Einzelprüfung „Theoretische Informatik / Algorithmen / Datenstrukturen (nicht vertieft)“

Einzelprüfungsnummer 46115 / 2019 / Herbst

Thema 1 / Aufgabe 6

(*Mystery-Stacks*)

Stichwörter: Stapel (Stack)

Gegeben sei die Implementierung eines Stacks ganzer Zahlen mit folgender Schnittstelle:

```java
import java.util.Stack;

/**
 * Um schnell einen lauffähigen Stack zu bekommen, verwenden wir den Stack aus
 * der Java Collection.
 */
public class IntStack {
  private Stack<Integer> stack = new Stack<Integer>();

  /**
   * Legt Element i auf den Stack.
   *
   * @param i Eine Zahl, die auf dem Stack gelegt werden soll.
   */
  public void push(int i) {
    stack.push(i);
  }

  /**
   * Gibt oberstes Element vom Stack.
   *
   * @return Das oberste Element auf dem Stapel.
   */
  public int pop() {
    return stack.pop();
  }

  /**
   * Fragt ab, ob Stack leer ist.
   *
   * @return Wahr, wenn der Stapel leer ist.
   */
  public boolean isEmpty() {
    return stack.empty();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/mystery_stack/IntStack.java`

Betrachten Sie nun die Realisierung der folgenden Datenstruktur `Mystery`, die zwei Stacks benutzt.

```java
public class Mystery {
  private IntStack a = new IntStack();
  private IntStack b = new IntStack();
```

```java
public void foo(int item) {
  a.push(item);
}

public int bar() {
  if (b.isEmpty()) {
    while (!a.isEmpty()) {
      b.push(a.pop());
    }
  }
  return b.pop();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/mystery_stack/Mystery.java`

(a) Skizzieren Sie nach jedem Methodenaufruf der im folgenden angegebenen Befehlssequenz den Zustand der beiden Stacks eines Objekts `m` der Klasse `Mystery`. Geben Sie zudem bei jedem Aufruf der Methode `bar` an, welchen Wert diese zurückliefert.

```java
Mystery m = new Mystery();
m.foo(3);
m.foo(5);
m.foo(4);
m.bar();
m.foo(7);
m.bar();
m.foo(2);
m.bar();
m.bar();
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/mystery_stack/Mystery.java`

Lösungsvorschlag

Code	Stack b	Stack b	Rückgabewert
`m.foo(3);`	{ 3 }	{ }	
`m.foo(5);`	{ 5, 3 }	{ }	
`m.foo(4);`	{ 4, 5, 3 }	{ }	
`m.bar();`	{ }	{ 5, 4 }	3
`m.foo(7);`	{ 7 }	{ 5, 4 }	
`m.bar();`	{ 7 }	{ 4 }	5
`m.foo(2);`	{ 2, 7 }	{ }	
`m.bar();`	{ 2, 7 }	{ }	4
`m.bar();`	{ }	{ 2 }	7

(b) Sei n die Anzahl der in einem Objekt der Klasse `Mystery` gespeicherten Werte. Im folgenden wird gefragt, wieviele Aufrufe von Operationen der Klasse `IntStack` einzelne Aufrufe von Methoden der Klasse `Mystery` verursachen. Begründen Sie jeweils Ihre Antwort.

(i) Wie viele Aufrufe von Operationen der Klasse `IntStack` verursacht die Methode `foo(x)` im besten Fall?

Lösungsvorschlag

Einen Aufruf, nämlich `a.push(i)`

(ii) Wie viele Aufrufe von Operationen der Klasse `IntStack` verursacht die Methode `foo(x)` im schlechtesten Fall?

Lösungsvorschlag

Einen Aufruf, nämlich `a.push(i)`

(iii) Wie viele Aufrufe von Operationen der Klasse `IntStack` verursacht die Methode `bar()` im besten Fall?

Lösungsvorschlag

Wenn der Stack `b` nicht leer ist, dann werden zwei Aufrufe benötigt, nämlich `b.isEmpty()` und `b.pop()`

(iv) Wie viele Aufrufe von Operationen der Klasse `IntStack` verursacht die Methode `bar()` im schlechtesten Fall?

Lösungsvorschlag

Wenn der Stack `b` leer ist, dann liegen all n Objekte im Stack `a`. Die Objekte im Stack `a` werden in der `while`-Schleife nach `b` verschoben. Pro Objekt sind drei Aufrufe nötig, also $3 \cdot n$. `b.isEmpty()` (erste Zeile in der Methode) und `b.pop()` (letzte Zeile in der Methode) wird immer aufgerufen. Wenn alle Objekt von `a` nach `b` verschoben wurden, wird zusätzlich noch einmal in der Bedingung der `while`-Schleife `a.isEmpty()` aufgerufen. Im schlechtesten Fall werden also $3 \cdot n + 3$ Operationen der Klasse `IntStack` aufgerufen.

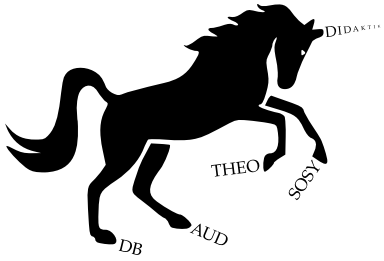
(c) Welche allgemeinen Eigenschaften werden durch die Methoden `foo` und `bar` realisiert? Unter welchem Namen ist diese Datenstruktur allgemein bekannt?

Lösungsvorschlag

`foo()` Legt das Objekt auf den Stack `a`. Das Objekt wird in die Warteschlange eingereiht. Die Methode müsste eigentlich `enqueue()` heißen.

`bar()` Verschiebt alle Objekte vom Stack `a` in umgekehrter Reihenfolge in den Stack `b`, aber nur dann, wenn Stack `b` leer ist. Entfernt dann den obersten Wert aus dem Stack `b` und gibt ihn zurück. Das zuerst eingereihte Objekt wird aus der Warteschlange entnommen. Die Methode müsste eigentlich `dequeue()` heißen.

Die Datenstruktur ist unter dem Namen Warteschlange oder Queue bekannt



Die Bschlangaul-Sammlung

Hermine Bschlangauland Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.Der TEX-Quelltext dieses Dokuments kann unter folgender URL aufgerufen werden: `https://github.com/bschlangaul-sammlung/examens-aufgaben/blob/main/Staatsexamen/46115/2019/09/Thema-1/Aufgabe-6.tex`