Prüfung
User Interaction (UI)

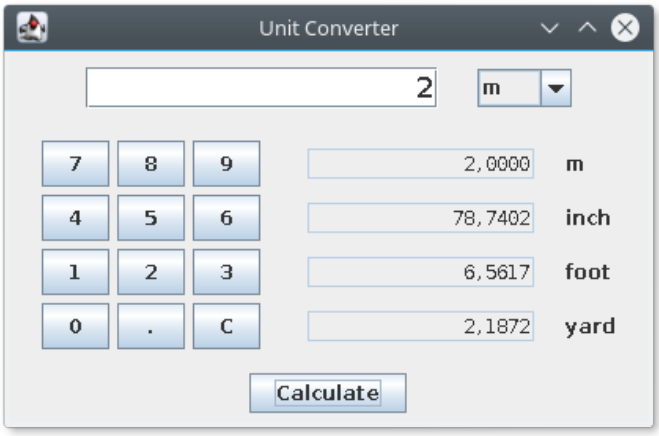
Prof. Dr.-Ing. Christian Heller <christian.heller@ba-leipzig.de>

Student	
Vor- und Nachname	Ihre Daten werden von der Klausuraufsichtsperson schriftlich auf der Anwesenheitsliste festgehalten.
Matrikelnummer	
Studienrichtung und Jahr	CS 2017-1
Anmeldename	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!

Prüfung	
Datum	Dezember 2018
Dauer [min]	100
Hilfsmittel	Dokumentation im lokalen Netzwerk (Intranet) sowie Recherche im Internet. NICHT gestattet: * Kommunikation in jeglicher Form * Anmeldung via SSH auf dem Rechner "fileserv" * Anmeldung mit Klausur-Login nach Ende der Prüfung Dies kann leicht geprüft werden (last cs16*, Server-Log-Dateien). Bitte unterlassen Sie also Täuschungsversuche in Ihrem eigenen Interesse.
Bemerkungen	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. Bitte duplizieren Sie Ihre Quelltextdateien (workspace) NICHT, da beim Korrigieren dann beide durchsucht werden müssen, was sinnlosen Aufwand verursacht. Diese Aufgabenstellung in Papierform können Sie nach dem Ende der Klausur behalten.

Bewertung						
Aufgabe	1	2	3	4	5	Summe
Punkte	10	20	30	10	30	100

Im Verlauf der Klausur soll ein Konverter für Längeneinheiten erstellt werden. Siehe Bildschirmschnappschuss!



Aufgabe 1: Frame [10]

Zweck: Ein Rahmenfenster wird gebaut.

a) Erstellen Sie eine Klasse namens „View“, die als Swing-Rahmenfenster dienen soll! [2]

b) Erstellen Sie darin eine „initialise“-Methode, in welcher dem Fenster ein Titel zugewiesen wird! [2]

c) Geben Sie dem Fenster eine Schließoperation zum Beenden der Anwendung! [2]

d) Setzen Sie die Fenstergröße so, dass alle enthaltenen Komponenten dargestellt werden! Schalten Sie das Fenster sichtbar! [2]

e) Erstellen Sie eine Startklasse namens „Launcher“, in deren „main“-Methode das Fenster erzeugt und initialisiert wird! [2]

Ergebnis: Das Hauptfenster der Anwendung wird angezeigt.

Aufgabe 2: View [20]

Zweck: Befüllen des Fensters mit Komponenten.

Es werden vier Kacheln unterschieden:

- `das Texteingabefeld zusammen mit dem Auswahlfeld oben`
- `die Eingabeknöpfe mittig`
- `die berechneten Werte mit Einheiten rechts`
- `der Berechnen-Knopf unten`

a) Erzeugen Sie in der „initialise“-Methode der View vier Kacheln des Typs „JPanel“! [2]

b) Erzeugen Sie desweiteren ein Array mit den Werten { "m", "inch", "foot", "yard" }! [2]

c) Erzeugen Sie ein Auswahlfeld des Typs „JComboBox“ und speichern Sie es als Instanzattribut! Verwenden Sie das oben erzeugte Array zum Befüllen! [2]

d) Erzeugen Sie ein Textfeld der Größe 20! Speichern Sie es als Instanzattribut! [2]

e) Definieren Sie für das Textfeld eine rechtsbündige Ausrichtung! Weisen Sie ihm als Schrift „Monospaced PLAIN“ der Größe: 18 pt“ zu! [2]

f) Weisen Sie der „JComboBox“ das „Action“-Kommando „choose“ zu! [2]

g) Definieren Sie für die erste Kachel ein „FlowLayout“ mit mittiger Ausrichtung, einem horizontalen Abstand von 25 px und einem vertikalen Abstand von 10 px! Fügen Sie ihr das Textfeld wie auch das Auswahlfeld (JComboBox) hinzu! [2]

h) Weisen Sie der zweiten Kachel einen Rand des Typs „EmptyBorder“ zu, dessen linker Abstand 20 px beträgt! Weisen Sie der zweiten Kachel ein „GridLayout“ mit vier Zeilen und drei Spalten zu! Setzen Sie den horizontalen wie auch vertikalen Zwischenraum auf 5 px! [2]

i) Weisen Sie auf gleiche Weise der dritten Kachel ein „GridLayout“ mit vier Zeilen und zwei Spalten zu! Der horizontale Abstand soll 10 px, der vertikale 5 px betragen! [2]

j) Verwenden Sie für die „View“ selbst ein „BorderLayout“ mit horizontalen und vertikalen Zwischenräumen von 10 px! Fügen Sie ihr die vier Kacheln in passender Weise hinzu! [2]

Ergebnis: Die Anwendungsoberfläche zeigt erste Komponenten.

