



Wiederholklausur

Studiengang: Wirtschaftsinformatik
Jahrgang: 2016
Modul: Programmierung und Programmiertechniken
Veranstaltung: Fortgeschrittene Programmierung, Algorithmen und Datenstrukturen
Prüfer/-in: Daniel Appenmaier
Datum: 21.11.2017
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:

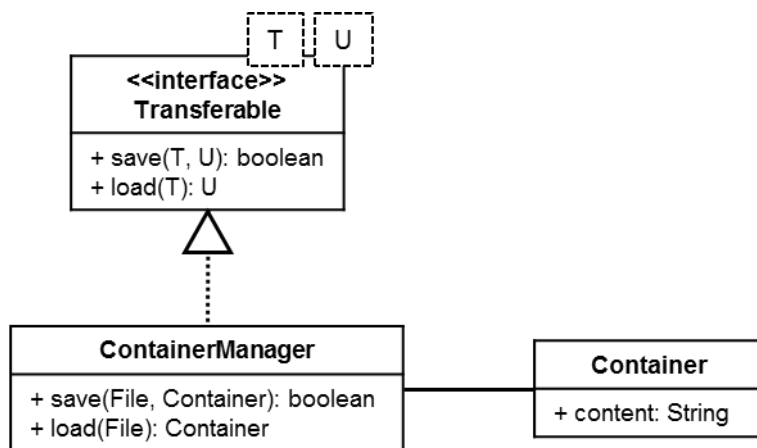
| Aufgabe | 1 | 2 | 3 | 4 | 5 | Gesamt |
|---------------------|----|----|----|----|----|--------|
| Maximale Punktzahl | 26 | 20 | 16 | 16 | 22 | 100 |
| Erreichte Punktzahl | | | | | | |

Aufgabe 1 (26 Punkte)

Gegeben sei das abgebildete Klassendiagramm.

- Erstellen Sie die Schnittstelle *Transferable* (3 Punkte), sowie die Klassen *Container* (1 Punkt) und *ContainerManager* (14 Punkte).
- Erläutern Sie kurz das Datenstrom-Prinzip in Java (2 Punkte) und benennen Sie die Klassen zur Ein- und Ausgabe...
 - byte-orientierter Daten in Dateien (1 Punkt)
 - zeichenorientierter Daten in Puffer (1 Punkt)
- Erläutern Sie kurz anhand des Klassendiagramms, was man unter *Serialisierung* versteht (2 Punkte).
- Erläutern Sie kurz den wesentlichen Unterschied zwischen einer Klasse und einer Schnittstelle (2 Punkte).

Klassendiagramm



Wörterbuch

transferable = übertragbar, container = Behälter

Hinweise zur Klasse *ContainerManager*

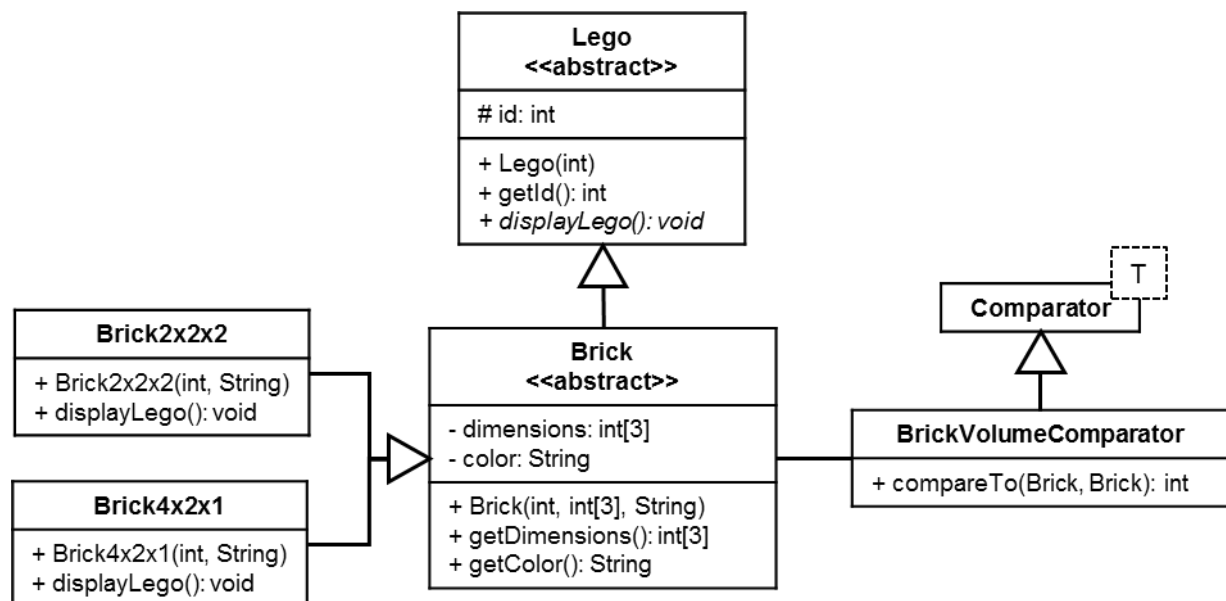
- Die Methode `save(File, Container)` soll den einkommenden Behälter in der einkommenden Datei speichern!
- Die Methode `load(File)` soll einen Behälter aus der einkommenden Datei auslesen und diesen zurückgeben!

