technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 1
Dr. Lars Hildebrand
Iman Kamehkhosh, Marcel Preuß, Henning Timm

Praktikum zu

Einführung in die Informatik für LogWings und WiMas

Wintersemester 2013/14

Übungsblatt 9

Bearbeitungszeit:
13.–17.01.2014

Mit Lösungen

Ergänzende Aufgaben

Aufgabe 9.4 – Eine weitere Klasse

a) Schreiben Sie eine zweite Klasse mit dem Namen „Bank“ und mit den Attributen Name, Bankleitzahl und Ort der Bank und einer Methode `void print()`, die die Daten ausgibt.

```
class Bank{
  String name;
  long blz;
  String ort;

  void print(){
    System.out.println("Name der Bank: "+ name);
    System.out.println("BLZ: "+ blz);
    System.out.println("Ort: "+ ort);
  }
}
```

b) Erweitern Sie Ihre Klasse „Kreditkarte“ um das Attribut Bank, sodass bei dem Konstruktor ein Objekt des Typs `Bank` übergeben wird und von der Methode `void print()` auch die Bankdaten ausgegeben werden.

```
class Kreditkarte {
   String inhaberName ;
   long nummer ;
   boolean gesperrt ;
   Bank bank;

   Kreditkarte( String name , long nummer , Bank bank) {
      this.inhaberName = name ;
      this.nummer = nummer ;
      this.gesperrt = false ;
      this.bank = bank;
   }

   void print(){
      System.out.println("Inhaber: "+inhaberName);
      System.out.println("Kartennr.: "+nummer);
      System.out.println("gesperrt?  "+gesperrt);
      bank.print();
   }
   .....
}
```

c) Testen Sie Ihr Programm, indem Sie in der `main`-Methode der Klasse „TestKreditkarte“ zunächst ein Objekt des Typs `Bank` erzeugen und dies zusätzlich dem Konstruktor `Kreditkarte(...)` als Parameter übergeben.

```
class TestKreditkarte {

  public static void main ( String [] args ) {

    Bank bank = new Bank();
    bank.name = "MusterBank";
    bank.blz = 12345;
    bank.ort = "Musterstadt";

    Kreditkarte karte ;
    karte = new Kreditkarte ("Petra Mustermann" , 1234567, bank);
    karte.setzeSperre(true);
    karte.print();

    if( karte.gesperrt ){
       System.out.println("Die Karte ist gesperrt .");
    }
    else {
       System.out.println("Die Karte ist nicht gesperrt .");
    }

    if( karte.istGesperrt() ){
       System.out.println("Die Karte ist gesperrt .");
    }
    else {
       System.out.println("Die Karte ist nicht gesperrt .");
    }
  }

}
```

Aufgabe 9.5 – Noch mal Arrays: Sieb des Eratosthenes

Um alle Primzahlen bis zu einer bestimmten Größe zu finden, gibt es ein Verfahren namens *Sieb des Eratosthenes*, das Sie eventuell schon aus der Schule kennen. Angenommen, alle Primzahlen bis zur Größe n sollen bestimmt werden, dann werden zunächst alle Zahlen von eins bis n aufgeschrieben. Danach werden alle Zahlen weggestrichen, die durch zwei teilbar sind, außer der zwei selbst, also $4, 6, 8$ usw. Nun wird mit der nächsten Zahl fortgefahren, die noch nicht gestrichen ist, in diesem Fall der drei. Es werden also alle Vielfachen dieser Zahl weggestrichen, also $6, 9, 12, \ldots$. Dieses Vorgehen wird solange wiederholt, bis keine weiteren Zahlen mehr gestrichen werden können.

Implementieren Sie eine Methode `siebDesEratosthenes`, die als Parameter eine ganze Zahl n erhält und ein Array von Booleans der Länge $n + 1$ zurückliefert, das für jede Zahl Auskunft gibt, ob es sich um eine Primzahl handelt.

