Inleiding Programmeren

Practicum 3

Deadline: zaterdag 19 september 2015, 23:00.

Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

1 Introductie

Het doel van deze opgave is ervaring opdoen met het definiëren en het gebruik van klassen met bijbehorende attributen. De opgave bestaat uit drie onderdelen. Als eerste moet je de *Breuk* klasse implementeren. Als tweede moet je de klasse *ComplexGetal* implementeren voor complexe getallen van de vorm $a + bi$ met a en b reële getallen (zie `https://staff.fnwi.uva.nl/j.vandecraats/#cg`). En als laatste moet je de klasse *ComplexGetal* zo aanpassen dat a en b breuken zijn.

2 Klasse Breuk

Een rationaal getal heeft de vorm a/b met a de teller en b de noemer. Bijvoorbeeld $1/2$, $3/5$ en $8/2$ zijn rationale getallen. De noemer van een rationaal getal kan niet gelijk zijn aan 0, maar de teller kan wel 0 zijn.
Een rationaal getal kan niet altijd exact gerepresenteerd worden in floating-point formaat,
bijvoorbeeld $1/3 = 0.33333\ldots$.
Java levert datatypen voor integers en doubles, maar niet voor rationale getallen. Daarom wordt jullie gevraagd een klasse Breuk te ontwerpen, waarmee rationale getallen als Breuk object kunnen worden opgeslagen. Voor het representeren van rationale getallen gebruiken we een teller en een noemer. De teller kan alle gehele getallen bevatten, en de noemer moet een geheel getal groter dan 0 zijn.
Schrijf een klasse Breuk in de file **Breuk.java** met de volgende attributen:

- Twee integer instantiatie-variabelen *teller* en *noemer*.
- Vier verschillende constructoren, elk met verschillende parameters. Let op dat in de constructor met 2 parameters getest moet worden of de noemer groter is dan 0. Als de noemer kleiner is dan nul, moeten teller en noemer met -1 vermenigvuldigd. worden. Verder moet de breuk opgeslagen worden in vereenvoudigde vorm, d.w.z. als we als teller 2 opgeven en als noemer 12, willen we de breuk $1/6$ opslaan.

```
class Breuk implements BreukInterface {
    int teller, noemer;

    Breuk(int t, int n) { // breuk met n > 0
        ...               // opslaan in vereenvoudigde vorm
                          // voorbeeld breuk 2/12, ggd(2,12) = 2
                          // sla op 2/ggd en 12/ggd in teller en noemer
    }

    Breuk(int t) { // breuk met noemer gelijk aan 1
        ...        // roep hier constructor met 2 parameters aan
    }

    Breuk() {      // breuk met teller gelijk aan 0
        ...        // roep hier constructor met 2 parameters aan
    }
    Breuk(Breuk original) {
        ...
    }
}
```

- Op breuken wil je de volgende operaties kunnen doen: *optellen*, *aftrekken*, *vermenigvuldigen* en *delen*. Voor optellen en aftrekken moet je eerst gelijknamig maken en dan de breuk vereenvoudigen door te delen door de grootste gemene deler. Delen is vermenigvuldigen met het omgekeerde (delen door a/b is vermenigvuldigen met b/a). Daarom willen we ook een methode die de omgekeerde breuk berekent.

  Om afspraken te maken hoe deze methoden aangeroepen moeten worden, gebruiken we de interface *BreukInterface.java*, waarin alleen de header van de methoden gegeven is. In de Breuk klasse zorgt *implements BreukInterface* ervoor dat je precies die signatuur voor de methoden gebruikt. Creëer de file BreukInterface.java met daarin volgende methode-headers:

```
interface BreukInterface {
    public Breuk telop(Breuk breuk);
    public Breuk trekaf(Breuk breuk);
    public Breuk vermenigvuldig(Breuk breuk);
    public Breuk deel(Breuk breuk);
    public Breuk omgekeerde();
}
```

  Als voorbeeld geven we de implementatie van de methode *vermenigvuldig* die twee breuken vermenigvuldigt:

```
public Breuk vermenigvuldig(Breuk b2) {
    int nieuweteller = teller * b2.teller;
    int nieuwenoemer = noemer * b2.noemer;
    Breuk r = new Breuk(nieuweteller, nieuwenoemer);
    return r;
}
```

  Let op dat elke methode public moet zijn.

  Implementeer verder nog de volgende methoden in de Breuk klasse:

```
/*
 * Converteert deze Breuk naar string formaat:  "teller/noemer."
 */
public String toString() {
    return ...
}
// http://nl.wikipedia.org/wiki/Algoritme_van_Euclides
// te gebruiken voor het vereenvoudigen van een breuk
static int ggd(int x, int y) {
    ...   // zorg dat x >= 0
    ...   // y > 0 (is noemer)
}
```

- In de getaltheorie, een deelgebied van de wiskunde, is het algoritme van Euclides een efficiënte methode voor het berekenen van de grootste gemene deler (ggd) van twee positieve gehele getallen. De methode ggd() mag recursief zijn en kan alleen voor niet-negatieve getallen gebruikt worden.