Aufgabe 2 (20 Punkte)

Gegeben sei das abgebildete Klassendiagramm.

- Erstellen Sie die Klassen *Lego* und *BrickVolumeComparator* (8 Punkte).
- Erstellen Sie eine ausführbare Klasse, die die aufgeführten Bausteine erzeugt, in einer Liste speichert, diese mit Hilfe der Klasse *BrickVolumeComparator* aufsteigend ihres Volumens sortiert und anschließend über eine for-Schleife auf der Konsole ausgibt (8 Punkte).
 - Klasse: *Brick2x2x2*; Id: 1; Abmessungen: 2, 2, 2; Farbe: Rot
 - Klasse: *Brick2x2x2*; Id: 2; Abmessungen: 2, 2, 2; Farbe: Grün
 - Klasse: *Brick4x2x1*; Id: 3; Abmessungen: 4, 2, 1; Farbe: Blau
- Erläutern Sie kurz anhand des Klassendiagramms das Vererbungsprinzip in Java (4 Punkte).

Klassendiagramm



Wörterbuch

brick = Baustein, dimensions = Abmessungen

Hinweise zur Klasse *Lego*

- Der Konstruktor soll das Attribut *id* initialisieren!
- Die Methode *getId()* soll die Id zurückgeben!

Hinweise zur Klasse *Brick*

- Der Konstruktor initialisiert die Attribute *id*, *dimensions* und *color*!
- Die get-Methode geben die Abmessungen bzw. die Farbe zurück!

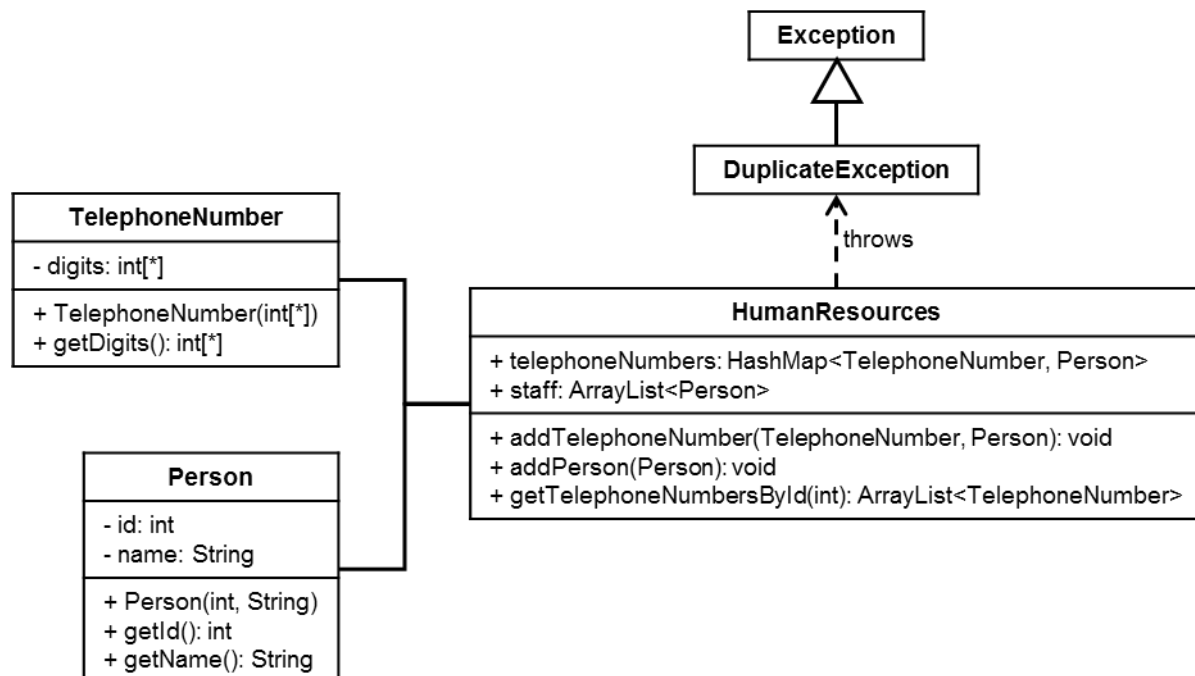
Hinweise zu den Klassen *Brick2x2x2* und *Brick4x2x1*

- Der Konstruktor initialisiert die Attribute *id*, sowie *color* und initialisiert das Attribut *dimensions* mit dem Wert *2x2x2* bzw. *4x2x1*!
- Die Methode *displayLego()* gibt alle Attribute auf der Konsole aus!

