5. Praktische Übung in "Programmieren": Klasse Complex

Es soll ein einfaches **Java-Projekt** realisiert werden. Der Projektname ist frei wählbar.
Inhalt des Projekts ist die Definition und der Test einer Klasse für komplexe Zahlen.
Das Projekt soll aus zwei Klassen bestehen :

- **`Complex`** : Diese Klasse dient zur Darstellung und Bearbeitung komplexer Zahlen
- **`ComplDemo`** : Diese Klasse dient zur Demonstration und zum Test der Klasse `Complex`.

Beide Klassen sollen sich in einem Package mit dem Namen `prakt.v5` befinden, aber auch außerhalb dieses Packages verwendbar sein.

- **Klasse `Complex`**

  ◇ Objekte der **nicht ableitbaren** Klasse **`Complex`** speichern **Realteil** und **Imaginärteil** einer komplexen Zahl in **privaten nicht änderbaren Datenkomponenten** vom Typ **`double`.**
  Zwei weitere ebenfalls **nicht änderbare private `double`-Komponenten** enthalten die Polarkordinaten (**Betrag** und **Phase** im Bogenmaß) der komplexen Zahl.

  ◇ Realteil und Imaginärteil der komplexen Zahl sind dem **Konstruktor** als **Parameter** zu übergeben.
  Im Konstruktor sind alle Datenkomponenten zu setzen.
  Zur Ermittlung von Betrag und Phase lassen sich die **statischen** Methoden **`sqrt()`** und **`atan2()`** der Klasse **`Math`** (package `java.lang`) einsetzen.
  Zusätzlich zu diesem Konstruktor besitzt die Klasse einen **parameterlosen Konstruktor**, der Realteil und Imaginärteil auf **`0.0`** setzt.

  ◇ Weiterhin **überschreibt** die Klasse `Complex` die folgenden von der Klasse **`Object` geerbten Methoden** in geeigneter Weise :

  ▻ **`toString()`** : Erzeugung einer Stringdarstellung der enthaltenen komplexen Zahl in folgendem Format : `(2.345678 + j* 0.975310)`
  Real- und Imaginärteil werden jeweils mit 6 Nachpunktstellen dargestellt.

  ▻ **`equals()`** : Rückgabe von `true`, wenn das als Parameter übergebene Objekt ein `Complex`-Objekt ist und in Real- und Imaginärteil mit dem aktuellen Objekt übereinstimmt, andernfalls Rückgabe von `false`.

  ▻ **`hashCode()`** : Erzeugung eines sinnvollen Hash-Codes (Rueckgabewert).

  ▻ **`clone()`** : Erzeugung eines neuen `Complex`-Objekts, das dieselben Datenkomponenten wie das aktuelle Objekt besitzt (also dieselbe komplexe Zahl darstellt).

  ◇ Desweiteren definiert die Klasse `Complex` eine Instanz-Methode

  ▻ **`toPolarStr()`:** Erzeugung eines Strings (Funktionswert), der die **Polarkoordinatendarstellung** der enthaltenen komplexen Zahl in folgendem Format enthält :
  `2.540361*exp(1.176752*j)`
  Betrag und Phase werden ebenfalls jeweils mit 6 Nachpunktstellen dargestellt.

a) **Realisieren** Sie die Klasse `Complex` mit **allen o.a. Komponenten**.

Ergänzen Sie die Klasse um eine geeignet formulierte statische Methode **`main()`,** mit der die Klasse in ihrer bisherigen Funktionalität **getestet** werden kann.
In der Methode sind drei `Complex`-Objekte anzulegen, die die komplexen Zahlen `2.5-j*0.7` , `0.5+j*1.0` und `1.25-j*0.35` darstellen.
Ein viertes `Complex`-Objekt ist durch Klonen des ersten Objekts zu erzeugen.
Alle vier Objekte sind in beiden Darstellungsformen in jeweils einer Zeile in die Standardausgabe auszugeben
Das vierte Objekt ist mit den anderen drei Objekten zu vergleichen, das Vergleichsergebnis ist auszugeben.
Der jeweilige Hashcode der vier Objekte ist auszugeben.

weiter Seite 2

◇ Um die Realisierung von **Rechenoperationen** mit komplexen Zahlen zu ermöglichen, muss die Klasse `Complex` um **geeignete Memberfunktionen** ergänzt werden.

b) **Ergänzen** Sie die o.a. Klassendefinition um die Instanz-Methoden **`add()`**, **`sub()`**, **`mult()`** und **`div()`**. Allen Methoden wird der zweite Operand der jeweiligen Operation als Parameter übergeben. Der Rückgabewert jeder Funktion ist das Ergebnis der von ihr realisierten Operation.

