von Detaildaten in einem eigenen Dialogfenster.

a) Erstellen Sie eine Klasse namens "Details", welche von "JDialog" erbt! Geben Sie ihr vier Attribute des Types "JTextField": "lastName", "firstName", "dayOfBirth", "skills"! [2]

b) Erstellen Sie eine "initialise"-Methode zur Erzeugung folgender zehn Objekte: vier vom Typ "JTextField", welche den entsprechenden Instanzattributen zuzuweisen sind; vier vom Typ "JLabel", welche lokalen Variablen zuzuweisen sind; zwei vom Typ "JButton", welche ebenfalls lokalen Variablen zuzuweisen sind! [2]

c) Legen Sie die Aufschrift der vier Beschriftungsfelder (Labels) fest auf: "Last name: ", "First name: ", "Day of birth: ", "Skills: "! Legen Sie die Aufschrift der zwei Knöpfe (Buttons) fest auf: "Save", "Abort"! [2]

d) Nutzen Sie ein "GridLayout", um die Objekte dem Dialogfenster hinzuzufügen! [2]

e) Geben Sie dem Dialog einen Fenstertitel! Legen Sie seine Größe so fest, dass sie sich automatisch den enthaltenen grafischen Komponenten anpasst! Ändern Sie optional auch seine Position! Bewirken Sie, dass das Fenster beim Schließen nicht zerstört, sondern nur versteckt wird! Machen Sie das Dialogfenster modal! [2]

f) Reichen Sie das "Controller"-Objekt von der "main"-Methode aus weiter an die "initialise"-Methode der "Details"-Klasse! Weisen Sie es dort den beiden Knöpfen als Ereignislauscher zu! [2]

g) Weisen Sie den beiden Knöpfen außerdem je eine passende Aktion bzw. Kommando "save" oder "abort" zu, wofür in der "Controller"-Klasse zwei neue Konstanten zu erstellen sind! [2]

h) Erstellen Sie in der "Details"-Klasse eine Methode namens "open", die einen Parameter vom Typ "Student" entgegennimmt! Weisen Sie darin den vier Textfeldern die passenden Werte des Studenten-Objektes zu! Schalten Sie das Dialogfenster anschließend sichtbar! [2]

i) Erstellen Sie auf gleiche Weise eine Methode namens "save" mit umgekehrter Funktionalität, d. h. die Inhalte der Textfelder sollen dem Studenten-Objekt zugewiesen werden! Verstecken Sie das Dialogfenster anschließend! [2]

j) Erstellen Sie schließlich eine weitere Methode namens "abort", in welcher das Dialogfenster versteckt wird! [2]

k) Assoziieren Sie die Klassen "MasterView" und "Details", indem erstere ein Attribut vom Typ letzterer bekommt! Erzeugen Sie ein "Details"-Objekt in der "initialise"-Methode und weisen Sie es dem Attribut zu! Rufen Sie außerdem seine "initialise"-Methode auf! [2]

l) Passen Sie die Ereignisverarbeitungsmethode der "Controller"-Klasse so an, dass ihre Zweige für "add" und "edit" jeweils den "Details"-Dialog via dessen "open"-Methode sichtbar schalten! [2]

m) Ergänzen Sie einen Zweig für die "abort"-Aktion, in welchem die gleichnamige Methode der "Details"-Klasse aufgerufen wird! [2]

n) Ergänzen Sie einen weiteren Zweig für die "save"-Aktion! Bestimmen Sie wie in den anderen Zweigen den Index des aktuell selektierten Studenten und fangen ungültige Werte ab! Holen Sie sich mittels des gültigen Indexes anschließend das passende Studenten-Objekt aus dem "model"-Attribut! [2]

o) Rufen Sie, unter Mitgabe des Studenten-Objektes, anschließend die "save"-Methode des "Details"-Dialoges auf! Veranlassen Sie ein Neuzeichnen der "MainView", damit die geänderten Studentendaten im Hauptfenster angezeigt werden! [2]

Ergebnis: Das Dialogfenster gestattet das Anlegen und Bearbeiten von Studenten.

Task 5: Internationalisation (i18n) [10]

Zweck: Auslagern von als Literal hart im Quelltext stehenden Texten in externe Dateien.

a) Stellen Sie zwei Ressourcen-Dateien bereit: für in Deutschland gesprochenes Deutsch und für in Großbritannien gesprochenes Englisch! Geben Sie dort die Übersetzungen der Aufschriften der drei im Hauptfenster verwendeten Knöpfe an! Fügen Sie außerdem die Übersetzungen der zwei im Detail-Dialog verwendeten Knöpfe sowie der vier Bezeichnungsfelder (Labels) ein! [2]

b) Erzeugen Sie in der "main"-Methode ein Objekt vom Typ "Locale", wofür zwei Kommandozeilenargumente verwendet werden! Das erste soll für das Sprach-, das zweite für das Länderkürzel stehen! [2]

c) Erzeugen Sie unter Verwendung des "Locale"-Objektes ein Objekt vom Typ "ResourceBundle"! Der Name bzw. das Präfix der zu öffnenden Dateien soll "labels" lauten! [2]

d) Reichen Sie das "ResourceBundle"-Objekt durch an die "initialise"-Methode aller relevanten Klassen! [2]

e) Verwenden Sie es dort, um Texte aus der entsprechenden Ressourcen-Datei auszulesen! [2]

Ergebnis: Die gewünschte Sprache (Deutsch oder Englisch) ist beim Programmstart wählbar.

Viel Erfolg!