Prof. Dr. Stefan Brass
Dipl.-Inform. Annett Thüring
Dipl.-Inform. Steffen Schiele
Institut für Informatik
MLU Halle-Wittenberg

21. Januar 2013

Vorlesung "Objektorientierte Programmierung" — 2. Programmiertest (Aufgabe A) —

Hinweise/Regeln:

Vermeiden Sie bitte jedes Verhalten, was als Täuschungsversuch (miss-)verstanden werden könnte. Wir müssten Sie sonst disqualifizieren, d.h. Sie hätten den Programmiertest nicht bestanden. Sie werden sowohl durch das Aufsichtspersonal im Rechnerpool überwacht, als auch auf den Rechnern per Programm/Aufzeichnung, sowie über das Netz.

- Bearbeitungsdauer: 70 Minuten (plus eventuell leichte Verlängerung).
- Es gibt nur "bestanden" oder "nicht bestanden", keine Punkte für partiell korrekte Lösungen. Eventuell werden am Ende einige Minuten Bearbeitungszeit angehängt, wenn mehrere Teilnehmer knapp vor einer fertigen Lösung sind — aber nicht funktionierende Lösungen werden nicht akzeptiert, selbst wenn nur eine Kleinigkeit fehlt.
- Sie dürfen bis zu 3 Blätter "Spickzettel"/"Quick Reference" verwenden, sowie ein Buch (nicht zu groß, es muß noch auf den Tisch passen ohne den Nachbarn zu stören — ein Aktenordner wäre nicht akzeptabel). Sie dürfen außerdem ein Blatt leeres Papier und einen Stift verwenden, um sich Notizen zu machen.
- Mobiltelefone bitte ausschalten (oder mit der Aufsicht besprechen).
- Sie dürfen ein eigenes Notebook verwenden, wenn Sie es bisher in den Übungen benutzt haben. WLAN und sonstige Netzverbindungen müssen ausgeschaltet sein.
- Es ist nicht erlaubt, einen Web-Browser zu verwenden, ein EMail-Programm, oder sonstige Netzwerk-Zugriffe. Automatische Netzwerk-Zugriffe Ihrer Entwicklungsumgebung sind in Ordnung, aber ggf. geöffnete Webseiten schließen Sie bitte sofort.
- Die automatisch eingeblendeten Methoden-Signaturen und andere Hinweise gehören zur Entwicklungsumgebung, aber ansonsten verwenden Sie die Hilfe bitte nicht (insbesondere keine Suchfunktionen).
- Sie dürfen nicht versuchen, Dateien außerhalb Ihres Homeverzeichnisses abzulegen, oder auf Dateien außerhalb Ihres Homeverzeichnisses zuzugreifen (außer Dateien, die der Compiler bzw. die Entwicklungsumgebung verwendet).
- Auch auf vorhandene Dateien innerhalb Ihres Homeverzeichnisses dürfen Sie nicht zugreifen (außer Voreinstellungen etc. durch die Entwicklungsumgebung). Sie müssen daher das Programm vollständig neu eintippen, und nicht durch Modifikation von eventuell in Ihrem Homeverzeichnis bereits vorhandenen Dateien erstellen.
- Auch vorhandene Text-Dokumente (z.B. PDF) dürfen Sie nicht anzeigen.
- Die Homeverzeichnisse werden für Zugriffe von außen gesperrt. Falls Sie spezielle Zugriffsrechte gesetzt hatten, müssen Sie diese nach dem Test selbst wieder herstellen.

- Tauschen Sie keinesfalls irgendwelche Dinge mit den Nachbarn aus. Notfalls rufen Sie eine Aufsichtsperson zur Kontrolle.
- Sie müssen Mindestanforderungen an den Programmierstil erfüllen, z.B. entsprechend der Programmstruktur einrücken.
- Fragen Sie, wenn Ihnen die Aufgabe nicht klar ist! Wenn Sie an einer unverständlichen Fehlermeldung länger festhängen, können Sie probieren, zu fragen. Wir wollen aber nicht zu viele Tipps geben.
- Wenn Sie glauben, fertig zu sein, melden Sie sich bitte zur Kontrolle. Sollten wir noch einen Fehler finden, können Sie bis zur offiziellen Abgabefrist weiterarbeiten.