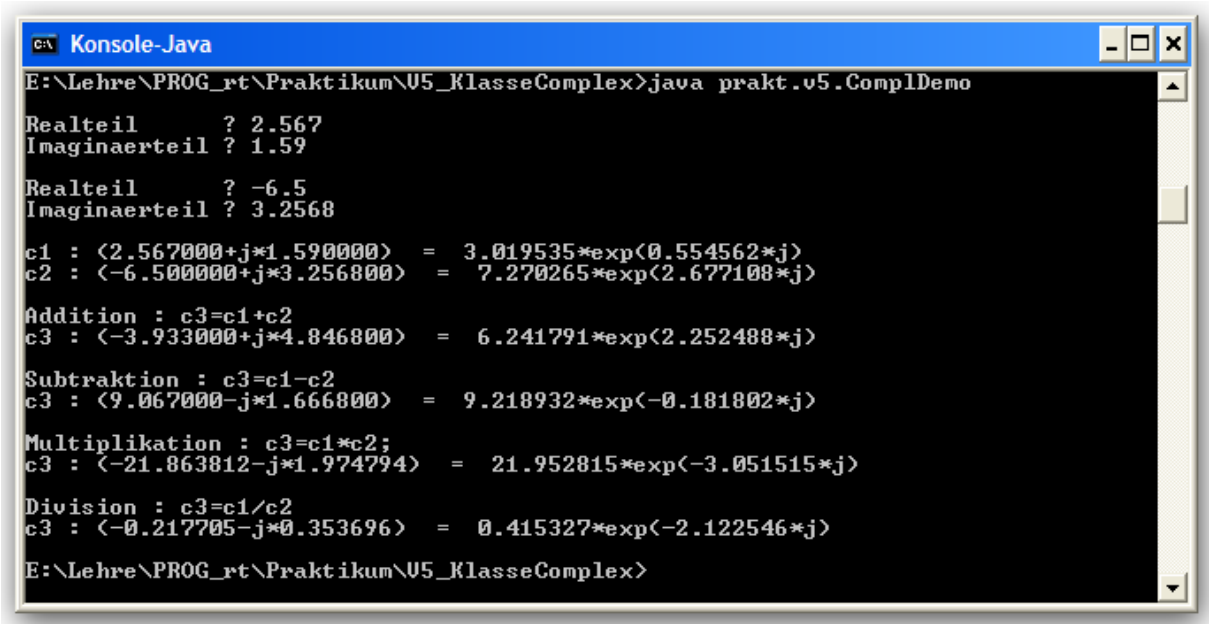
- **Klasse `ComplDemo`**

◇ Diese Klasse soll lediglich über **zwei statische Methoden** verfügen :

▻ **`liesComplex()`** : **Einlesen** (nach entsprechender **Eingabeaufforderung**) von **Realteil** und **Imaginärteil** einer **komplexen Zahl** aus der Standardeingabe.
Das damit gebildete **`Complex`–Objekt** wird als **Funktionswert** zurückgegeben.

▻ **`main()`** : **Demonstration** und **Test** der Klasse **`Complex`:**
- Einlesen zweier komplexer Zahlen von der Standard-Eingabe.
- Ausgabe beider Zahlen in beiden Darstellungsformen in die Standardausgabe.
- Anwendung der vier Grund-Rechenoperationen auf die beiden Zahlen.
- Ausgabe des jeweiligen Operationsergebnisses in beiden Darstellungsformen in die Standardausgabe.

Die von der `main()`–Funktion der Klasse `ComplDemo` erzeugte Konsolen-Ausgabe soll dem folgenden Beispiel entsprechen :



c) **Realisieren** Sie die Klasse **`ComplDemo`.**

- **Modifikation:**

◇ Die in der Klasse `ComplDemo` enthaltene Methode **`liesComplex()`** könnte auch als **statische Methode** der Klasse **`Complex`** definiert werden.

d) **Überlegen** Sie, wo diese Methode sinnvoller aufgehoben ist.
Begründen Sie Ihre Antwort.

e) **Modifizieren** Sie beide Klassen so, dass die Methode **`liesComplex()`** von der Klasse `ComplDemo` in die Klasse **`Complex`** **verschoben** wird.
Die Funktionalität der `main()`–Methode der Klasse `ComplDemo` soll sich nicht ändern.