- Je kunt de klasse Breuk testen met de volgende TestBreuken klasse. Breid deze klasse uit met code voor het testen van alle methoden uit de Breuk klasse.

```
public class TestBreuken {
    public static void main(String[] args) {
        Breuk a = new Breuk(1, 5);
        Breuk b = new Breuk(9, 12);
        System.out.println("test het paar ");
        System.out.println("a + b      = " + a.telop(b));
        System.out.println("a - b      = " + a.trekaf(b));
        System.out.println("a - b + b = " + a.trekaf(b).telop(b));
    }
}
```

De uitvoer van een TestBreuken klasse zou kunnen zijn:

```
a                  = 1/5
b                  = 3/4
a + b              = 19/20
a - b              =  - 11/20
a - b + b          = 1/5
(a - b) + (b - a)  = 0
a * b              = 3/20
a / b              = 4/15
(a / b) * b        = 1/5
(a / b) * (b / a)  = 1
omgekeerde(a)      = 5
a * omgekeerde(a)  = 1
```

3 Klasse ComplexGetal met a en b doubles

Een complex getal is een uitdrukking van de vorm $a+bi$, waarin a en b beide reële getallen zijn en i een nieuw getal voorstelt, de imaginaire eenheid, met de eigenschap (rekenregel): $i^2 = -1$. Een complex getal $a+bi$ correspondeert met een punt met coördinaten (a,b) in het vlak. a is het reële deel van het complexe getal, en b het imaginaire deel. Er gelden de volgende rekenregels voor complexe getallen:

1. **som**

   $(a+bi)+(c+di) = (a+c)+(b+d)i$

2. **verschil**

   $(a+bi)-(c+di) = (a-c)+(b-d)i$

3. **product**

   $(a+bi)(c+di) = ac+adi+bci+bdi^2 = (ac-bd)+(ad+bc)i$

4. **quotiënt**

   $\frac{a+bi}{c+di} = \frac{(a+bi)(c-di)}{(c+di)(c-di)} = \frac{ac+bd}{c^2+d^2} + \frac{bc-ad}{c^2+d^2}i$

5. **omgekeerde**

$$\frac{1}{a+bi} = \frac{a-bi}{(a+bi)(a-bi)} = \frac{a}{a^2+b^2} + \frac{-b}{a^2+b^2}i$$

Schrijf een klasse ComplexGetal in de file **ComplexGetal.java** met de volgende attributen:

- Twee double instantiatie-variabelen a en b.
- Een constructor met twee parameters.
- In *ComplexGetalInterface.java* staan de methoden die in de klasse ComplexGetal geïmplementeerd moeten worden. Deze file moet je precies zo overtikken.

```
interface ComplexGetalInterface {
    public ComplexGetal telop(ComplexGetal cg);
    public ComplexGetal trekaf(ComplexGetal cg);
    public ComplexGetal vermenigvuldig(ComplexGetal cg);
    public ComplexGetal deel(ComplexGetal cg);
    public ComplexGetal omgekeerde();
}
```

- De header van de klasse ComplexGetal ziet er als volgt uit:

```
public class ComplexGetal implements ComplexGetalInterface {
```

  Hierdoor word je gedwongen de in de interface gedefinieerde methodenheaders te gebruiken in je implementatie.

- Hierbij een stukje code voor het testen van de klasse ComplexGetal wat je kunt zetten in de klasse TestComplexeGetallen.java. Breid deze zelf uit.

```
ComplexGetal a = new ComplexGetal(5.0, 6.0);
ComplexGetal b = new ComplexGetal(-3.0, 4.0);

System.out.println("a             = " + a);
System.out.println("b             = " + b);
System.out.println("b + a         = " + b.telop(a));
System.out.println("a * b         = " + a.vermenigvuldig(b));
System.out.println("(a / b) * b   = " + a.deel(b).vermenigvuldig(b));
```

En hier wat voorbeelduitvoer van de TestComplexeGetallen.java.

```
a             = 5.0 + 6.0i
b             = -3.0 + 4.0i
b + a         = 2.0 + 10.0i
a - b         = 8.0 + 2.0i
a * b         = -39.0 + 2.0i
b * a         = -39.0 + 2.0i
a / b         = 0.36 - 1.52i
(a / b) * b   = 5.0 + 6.0i
```

4 Klasse ComplexGetal met a en b breuken

Copieer de file *ComplexGetal.java* naar een andere file, zodat je wijzigingen kunt gaan aanbrengen. We willen nu complexe getallen $a+bi$ modelleren met a en b breuken.
Deze complexe getallen zien er als volgt uit: $a+bi = \frac{p}{q} + \frac{r}{s}i$. Het reële deel van zo'n complex getal is de breuk $\frac{p}{q}$, en het imaginaire deel is de breuk $\frac{r}{s}$.
De rekenregels voor deze speciale complexe getallen zijn als volgt:

1. **som**
   $\left(\frac{p_1}{q_1} + \frac{r_1}{s_1}i\right) + \left(\frac{p_2}{q_2} + \frac{r_2}{s_2}i\right) = \left(\frac{p_1}{q_1} + \frac{p_2}{q_2}\right) + \left(\frac{r_1}{s_1} + \frac{r_2}{s_2}\right)i$
   Dat betekent dat de som van twee complexe getallen berekend kan worden door de reële breuken van beide getallen op te tellen met behulp van de methode *telop()* van de *Breuk*-klasse, en door de imaginaire breuken met methode *telop()* op te tellen.

