EPROG 3. Teilprüfung

Dauer 70min. Keine Unterlagen erlaubt. Loginname ist q<matrikelnummer> (also z.B. q0797801). Passwort ist Ihre Matrikelnummer.

Speichern Sie Ihre Lösungen in den dafür vorgesehenen Unterverzeichnissen ~/test/Beispiel1/ und ~/test/Beispiel2/.

Die Java-Dokumentation finden Sie im Verzeichnis ~/test/doc/

Aufgabe 1:

Schreiben Sie eine Methode, die ein Pascalsches Dreieck auf dem Bildschirm ausgibt. Ein Pascalsches Dreieck besteht aus Koeffizienten $c(n, k)$, wobei n der Zeilenindex und k der Spaltenindex ist (genauer gesagt sind $c(n, k)$ Binomialkoeffizienten). Ein Pascalsches Dreieck sieht wie folgt aus:

	k:0	1	2	3	4
n:0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
4	1	4	6	4	1

Dabei wird ein Koeffizient (Eintrag im Dreieck) als Summe des Koeffizienten in der selben Spalte der Vorgängerzeile und dessen linken Nachbarn gebildet, also $c(n, k) = c(n-1, k-1) + c(n-1, k)$ (siehe obige Darstellung).

Vervollständigen Sie die Datei
~/test/Beispiel1/Pascal.java

Die Klasse Pascal besitzt 3 Methoden:

- `void pt(int n)` zeichnet das Pascalsche Dreieck mit Zeilen 0 bis n , und greift dabei auf folgende Methode zurück:
- `void pline(int n)` zeichnet die Zeile n des Pascal'schen Dreiecks.
- `int c(int n, int k)` berechnet den Koeffizienten an der Stelle (n, k) . Diese Methode soll rekursiv programmiert werden (obige Formel).

Hinweis:

Vorbedingungen der gefragten Methoden müssen nicht überprüft werden, es sei denn es wird in der Angabe explizit verlangt. Die vorgefertigte Datei `Main.java` enthält die Definition der ausführbaren Klasse, die Sie zum Testen Ihrer Klassen benutzen können. Ihr Inhalt wird nicht beurteilt.

Aufgabe 2:

Erstellen Sie ein System, das eine Liste von komplexen Zahlen verwaltet (speichert). Es soll damit möglich sein, die darin enthaltenen Zahlen nach verschiedenen Kriterien sortiert ausgeben zu lassen. Das System besteht aus den Klassen `Complex`, `ComplexDistanceSorter`, `ComplexPhaseSorter` und `ComplexList`.

Für diese Aufgabe ist es notwendig, eine komplexe Zahl in kartesischer Darstellung ($real, img$) in Polarkoordinatendarstellung (r, ϕ) umzuwandeln, wobei r der Betrag (*distance*) und ϕ der Winkel (*phase*) der komplexen Zahl ist. Als Datentyp für die Komponenten soll `double` verwendet werden.

Es gilt

$$r = \sqrt{real^2 + img^2}$$
$$\phi = \arctan\left(\frac{img}{real}\right),$$

wobei hier Sonderfälle beachtet werden müssen. Zur Vereinfachung stellt Java die Methode `Math.atan2(double, double)` zur Verfügung, die ϕ aus $real$ und img berechnet und dabei alle diese Sonderfälle berücksichtigt.

Erzeugen Sie die Dateien `Complex.java`, `ComplexDistanceSorter.java`, `ComplexPhaseSorter.java` und `ComplexList.java`, die folgende Klassendefinitionen enthalten:

Complex:

Die Klasse `Complex` repräsentiert eine komplexe Zahl in kartesischer Darstellung (Realteil und Imaginärteil). Die Datenelemente sollen mit Auskunftsmethoden (Getter-Methoden) abgerufen werden können. Die Methode `toString()` soll die Zahl in der Form " $real;img$ " als String zurückliefern.

ComplexList:

Die Klasse `ComplexList` repräsentiert eine Menge (genauer gesagt eine Liste), zu der komplexen Zahlen hinzugefügt werden können (`ComplexList` speichert beliebig viele komplexe Zahlen).

Die Methode

```
add(Complex)
```

fügt neue komplexe Zahlen hinzu.

Die Methoden

```
public void list()
```

```
public void listDistance()
```

```
public void listPhase()
```

sollen die Zahlen unsortiert bzw. sortiert nach dem Betrag und sortiert nach dem Winkel ausgegeben. Damit die Zahlen sortiert werden können, sollen 2 Klassen geschrieben werden, die die Ordnung der Zahlen nach 2 verschiedenen Kriterien festlegen (siehe unten). Die Sortierung selbst kann dann einfach mittels der Methode `java.util.Collections.sort()` durchgeführt werden (es ist daher nicht notwendig einen Sortieralgorithmus zu entwerfen).

ComplexDistanceSorter:

Die Klasse `ComplexDistanceSorter` soll das Interface `Comparator<Complex>` implementieren. Siehe dazu die java Dokumentation der Schnittstelle `java.util.Comparator`. Die einzige Methode, die dabei implementiert werden muss ist `int compare(Complex, Complex)`, die die Beträge der komplexen Zahlen in Polardarstellung vergleicht.

ComplexPhaseSorter:

Die Klasse `ComplexPhaseSorter` soll das Interface `Comparator<Complex>` implementieren. Siehe dazu die java Documentation der Schnittstelle `java.util.Comparator`. Die einzige Methode die dabei implementiert werden muss ist `int compare(Complex,Complex)`, die den Winkel der komplexen Zahlen in Polardarstellung vergleicht.

Hinweis:

Vorbedingungen der gefragten Methoden müssen nicht überprüft werden, es sei denn es wird in der Angabe explizit verlangt. Die vorgefertigte Datei `Main.java` enthält die Definition der ausführbaren Klasse, die Sie zum Testen Ihrer Klassen benutzen können. Ihr Inhalt wird nicht beurteilt.

TRINKT APFELSAFT!!!!!!!!11111einseins