Aufgabe 3: Buttons and Fields [30]

Zweck: Ergänzen der noch fehlenden Knöpfe und Felder in der Nutzungsoberfläche.

a) Erzeugen Sie in der „View“-Klasse zwei Methoden namens „addButton“ und „addField“! [2]

b) Lassen Sie beide zwei Parameter empfangen: einen vom Typ „JPanel“ und einen weiteren vom Typ „String“! [2]

c) Erzeugen Sie in der „addButton“-Methode ein „JButton“-Objekt und weisen Sie ihm als Aufschrift die als Parameter übergebene Zeichenkette zu! [2]

d) Fügen Sie den Knopf zur als Parameter übergebenen Kachel hinzu! [2]

e) Erzeugen Sie in der „addField“-Methode ein „JTextField“-Objekt der Größe 20 sowie ein „JPanel“-Objekt! [2]

f) Weisen Sie dem Textfeld als Schrift „Monospaced PLAIN“ der Größe 12 pt zu! [2]

g) Richten Sie es rechtsbündig aus! Verhindern Sie ein Bearbeiten des Textfeldes! [2]

h) Weisen Sie der Kachel vom Typ „JPanel“ ein FlowLayout mit linksbündiger Ausrichtung zu! Der horizontale Abstand soll 20 px, der vertikale 5 px betragen! [2]

i) Fügen Sie der Kachel das zuvor erzeugte Textfeld hinzu! Erzeugen Sie außerdem ein „JLabel“-Objekt, welches ebenfalls hinzuzufügen ist! Verwenden Sie den übergebenen Zeichenketten-Parameter als Aufschrift für das „JLabel“! [2]

j) Fügen Sie schließlich die lokal erzeugte Kachel der als Parameter übergebenen Kachel hinzu! Lassen Sie die Methode das lokal erzeugte Textfeld als Wert zurückliefern! [2]

k) Rufen Sie die „addButton“-Methode innerhalb der „initialise“-Methode in passender Weise auf, um zwölf Knöpfe zu erzeugen und der dortigen zweiten Kachel hinzuzufügen! [2]

l) Rufen Sie die „addField“-Methode innerhalb der „initialise“-Methode vier mal auf, um die Anzeigefelder rechts im Bildschirmschnappschuss zu erzeugen! [2]

m) Weisen Sie die Rückgabe-Objekte der „addField“-Methode vier neu zu erstellenden Instanzattributen der Klasse „View“ zu! [2]

n) Rufen Sie die „addButton“-Methode ein letztes Mal auf, um den Knopf mit der Aufschrift „Calculate“ zu erzeugen und der unten liegenden Kachel hinzuzufügen! [2]

o) Erstellen Sie eine Methode namens „reset“, in welcher alle vier Textfelder (die ja als Instanzattribute vorhanden sind) geleert werden (Zuweisen einer leeren Zeichenkette)! [2]

Ergebnis: Die Anwendungsoberfläche entspricht dem Bildschirmschnappschuss.

Aufgabe 4: Model [10]

Zweck: Bereitstellen und Umrechnen der Daten.

a) Erstellen Sie eine Klasse namens „Model“! Geben Sie ihr die vier Instanzattribute: meter, inch, foot und yard vom Typ „double“! Geben Sie ihr außerdem ein Instanzattribut „unit“ vom Typ „String“ sowie eines namens „builder“ vom Typ „StringBuilder“! [2]

b) Generieren Sie Zugriffsmethoden für die vier Längeneinheiten! Passen Sie die „set“-Methoden gemäß folgender Tabelle an [2]:

Gegeben	**Meter**	**Inch**	**Foot**	**Yard**

Formel	meter = m;
	inch = 1. / 0.0254 * m;
	foot = 1. / 0.3048 * m;
	yard = 1. / 0.9144 * m;
		meter = 0.0254 * i;
		inch = i;
		foot = 1. / 12. * i;
		yard = 1. / 36. * i;
			meter = 0.3048 * f;
			inch = 12 * f;
			foot = f;
			yard = 1. / 3. * f;
				meter = 0.9144 * y;
				inch = 36 * y;
				foot = 3 * y;
				yard = y;

c) Erstellen Sie eine Methode namens „calculate“, welche zwei Parameter vom Typ „double“ (für den Wert) und „String“ (für die Einheit) empfängt! Rufen Sie in Abhängigkeit von der Einheit die jeweils passende „set“-Methode auf! [2]

d) Definieren Sie die übliche „initialise“-Methode! Übergeben Sie einen Anfangswert vom Typ „double“, welcher aber NICHT direkt zugewiesen werden soll, sondern via „setMeter“-Methode! Übergeben Sie außerdem einen „String“-Anfangswert für das „unit“-Attribut! Erzeugen Sie schließlich ein Objekt des Typs „StringBuilder“ und weisen es dem passenden Attribut zu! [2]

e) Definieren Sie eine Klassenmethode (statisch) „format“, die einen Parameter vom Typ „double“ übergeben bekommt! Nutzen Sie darin ein Objekt des Typs „DecimalFormat“, um die übergebene Gleitkommazahl auf vier Nachkommastellen zu begrenzen! [2]

Ergebnis: Die nötige Hintergrundarbeit für die Datenhaltung wurde erledigt.

