Programmierung für Mathematik HS12
Übung 6

29. Oktober 2012

1 Aufgabe: Arrays

1.1 Ziele

- Eine Klasse ändern und dabei die Schnittstellen beibehalten.
- Arrays vertiefen.

1.2 Aufgabenstellung

In der letzten Übung haben Sie eine Klasse `MyList` geschrieben. Das Array in dieser Liste nutzt den Speicher optimal, in dem Sinn, dass genau nur soviel Speicher belegt wird wie auch tatsächlich gebraucht wird. Allerdings muss dafür die Grösse des Arrays immer geändert werden, wenn neue Werte hinzugefügt oder gelöscht werden. Ziel dieser Übung ist die Klasse `MyList` umzugestalten, so dass es nicht mehr notwendig ist, das Array bei jeder Änderung anzupassen.

Implementieren Sie die Klasse `MyList` neu. Im Vergleich zu Ihrer ersten Implementation soll sie zusätzlich zum `int[] array` eine Instanzvariable besitzen, die angibt wieviele Werte im Array gespeichert sind. Beginnen Sie damit die setter Methode anzupassen. Implementieren Sie die weiteren Methoden folgendermassen:

- `add` : Diese Methode soll, falls das interne Array gross genug ist, den Parameter an letzter Stelle hinzufügen. Falls die Grösse des Arrays nicht ausreicht, so soll das Array mit einem anderthalb Mal so grossen Array ersetzt werden.
- `erase` : Diese Methode dekrimiert die Instanzvariable, die die Anzahl der gespeicherten Werte speichert um 1. Falls der Wert dieser Variable danach kleiner ist, als die Hälfte der Grösse des internen Arrays, so soll das Array mit einem halb so grossen Array ersetzt werden.

- `print` : Diese Methode gibt die Werte des Arrays der Reihe nach in der Konsole aus.

Schreiben Sie auch die Methoden mit der folgenden Signatur:

```
public void add(int position,int a)
```
Diese Methode haben Sie bereits in Übung 5 implementiert. Passen Sie die Methode der neuen Aufgabenstellung an.

```
public void erase(int position)
```
Diese Methode ermöglicht es, einen bestimmten Wert des Arrays zu löschen, so dass alle Werte ab dieser Stelle um 1 zurück rücken.
Benutzen Sie denselben `TestDriver` um Ihre Implementierung zu testen. Erweitern Sie anschliessend den `TestDriver` mit einem Aufruf der neu implementieren `erase` Methode um auch diese Implementierung zu testen.

2 Aufgabe: Mehrdimensionale Arrays verstehen

2.1 Ziele

- Multi-dimenionale Arrays verstehen.

2.2 Aufgabenstellung

2.2.1 Code verstehen

Betrachten Sie folgende `main` Methode. Überlegen Sie sich wie Sie den Code vor dem ersten Aufruf von `System.out.println()` anpassen müssen, damit die Ausgabe einer XOR Warheitstabelle entspricht.

`main` Methode

```
public static void main(String[] args){

    boolean[][] warheitsTafelXor = new boolean[4][3];

    warheitsTafelXor[1][1]=warheitsTafelXor[2][0]=true;
    warheitsTafelXor[3][0]=warheitsTafelXor[3][1]=true;
    for(int i=0;i<4;i++){
        warheitsTafelXor[i][2]= true;
    }

    //System Output
    System.out.println("  p   |   q   | p xor q");
    System.out.println("-------------------------");
    for(int i=0;i<4;i++){
        System.out.println(
        (warheitsTafelXor[i][0] ? "true  | ": "false | ")
        +(warheitsTafelXor[i][1] ? "true  | ": "false | ")
        +warheitsTafelXor[i][2]);
    }
}
```

Betrachten Sie nun folgende `main` Methode und entscheiden Sie sich wieviele `int` Werte am Ende im Array gespeichert sind. Versuchen Sie auch die Ausdehnung in den verschiedenen Dimensionen herauszufinden.

main Methode

```
public static void main(String[] args){

    int[][][] a= new int[10][][];

    for(int i=0;i<10;i++){
        a[i]= new int[i+2][];
        for(int j=0;j<i+2;j++){
            a[i][j]=new int[(2*i)-j+1];
        }
    }
}
```

3 Aufgabe: Matrix Klasse für beliebige Grösse implementieren

3.1 Ziele

- Multi-dimenionale Arrays benutzen können.

3.2 Aufgabenstellung

Schreiben Sie eine neue `Matrix` Klasse (vgl. Übung 5). Diese soll beim Aufruf der Methode mit folgender Signatur: `public void set(int rows,int columns)` intern ein entsprechendes Array speichern. Die Matrix soll zu Beginn der null Matrix entsprechen. Schreiben Sie getter Methoden für die Variablen `rows` und `columns` . Erweitern Sie diese Klasse um folgende Methoden:

- `setValue(int row, int colum, int value)` : die, den entsprechenden Eintrag der Matrix auf `value` setzt.
- `public Matrix add(Matrix b)` : die, falls die beiden Dimensionen der Matrizen übereinstimmen die Summe der beiden returniert.
- `public Matrix mult(Matrix b)` : die, falls die Dimensionen der beiden Matrizen es erlauben, das Matrix Produkt der beiden Matrizen returniert.

Die Implementierung der Methoden `add, mult` sollen die Matrizen nicht geändert werden. Wir eine Methode mit unzulässigen Parametern aufgerufen, wird eine entsprechende Meldung auf der Konsole ausgegeben und `null` returniert.

Schreiben Sie noch zwei weitere Methoden die die Initialisierung der Matrizen erleichtern sollen:

- `public Matrix identity(int n)` die die Matrix, die in der Instanzvariable gespeichert ist, auf die Identität der Dimension n setzt und diese gleich returniert.
- `public Matrix scalMult(int r)` die, die Matrix, die in der Instanzvariable gespeichert ist, mit `r` multipliziert returniert.