



## Klausur

**Studiengang:** Wirtschaftsinformatik  
**Jahrgang:** 2017  
**Modul:** Programmierung und Programmiertechniken  
**Veranstaltung:** Fortgeschrittene Programmierung, Algorithmen und Datenstrukturen  
**Prüfer/-in:** Daniel Appenmaier  
**Datum:** 17.09.2018  
**Bearbeitungszeit:** 100 Minuten  
**Max. Punktzahl:** 100 Punkte  
**Hilfsmittel:** Taschenrechner (nicht programmierbar) und Beiblatt zur Klausur  
**Sonstige Hinweise:** Benötigte Klassen- oder Schnittstellen-Imports müssen von Ihnen nicht explizit angegeben werden!

**Matrikelnummer:** \_\_\_\_\_ (hier eintragen!)

**Viel Erfolg!**

**Durch Prüfer/-in auszufüllen:**

Aufgabe	1	2	3	4	Gesamt
Maximale Punktzahl	30	30	20	20	100
Erreichte Punktzahl					

**Aufgabe 1 (30 Punkte)**

- a) Erläutern Sie den wesentlichen Unterschied zwischen einer „normalen“ Methode und einer abstrakten Methode und benennen Sie, ob eine abstrakte Methode zwingend in einer abstrakten Klasse definiert sein muss (3 Punkte).
- b) Erläutern Sie, was man in der Programmierung unter dem Begriff *Downcast* versteht (2 Punkte).
- c) Erläutern Sie den technischen Aufbau einer Aufzählungskonstanten (2 Punkte).
- d) Erläutern Sie den wesentlichen Vorteil des Ausnahmenbehandlungs-Prozesses in Java (2 Punkte).
- e) Skizzieren und erläutern Sie die Ereignisbehandlung in Java und benennen Sie das skizzierte Modell (7 Punkte).
- f) Erläutern Sie, was man in der Programmierung unter dem Begriff *Serialisierung* versteht (2 Punkte).
- g) Benennen Sie die 4 Varianztypen bei der generischen Programmierung (2 Punkte).
- h) Skizzieren Sie den Aufbau einer *ArrayList* und einer *LinkedList* anhand der Zahlen 4, 2, 5, 1 und 3 und erläutern Sie, bei welchen Operationen die jeweilige Liste Vorteile besitzt (6 Punkte).
- i) Erläutern Sie, warum es sinnvoll ist, Datensammlungen vor der Suche zu Sortieren und benennen Sie die Zwei in der Vorlesung vorgestellten Suchverfahren, die nach dem Teile-und-Herrsche-Prinzip funktionieren (4 Punkte).

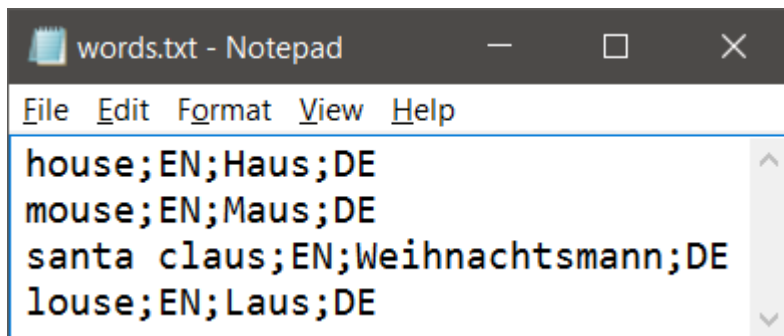
## **Aufgabe 2 (30 Punkte)**

Gegeben sei das beigefügte Klassendiagramm.