Aufgabe 5: Controller [30]

Zweck: Verarbeiten von Nutzereingaben. Hinweis: Für alle Knöpfe, denen kein explizites Kommando zugewiesen wurde, kann ihre Aufschrift als Kommando verwendet werden.

a) Erstellen Sie eine „Controller“-Klasse, die Ereignisse vom Typ „ActionEvent“ abfängt! Halten Sie darin per Instanzattribut je einen Verweis auf das Model und auf die View! Generieren Sie einen passenden Konstruktor zur Initialisierung! [2]

b) Extrahieren Sie in der Methode zur Ereignisverarbeitung zunächst das empfangene Kommando und speichern es in einer lokalen Variablen! Setzen Sie anschließend die Länge des „StringBuilder“-Objektes via „Model“ auf (arithmetisch) Null zurück! Weisen Sie den aktuellen Inhalt des Texteingabefeldes (via „View“) dem „StringBuilder“ mittels „append“-Methode zu! [2]

c) Prüfen Sie mittels „compareTo“-Methode, ob es sich beim übergebenen Kommando um eine Ziffer zwischen 0 und 9 handelt! Falls ja, so übergeben Sie es an den „StringBuilder“ (append)! Weisen Sie dem Texteingabefeld den Wert des „StringBuilder“-Objektes zu („toString“)! [2]

d) Prüfen Sie anschließend, ob ein Dezimalpunkt eingegeben wurde! Prüfen Sie desweiteren, ob im „StringBuilder“-Objekt bereits ein Dezimalpunkt vorhanden ist! Falls nicht, so gehen Sie anschließend vor wie in Aufgabe c)! [2]

e) Kommt als Kommando das „C“ an, so ist die Länge des „StringBuilder“-Objektes auf (arithmetisch) Null zu setzen! Dem Texteingabefeld ist eine leere Zeichenkette zuzuweisen! [2]

f) Kommt als Kommando die Aufschrift des Knopfes „Calculate“ an, so ist zunächst die „reset“-Methode der „View“ aufzurufen! Rufen Sie dann die „calculate“-Methode des Models auf und geben Sie als Argumente mit: (1) die in eine Gleitkommazahl umgewandelte Zeichenkette des „StringBuilder“-Objektes; (2) das „unit“-Attribut des Models! [2]

g) Fangen Sie eine eventuell auftretende „NumberFormatException“ ab und setzen Sie in diesem Falle die Werte zurück auf die gleiche Weise wie in Aufgabe e)! [2]

h) Ersetzen Sie im „StringBuilder“-Objekt ein zuvor eventuell manuell eingegebenes Dezimalkomma durch den Dezimalpunkt! [2]

i) Kommt das Kommando „choose“ an, so weisen Sie den gewählten „JComboBox“-Eintrag dem Attribut „unit“ des Models zu! Rufen Sie anschließend die „reset“-Methode der „View“ auf! [2]

j) Veranlassen Sie am Ende der Methode zur Ereignisverarbeitung für ALLE Fälle ein Neuzeichnen der View! [2]

k) Instanziieren Sie die „Model“-Klasse in der „main“-Methode und rufen Sie ihre „initialise“-Methode auf, wobei 0.0 und „m“ als Initialwerte mitzugeben sind! Erweitern Sie die „initialise“-Methode der Klassen „View“ und „Controller“ um das „Model“ als Parameter und weisen Sie es dortigen Instanzattributen zu! [2]

l) Instanziieren Sie die „Controller“-Klasse in der „main“-Methode und rufen Sie ihre „initialise“-Methode auf! [2]

m) Erweitern Sie insbesondere die Methoden „initialise“ und „addButton“ der „View“-Klasse um einen Parameter vom Typ „Controller“! Weisen Sie ihn als „ActionListener“ sowohl den Knöpfen wie auch der „JComboBox“ zu! [2]

n) Überschreiben Sie in der „View“-Klasse die geerbte Methode „paint“! Markieren Sie das Überschreiben mittels Annotation! Rufen Sie zunächst die entsprechende Methode der Elternklasse auf! [2]

o) Setzen Sie in der „paint“-Methode alle vier Textausgabefelder auf den jeweils passenden Wert des Models! Nutzen Sie die „format“-Methode der „Model“-Klasse, um die Werte zu formatieren! [2]

Ergebnis: Die Anwendung ist voll funktionstüchtig.

Viel Erfolg!