2. **verschil**
   $\left(\frac{p_1}{q_1} + \frac{r_1}{s_1}i\right) - \left(\frac{p_2}{q_2} + \frac{r_2}{s_2}i\right) = \left(\frac{p_1}{q_1} - \frac{p_2}{q_2}\right) + \left(\frac{r_1}{s_1} - \frac{r_2}{s_2}\right)i$
   Hier moet twee keer de methode *trekaf()* van de *Breuk*-klasse gebruikt worden.

3. **product**
   $\left(\frac{p_1}{q_1} + \frac{r_1}{s_1}i\right)\left(\frac{p_2}{q_2} + \frac{r_2}{s_2}i\right) = \left(\frac{p_1p_2}{q_1q_2} - \frac{r_1r_2}{s_1s_2}\right) + \left(\frac{p_1r_2}{q_1s_2} + \frac{r_1p_2}{s_1q_2}\right)i$
   Hier moeten de methoden *telop()*, *trekaf()* en *vermenigvuldig()* van de *Breuk*-klasse gebruikt worden.

4. **omgekeerde**
   $$\frac{1}{\frac{p}{q} + \frac{r}{s}i} = \frac{\frac{p}{q}}{\frac{p^2}{q^2} + \frac{r^2}{s^2}} - \frac{\frac{r}{s}i}{\frac{p^2}{q^2} + \frac{r^2}{s^2}}$$
   Hier moet de methoden *telop()*, *vermenigvuldig()*, *deel()* uit de *Breuk*-klasse gebruikt worden. Verder moet het imaginaire deel vermenigvuldigd worden met -1.

5. **quotiënt**
   Het quotiënt kun je berekenen door te vermenigvuldigen met het omgekeerde.

Verander de klasse ComplexGetal in de file **ComplexGetal.java** als volgt:

```
public class ComplexGetal implements ComplexGetalInterface {
    Breuk re;   // reele deel
    Breuk im;   // imaginaire deel

    public ComplexGetal(int t1, int n1, int t2, int n2) {
        re = new Breuk(t1, n1);
        im = new Breuk(t2, n2);
    }

    public ComplexGetal(Breuk a, Breuk b) {
        ...
    }
}
```

Een voorbeeld voor de implementatie van de methode *telop* is:

```
public ComplexGetal telop(ComplexGetal cg) {
    Breuk reeel     = re.telop(cg.re);
    Breuk imaginair = im.telop(cg.im);
    return new ComplexGetal(reeel, imaginair);
}
```

De som van twee complexe getallen krijg je door de reële breuken op te tellen en de imaginaire breuken.
Pas de file *TestComplexeGetallen.java* aan zodat de nieuwe ComplexGetal klasse getest kan worden.

```
public class TestComplexeGetallen {
    public static void main(String[] args) {
        ComplexGetal a = new ComplexGetal(5, 3, 1, 2);
        ComplexGetal b = new ComplexGetal(1, 6, -1, 3);

        System.out.println("a                 = " + a);
        System.out.println("b                 = " + b);
        System.out.println("a + b             = " + a.telop(b));
        System.out.println("a - b             = " + a.trekaf(b));
        System.out.println("a - b + b         = " + a.trekaf(b).telop(b))
    }
}
```

En hier weer voorbeelduitvoer:

```
a                 = 5/3 + 1/2i
b                 = 1/6  - 1/3i
a + b             = 11/6 + 1/6i
a - b + b         = 5/3 + 1/2i
(a - b) + (b - a) = 0 + 0i
a * b             = 4/9  - 17/36i
a / b             = 4/5 + 23/5i
(a / b) * b       = 5/3 + 1/2i
(a / b) * (b / a) = 1 + 0i
omgekeerde(a)     = 60/109  - 18/109i
a * omgekeerde(a) = 1 + 0i
```

5 Inleveren

Lever de files **Breuk.java**, **TestBreuken.java**, **BreukInterface.java**, **ComplexGetal.java**, **TestComplexeGetallen.java** en **ComplexGetalInterface.java** in.
Deze opgave moet voor het inleveren worden ingepakt met het inpakscript dat voor deze opgave op Blackboard staat. Het door het inpakscript gegenereerde bestand moet op Blackboard worden ingeleverd. Andere bestanden worden niet nagekeken!