Aufgabe 3 (16 Punkte)

Gegeben sei das abgebildete Klassendiagramm. Erstellen Sie die Klasse *HumanResources* anhand des abgebildeten Klassendiagramms.

Klassendiagramm



Wörterbuch

staff = Personal, digits = Ziffern

Hinweise zur Klasse *HumanResources*

- Die Methode `addTelephoneNumber(TelephoneNumber, Person)` soll dem Attribut `telephoneNumbers` die einkommende Telefonnummer samt zugehöriger Person hinzufügen!
- Die Methode `addPerson(Person)` soll dem Attribut `staff` die einkommende Person hinzufügen! Wenn die Person bereits Teil des Personals ist, soll die Ausnahme `DuplicateException` ausgelöst werden!
- Die Methode `getTelephoneNumbersById(int)` soll alle Telefonnummern zur einkommenden Personen-Id zurückgeben!

Aufgabe 4 (16 Punkte)

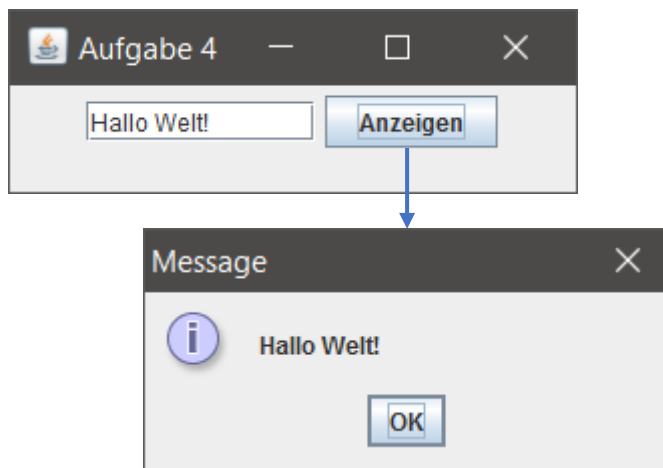
Gegeben sei die abgebildete grafische Benutzeroberfläche.

- Benennen Sie die wesentlichen Bestandteile einer Swing-Oberfläche (4 Punkte).
- Erstellen Sie eine ausführbare Klasse, die die abgebildete grafische Benutzeroberfläche realisiert (12 Punkte).

Hinweise

- Das Hauptfenster (Titel: Aufgabe 4) soll eine Größe von 300x100 besitzen!
- Das Eingabefeld soll eine Spaltenanzahl von 10 besitzen!
- Beim Betätigen der Drucktaste *Anzeigen* soll der Text des Eingabefeldes auf einem Nachrichtendialog angezeigt werden!
- Die Methode *setColumns(int)* der Klasse *JTextField* legt die Spaltenanzahl für ein Eingabefeld fest!
- Die statische Methode *.showMessageDialog(Component, String)* der Klasse *JOptionPane* erzeugt einen Nachrichtendialog!

Grafische Benutzeroberfläche



Aufgabe 5 (22 Punkte)

Gegeben sei die abgebildete Liste, sowie die beigefügte Implementierung des Quicksort-Verfahrens.

- Erläutern Sie kurz das Mergesort-Verfahren (4 Punkte) und benennen Sie das zugrundeliegende Prinzip (2 Punkte).
- Wenden Sie das Quicksort-Verfahren auf die Liste an (16 Punkte). 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 | 7 | | | | | | | | |
| 2 | 1 | | | | | | | | |
| 3 | 3 | | | | | | | | |
| 4 | 6 | | | | | | | | |
| 5 | 2 | | | | | | | | |
| 6 | 8 | | | | | | | | |
| 7 | 9 | | | | | | | | |
| 8 | 4 | | | | | | | | |

| Durchgang | l | r | c | i | j | l/j | i/r |
|-----------|---|---|---|---|---|-----|-----|
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | | |
| 6 | | | | | | | |
| 7 | | | | | | | |
| 8 | | | | | | | |