Aufgabe (Variante A)

Schreiben Sie eine Klasse `IntArray`, die Arrays von `int`-Werten mit beliebig wählbaren Indexbereich implementiert. Bei normalen Arrays beginnt der Indexbereich ja immer mit 0. Bei Ihrer Klasse soll man eine untere Grenze `u` und eine obere Grenze `o` für die möglichen Index-Werte angeben können (beide Grenzen sind "inklusive" zu verstehen, d.h. beide sollen gerade noch gültige Indexwerte sein). Wenn man z.B. sagt, dass man den Indexbereich `5` bis `10` will, könnte das Array z.B. so aussehen:

Index	Inhalt
5	50
6	0
7	70
8	0
9	0
10	100

(Die im Array gespeicherten Werte sind willkürlich. Dieser Zustand würde sich nach den Beispiel-Aufrufen im Testprogramm auf der nächsten Seite ergeben.)

Ihre Klasse soll folgende Schnittstelle anbieten:

- Einen Konstruktor mit Parametern für untere und obere Grenze (`u` und `o`), beide vom Typ `int`. Wie in Java üblich, sollen alle Array-Einträge automatisch auf den Wert `0` initialisiert sein. Den Fall ungültiger Parameter (`u > o`) brauchen Sie hier nicht zu behandeln.
- Eine Methode `store` zum Speichern von Werten in das Array. Die Methode hat zwei Parameter, den ersten für den Index, und den zweiten für den zu speichernden Wert. Beide sind vom Typ `int`. Die Methode liefert keinen Wert zurück. Den Fall ungültiger Indexwerte (außerhalb des Bereiches von `u` bis `o`) brauchen Sie nicht zu behandeln.
- Eine Methode `retrieve` zum Abfragen eines Wertes aus dem Array: Die Methode hat einen `int`-Parameter für den Index, und liefert den an dieser Stelle im Array gespeicherten Wert zurück (ebenfalls ein `int`). Falls der Index ungültig ist (d.h. kleiner als `u` oder größer als `o`), soll `-1` geliefert werden.

- Eine Methode `print` zur Ausgabe des Arrays mit einer Zeile pro Indexwert, jeweils von der Form `Index: Wert`. Im obigen Beispiel wäre die erste Zeile also "`5: 50`" und die letzte "`10: 100`". Diese Methode hat keine Parameter und gibt nichts zurück.

Achten Sie darauf, dass nur die erwähnten Methoden von außen zugreifbar sind, und zwar auch von außerhalb des Paketes. Attribute dürfen nicht von außen zugreifbar sein.

Eine mögliche Anwendung Ihrer Klasse sieht so aus:

Listing 1: Hauptprogramm

```
1  public class IntArrayTest
2  {
3      public static void main(String[] args)
4      {
5          IntArray a = new IntArray(5, 10);
6          a.store(10, 100);
7          a.store( 5,  50);
8          a.store( 7,  70);
9          a.print();
10         System.out.println(a.retrieve(10)); // 100
11         System.out.println(a.retrieve( 9)); //   0
12         System.out.println(a.retrieve( 3)); //  -1
13     }
14 }
```

Tipps:

- Natürlich merken Sie sich die beiden Grenzen in Attributen Ihrer Klasse.
- Außerdem brauchen Sie ein Array `arr` der Größe `o-u+1`.
- Wenn Sie auf den Index `i` zugreifen sollen, greifen Sie tatsächlich auf `i-u` zu. Im Beispiel (Indexbereich von `5` bis `10`) würde also `a.store(5, 50)` den Wert `50` an `arr[0]` speichern. Entsprechend würde `a.store(7, 70)` den Wert `70` an `arr[2]` speichern.
- Diese Umrechung des Indexbereiches ist die wesentliche Aufgabe der Klasse. An der Schnittstelle tut sie so, als wäre der Indexbereich z.B. `5` bis `10` (allgemein `u` bis `o`). Bei dem zur Implementierung verwendeten Java-Array beginnt der Indexbereich aber bei `0`, deswegen werden alle Index-Angaben um `5` (allgemein `u`) verringert.