

1 Stack

1.1 Einleitung

Ein Stack ist ein Stapelspeicher bzw. Kellerspeicher. Das Besondere am Stack ist, dass er für seine Funktionalität nur wenige Methoden braucht.

Um Elemente in den Stack zu legen wird die Methode *push()* benötigt. Dieses Element wird an vorderster bzw. oberster Stelle in die vorhandenen Daten eingebunden. Der Stack arbeitet nach dem LIFO-Verfahren. Die Abkürzung steht für „Last In First Out“ und das bedeutet, dass das zuletzt hinzugefügte Element auch als erstes wieder aus dem Stack herausgenommen werden muss.

Um dieses neueste Element aus dem Stack zu entfernen, wird die Methode *pop()* verwendet.

Außerdem wird noch mindestens eine Methode *size()* benötigt, die die Länge des bzw. die Anzahl der Elemente im Stack zurückgibt.

Sinnvoll ist vielleicht noch eine Methode *peek()*, die das oberste Element des Stacks zurückgibt, ohne dieses aus dem Stack zu entfernen.

In Java ist der Stack eine Klasse, die direkt von der Superklasse Vector erbt. Vector lässt sich über AbstractList und AbstractCollection auf die Klasse Object zurückführen. Damit stehen Stack auch sämtliche Methoden dieser Superklassen zur Verfügung.

1.2 Methodenbeschreibung

Im Folgenden werden die von Java für den Stack zur Verfügung gestellten Methoden beschrieben, die wichtig sind und in der Implementation verwendet wurden.

- *empty()*
 - prüft, ob der Stack Elemente enthält und liefert einen boolean-Wert zurück
- *peek()*
 - liefert das zuletzt zum Stack hinzugefügte Element zurück, ohne es aus dem Stack zu löschen
- *pop()*
 - gibt das zuletzt zum Stack hinzugefügte Element zurück und entfernt es aus dem Stack
- *push(E item)*
 - fügt das als Parameter übergebene Element dem Stack hinzu

- `size()`
 - gibt die Anzahl, der im Stack enthaltenen Elemente zurück

1.3 Effizienzanalyse

Graph einfügen und Effizienz ausrechnen. Für einzelne Probleme anwenden.

1.4 Schlussfolgerung

hier die resultierenden Stärken und Schwächen rausschreiben.