



Klausur

Objektorientierte Programmierung Praktikum

Studiengang

Wirtschaftsinformatik und E-Business (Bachelor) WI PLUS

Prof. Dr. Thomas Bayer
 Wintersemester 2018
 (Bitte in Druckbuchstaben ausfüllen!)

Matrikel-Nr:	Datum: 04.02.2019
Sitzplatz:	Semester: WS 2018
Hilfsmittel: keine	Bearbeitungszeit: 90 Minuten

(Bitte nicht ausfüllen!)

Unterschrift Prüfer:						Note:				
Aufgabe	CL	push	pop	grsc	forA	CL	push	pop	rem	Gesamt
max. Punkte	10	15	5	10	5	10	15	15	15	100
Punkte										

Ausgangssituation (Stack und ActivityStack)

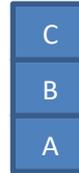
Zuerst soll ein Stack implementiert werden. Danach wird der Stack objektorientiert so erweitert, dass er Activities aufnehmen und ihren Lebenszyklus verwalten kann (s.u.)

Stack (Stapel): Last In First Out (LIFO)

- push(o): Das Objekt wird auf den Stapel gelegt (ganz oben!)
- pop: Das oberste Objekt wird vom Stapel genommen

Beispiel:

- push(A); push(B); push(C); führt zu
- pop() liefert C
- pop() liefert B
- pop() liefert A; der Stack ist jetzt leer



Index: Das unterste Element hat Index 0. Im Beispiel hat A Index 0, B Index 1, C Index 2

Interface

```
public interface ISimpleStack<T> {  
    // Nimmt ein Objekt auf, das nicht null ist und noch nicht im Stack enthalten  
    // ist. Rückgabe: Anzahl an Elemente im Stack nach der Aufnahme oder 0,  
    // falls t == null  
    // ist das Objekt bereits enthalten, wird es ganz oben auf den Stapel gelegt.  
    // Hinweis: Verwenden Sie entsprechende Operationen der Collection/des Array  
    // und nicht pop/push  
    int push(T t);  
  
    // Entfernt das oberste Objekt und gibt es zurück. Liefert null, falls der  
    // Stack leer ist  
    T pop();  
  
    // gibt einen Verweise auf das i-te Objekt zurück. Das Objekt bleibt im  
    // Stack. Falls i nicht existiert, führt dies zur IndexOutOfBoundsException.  
    // Das unterste Element hat Index 0, das oberste Index size()-1  
    T get(int i);  
  
    // Wie get(i), nur wird das Objekt aus dem Stack entfernt  
    T remove(int i);  
  
    // Anzahl an Objekte im Stack  
    int size();  
  
    // true, falls das Objekt im Stack enthalten ist  
    boolean contains(Object o);  
  
    // Wendet action.execute auf alle Objekte im Stack an  
    void forAll(IAction<T> action);  
}
```

ActivityStack:

Wie ein Stack, nur werden lediglich Instanzen der Klasse SimpleActivity aufgenommen (die Klasse besitzt keinen Typparameter). Zusätzlich ist der Lebenszyklus der Activity zu beachten:

- **push(SimpleActivity a)**: a wird auf den Stapel gelegt und aktiviert (a.activate()), sofern a nicht null ist. War der Stack nicht leer, wird die vorher aktive Activity (ganz oben) passiviert (a.passivate())
- **pop**: Die oberste Aktivität wird passiviert und zerstört (destroy()). Danach wird die Aktivität, die ganz oben liegt, aktiviert (sofern der Stack nicht leer ist)
- **remove(i)**: Die Aktivität an Position i wird entfernt und zerstört (destroy). Wenn die Aktivität ganz oben liegt, ist der Ablauf wie bei **pop**

Aufgaben

- Implementieren Sie zuerst den SimpleStack. Verwenden Sie entweder ein Array oder eine Liste (Array- oder LinkedList)
- Erweitern Sie die Klasse SimpleStack im Sinne der Objektorientierten Programmierung so, dass sie lediglich Instanzen von SimpleActivity aufnehmen kann und den Lebenszyklus wie oben beschrieben unterstützt. Code-Duplizierung bzw. Verwendung von Delegation führt zum Punkteabzug!!!
- Hinweis: Die Klasse SimpleActivityStack sollte kein Attribut enthalten.

Vorbereitung

1. Starten Sie Eclipse mit dem Workspace im Verzeichnis C:\InsightFiles
2. Erstellen Sie das Projekt 2018-WS-Testat-##### (##### = Ihre Matrikelnummer)
3. Importieren Sie das jar-File und JUnit5 (Projekt->Properties)
4. Erstellen Sie das Paket solution#####
5. Erstellen Sie die Klasse SimpleStackTest#####, die von ASimpleStackTest erbt
6. Bringen Sie die Unit-Tests zum Laufen

Vorgehen

- Implementieren Sie das Interface ISimpleStack mit der Klasse SimpleStack##### (##### = Ihre Matrikelnummer) – Nur die entsprechenden Methodenrumpfe!
- Passen Sie Ihre Testklasse so an, dass eine Instanz Ihrer Interfaceimplementierung übergeben wird
 - a. Methode getInstance() – Parameterloser Konstruktor
 - b. 20 Tests, 0 Errors, 15 Failures
- Erstellen Sie die Klasse SimpleActivityStack#####, die den Stack für SimpleActivity implementiert.
- Erstellen Sie die Klasse SimpleActivityStackTest#####, die von der Klasse ASimpleActivityStackTest erbt und Ihre Klasse instanziiert
- Programmieren Sie die Methoden aus
- Exportieren Sie Ihr Projekt als Archiv in das Verzeichnis C:\InsightFiles
- Vor Abgabe melden Sie sich!

Hinweise zur Benotung (100 Punkte)

- **Verwendung des eigenen Workspace (aus den Übungen): 0 Punkte**
- **Nachschlagen im eigenen Workspace (aus den Übungen): 0 Punkte**
- **(Versuchte) Manipulation der im jar-File enthaltenen Klassen/Interfaces: max. 25 Punkte**
- **Kopieren der im jar-File enthaltenen Klassen/Interfaces in ein eigenes Paket: 0 Punkte**
- **Programm nicht lauffähig (Syntaxfehler): max. 25 Punkte**
- **Code-Duplizierung (SimpleStack und SimpleActivityStack) – bis zu 55 Punkte Abzug!**
- **Delegation – bis zu 45 Punkte Abzug!**
- Sowohl Unit-Tests als auch der Quelltext werden für die Benotung herangezogen
- Unit-Tests dienen nur zur Orientierung!