Laboration 2 – Objektorienterad Programmering

Labbgrupp: 0
Medlemmar:
Johan Brook (900720-0216)
Robin Andersson (900122-0574)

Datum: 2011-09-13

Del 1

Relevanta filer

- `RatNum.java`
- `indata.txt`

Beskrivning

`RatNum.java` är en enkel klass som beskriver ett rationellt tal. Vanliga aritmetiska operationer såsom addition, subtraktion, division, etc. finns som metoder och kan anropas. Vid initialisering av ett nytt rationellt tal förkortas täljarna och nämnarna till deras största gemensamma delare.

För mer information om enskilda metoder, se kommentarer i källkoden i `RatNum.java`.

Vad programmet RatNumTest3.java gör

Programmet låter användaren mata in två bråktal separerat med matematiska eller logiska operatorer (såsom +, -, /, *, =, <) och visar resultatet av operationen, eller ett felmeddelande om något gick fel. Fel kan vara saker såsom en nämnare som är lika med 0 (ett `NumberFormatException` kastas ut), ett felformaterat uttryck (såsom 1/2 + 1/2 + 1/2), eller en okänd operator (såsom 1/2 & 1/2).

Högt upp i main-metoden anropas även en testmetod i filen `RatNumTest2.java` (som är hela det egentliga programmet `RatNumTest2`) och sedan en testmetod i det nuvarande programmet, där RatNums metoder `equals()` och `toDouble()` testas.

I main-metoden läses uttryck in som strängar och separeras i fält, och eventuella felaktigheter hanteras. Första delen av uttrycket (ett bråktal) används i `RatNum`s metod `parse()`, där ett objekt returneras. Konstruktorn i `RatNum` som tar en sträng matas med den andra delen av uttrycket. På så sätt har både metoden `parse()` och parse-konstruktorn testats. Programmet har nu två `RatNum`-objekt som används i den specificerade matematiska operationen (operatorn i uttrycket som användaren matade in) genom att anropa respektive metoder på objekten. Om exempelvis uttrycket ”1/3 + 1/4” matas in skapas två `RatNum`-objekt av ”1/3” och ”1/4” och operatorn ”+” extraheras, och metoden

`RatNum.add()` som utför addition anropas. Resultatet sparas i en deklarerad variabel. Om inget fel inträffade skrivs variabeln ut, och på så sätt anropas den av oss överskuggade metoden `toString()` indirekt.

Resultat

```
$ javac RatNum.java
$ javac RatNumTest3.java
$ java RatNumTest3 < indata.txt
```

Nedan visas output efter körning av testprogrammet `RatNumTest3.java` med data från filen `indata.txt`.

```
> 1/3 + 1/4     --> 7/12
> 2/9 * -4/5    --> -8/45
> 2/6 - 7/9     --> -4/9
> 7/-2 / -2/5   --> 8 3/4
> -5/10 + -3/4  --> -1 1/4
> -5/3 * 4      --> -6 2/3
> 7/9 * 2       --> 1 5/9
> -5 * 1/3      --> -1 2/3
> 2 / -5        --> -2/5
> 2/5 = 40/100  --> true
> 6/18 = -1/3   --> false
> 2/9 < 1/5     --> false
> -5/9 < 1/2    --> true
> 1/2 +1/3      --> Felaktigt uttryck!
> 1/5           --> Felaktigt uttryck!
> /4 + 1/3      --> NumberFormatException: For input string: ""
> 5/ + 1/3      --> NumberFormatException: For input string: ""
> 1//4 + 1/4    --> NumberFormatException: For input string: "/4"
> 1/ - 2 + 1/3  --> Felaktigt uttryck!
> 1/3 a + 1/3   --> Felaktigt uttryck!
> -/3 + 1/3     --> NumberFormatException: For input string: "-"
> 1/3 + 1/3 + 1/3   --> Felaktigt uttryck!
> 1/3 & 1/3     --> Felaktig operator!
> 1/0 + 1/3     --> NumberFormatException: Denominator = 0
> 1 / 0         --> NumberFormatException: Denominator = 0
```

Övriga noteringar

För metoden `RatNum.div()` (se rad nr. 204 i `RatNum.java`) hade vi först en alternativ lösning på division av rationella tal, som ger samma resultat. Se nedan.

```
public RatNum div(RatNum r){
     int numerator = this.numerator * r.getDenominator();
     int denominator = this.denominator * r.getNumerator();

     return new RatNum(numerator, denominator);
}
```

I den nuvarande lösningen återanvänder vi metoden `RatNum.mul()` med inverterad täljare och nämnare – som man gör när man utför division av bråk för hand på papper. Koden blir kort och koncis – endast en rad – och vi slipper deklarera temporära variabler, som i lösningen ovan (där temporära variabler nästan är ett krav för bra läsbarhet).

Vi lade även till ytterligare kod i metoden `RatNum.toString()` (rad nr. 228 i `RatNum.java`) för hantering av tom heltalsdel resp. bråkdel i ett rationellt tal. Vi upptäckte att en operation som ger ett heltal och ingen rest visade resultatet på blandad form, vilket inte är optimalt i det fallet. T.ex. så gav operationen ”1/2 + 1/2” talet ”1 0/4” istället för bara ”1”, vilket är snyggare i våra ögon, och är hur man skriver i verkligheten. Det blev några fler rader kod i metoden, men vi anser att resultatet rättfärdigar det.

Del 2

Relevanta filer

- `RseTestmatriser.java`

(vår egen kod är högst upp i filen).

I del 2 görs test på raders summor i matriser för att se om de är lika mycket. Vi skapade en separat metod `rowSum()` för beräkning av summan av elementen i en enskild rad (ett heltalsfält). Ett par tester utförs för att söka efter null-värden eller tomma fält, och slutligen returneras `true` eller `false`, beroende på om radernas summor överensstämde eller ej. Vår kod består av **25 rader** (utan whitespace och kommentarer).

För närmare detaljer, se källkoden i filen `RseTestMatriser.java`.