Tipp: Legen Sie in Ihrer Methode als erstes ein Array von Booleans der Länge $n + 1$ an und setzen Sie alle Elemente des Arrays auf `true`.

```
class Eratosthenes{

  public static void main(String[] args){
    int n = 100;
    boolean[] primzahlen;
    primzahlen = siebDesEratosthenes(n);

     //*** Ausgabe ***
    System.out.println("Primzahlen von 2 bis "+n+":");
    for(int i=0; i<primzahlen.length; i++){
       if(primzahlen[i]==true){
          System.out.print(i+", ");
       }
    }

  }

  public static boolean[] siebDesEratosthenes(int n){
    boolean[] sieb = new boolean[n+1]; // n+1, wegen Idex 0

    // Array ab Index 2(erste Primzahl) mit true vorbelegen
    for(int i=2; i<sieb.length; i++){
      sieb[i]=true;
    }

    for(int i=2; i<sieb.length; i++){
      for(int j=i+1; j<sieb.length; j++){
        if( j % i == 0)    // wenn j ganzzahlig durch i teilbar,
          sieb[j] = false; // dann markiere j als "keine Primzahl"
      }
    }
    return sieb;
  }
}
```

Aufgabe 9.6 – Pascalsches Dreieck

Beim Ausmultiplizieren von Binomen (Terme der Form $(x + y)^n$) treten Koeffizienten auf, die sich nach einem einfachen Schema berechnen lassen. Zum Beispiel erhält man für $n = 3$ folgendes: $(x + y)^3 = x^3 + 3x^2y^1 + 3xy^2 + y^3$. Die Koeffizienten lauten in diesem Fall $1, 3, 3, 1$; sie heißen *Binomialkoeffizienten*. Schreibt man die Koeffizienten, die sich für ein bestimmtes n ergeben, in eine Zeile und notiert alle Zeilen bis zu einem bestimmten n, so erhält man ein Pascalsches Dreieck. Für $n = 6$ sieht es so aus:

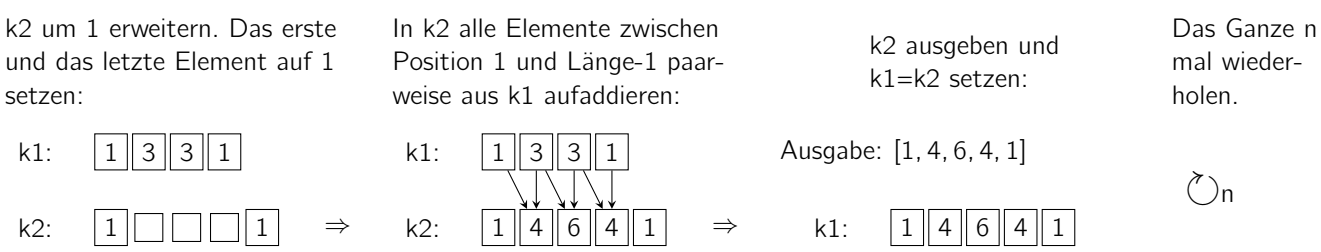
```
n
0                          1
1                      1       1
2                  1       2       1
3              1       3       3       1
4          1       4       6       4       1
5      1       5       10      10      5       1
6  1       6       15      20      15      6       1
```

Schreiben Sie nun eine Methode `pascalschesDreieck`, die einen ganzzahligen Parameter n hat und ein Pascalsches Dreieck bis zur Zeile n ausgibt. Auf eine schöne grafische Darstellung kommt es hier nicht an, daher reicht es, wenn die Ausgabe für $n = 5$ ungefähr so aussieht:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

Es gibt mehrere Möglichkeiten, diese Aufgabe zu lösen. Sie können zum Beispiel für jede Zeile des Dreiecks ein neues Array anlegen und die jeweils vorherige Zeile benutzen, um die aktuelle Zeile zu füllen. Sie können auch von Anfang an nur ein Array anlegen, das bereits groß genug ist, um alle Koeffizienten der letzten Zeile aufzunehmen, und dieses Array immer wieder überschreiben.

Bildlich:




```
class BinomKoeff{
  public static void main(String[] args){
    int n = 5;
    pascalschesDreieck(n);
  }

  public static void pascalschesDreieck(int n){
    int[] k1 = new int[1];
    k1[0] = 1;
    System.out.println("[1]");
    for(int i=0; i<n; i++){
       int[] k2 = new int[k1.length+1];
       k2[0]=1; k2[k2.length-1]=1;
       for(int j=1; j<k2.length-1; j++){
          k2[j] = k1[j-1] + k1[j];
       }
      System.out.println(java.util.Arrays.toString(k2));
      k1 = k2;
} } }
```