- Implementieren Sie die Methode *compareTo(Word)* der Klasse *Word* so, dass Wörter aufsteigend nach ihrem Wert (Attribut *value*) sortiert werden (2 Punkte).
- Implementieren Sie die Methode *addEntry(Word, Word)* der Klasse *Dictionary* so, dass zunächst überprüft wird, ob die Sprache (Attribut *language* der Klasse *Word*) des ersten eingehenden Wortes dem Attribut *sourceLanguage* entspricht und ob die Sprache des zweiten eingehenden Wortes dem Attribut *targetLanguage* entspricht. Im Fehlerfall soll die Ausnahme *InvalidLanguageException* ausgelöst werden, im Erfolgsfall sollen die beiden Wörter dem Wörterbuch (Attribut *entries*) als neuer Eintrag hinzugefügt werden (6 Punkte).
- Implementieren Sie die Methode *importDictionary()* der Klasse *Dictionary* so, dass alle Daten aus der Datei *C:/Temp/words.txt* ausgelesen werden und mit Hilfe der Methode *addEntry(Word, Word)* als entsprechende Einträge dem Wörterbuch hinzugefügt werden. Fangen Sie alle möglichen Ausnahmen ab und lösen Sie im Fehlerfall die Ausnahme *ImportException* aus (16 Punkte).
- Implementieren Sie die Methode *getTranslation(String)* der Klasse *Dictionary* so, dass überprüft wird, ob die eingehende Zeichenkette im Wörterbuch (Attribut *entries*) als Schlüssel (Attribut *value* der Klasse *Word*) existiert. Im Erfolgsfall soll der dazugehörige Wert als Zeichenkette zurückgegeben werden (6 Punkte).

### Tipps und Tricks

- Die Methode *split(String)* der Klasse *String* teilt eine Zeichenkette anhand der eingehenden Zeichenkette in mehrere kleinere Zeichenketten und gibt diese als String-Array zurück.
- Sollten Sie die Klassen *FileReader* und *BufferedReader* verwenden wollen, denken Sie bitte daran, die Ausnahme *IOException* abzufangen.
- Der Aufbau der Datei *C:/Temp/words.txt* sieht wie folgt aus:



```
words.txt - Notepad
File Edit Format View Help
house;EN;Haus;DE
mouse;EN;Maus;DE
santa claus;EN;Weihnachtsmann;DE
louse;EN;Laus;DE
```

### **Aufgabe 3 (20 Punkte)**

Gegeben sei das beigefügte Klassendiagramm, sowie der beigefügte Entwurf einer grafischen Benutzeroberfläche. Erstellen Sie die Klasse *DictionaryFrame*.

#### Hinweise zur Klasse *DictionaryFrame*

- Der Konstruktor soll das Attribut *dictionary* als Wörterbuch Englisch->Deutsch initialisieren und die grafische Benutzeroberfläche erzeugen.
- Die Methode *actionPerformed(ActionEvent)* soll den Text des Eingabefeldes an die Methode *getTranslation(String)* der Klasse *Dictionary* übergeben. Im Erfolgsfall (Rückgabewert ungleich *null*) soll der Rückgabewert der Methode auf einem Nachrichtendialog angezeigt werden, im Fehlerfall soll der abgebildete Nachrichtendialog angezeigt werden.

#### Hinweise zur grafischen Benutzeroberfläche

- Das Hauptfenster (Titel: Wörterbuch Englisch->Deutsch) soll eine Größe von 450x100 besitzen.
- Das Eingabefeld soll eine Spaltenanzahl von 20 besitzen.
- Das Hauptfenster soll beim zentriert positioniert werden und beim Beenden das dazugehörige Programm ebenfalls beenden.

#### Tipps & Tricks

- Die statische Methode *showMessageDialog(Component, Object, String, int)* der Klasse *JOptionPane* erzeugt aus dem eingehenden Text (Parametertyp *Object*), dem eingehenden Titel (Parametertyp *String*), sowie dem eingehenden Nachrichtentyp (Parametertyp *int*) einen Nachrichtendialog.
- Die Klasse *JOptionPane* stellt u.a. die Konstanten *INFO\_MESSAGE* und *ERROR\_MESSAGE* für den Nachrichtentyp bereit.

**Aufgabe 4 (20 Punkte)**

Gegeben sei die abgebildete Liste, sowie die beigefügte Implementierung des Quicksort-Verfahrens. Wenden Sie das Quicksort-Verfahren auf die Liste an. Schreiben Sie hierzu für jeden Durchlauf die jeweilige Reihenfolge der Zahlen auf, kreisen Sie den jeweiligen Teiler ein und markieren sie die entstandenen neuen Bereiche.

Index	0	1	2	3	4	5	6	7	8
0	5								
1	1								
2	7								
3	9								
4	3								
5	2								
6	8								
7	6								
8	4								

Durchgang	l	r	m	i	j	l/j	i/r
1							
2							
3							
4							
5							
6							
7							
8							