Inlämningsuppgift 2, Rationella tal

Avsikten med uppgiften är att du ska bli säker på grundläggande objektorienterade begrepp såsom klasser, objekt, instansmetoder, instansvariabler och konstruktorer. Uppgiften är att konstruera en klass `RatNum` som beskriver rationella tal. Ett rationellt tal uttrycks som bekant som kvoten mellan två heltal. Vi skriver rationella tal på formen `1/3`, `-1/5`. För att förenkla arbetet har uppgiften delats in i tre steg. I varje steg finns det ett färdigt huvudprogram som du ska använda för att testa det du gjort. OBS! Ändra ingenting i något av de färdiga huvudprogrammen!

Steg 1

När man beskriver rationella tal är det bäst att förkorta bort alla gemensamma faktorer i täljaren och nämnaren. För att kunna göra denna förkortning behövs en klassmetod som beräknar den största gemensamma divisorn till två heltal `m` och `n`. Skriv en sådan metod! Kalla metoden `sgd` och placera den i en klass med namnet `RatNum`. Metoden ska acceptera både positiva och negativa argument som inte är lika med noll, men den ska alltid ge ett resultat som är större än noll. Utforma din metod så att den genererar en exception av typen `IllegalArgumentException` om någon av parametrarna är lika med noll. Detta kan göras med satsen

```
throw new IllegalArgumentException ();
```

Tips. *Euklides algoritm* för att beräkna den största gemensamma divisorn till två positiva heltal *m* och *n* kan beskrivas på följande sätt:

- Dividera *m* med *n* och beteckna resten vid divisionen med *r*.
- Om *r=0* så är beräkningen klar och resultatet finns i *n*.
- Sätt annars *m* till *n* och *n* till *r* och gå tillbaka till steg 1.

Denna algoritm fungerar om talen *m* och *n* båda är ≥ 1. Se därför i metoden `sgd` till att parametrarna alltid görs positiva innan algoritmen tillämpas.

Test.

Testa metoden `sgd` genom att kopiera det färdiga huvudprogrammet `RatNumTest1` till samma bibliotek som klassen `RatNum`. Kompilera och kör programmet!

Steg 2

Det är nu dags att börja fylla i klassen `RatNum` så att den kan användas för att beskriva rationella tal. För varje objekt av klassen `RatNum` ska täljaren och nämnaren lagras som två heltal. Dessa ska alltid sakna gemensamma faktorer, dvs. ett rationellt tal ska alltid vara avkortat så långt det går. Nämnaren måste alltid vara större än noll. Negativa rationella tal representeras med en negativ täljare.

I detta steg ska du förse klassen `RatNum` med följande konstruktorer och metoder:

- En konstruktor utan parametrar som initierar det aktuella talet till `0/1`. (Täljaren blir alltså 0 och nämnaren 1.)

- En konstruktor med en parameter `a` (ett heltal), som betecknar täljaren. Det aktuella talet initieras till `a/1`.

- En konstruktor med två parametrar, `a` och `b` (två heltal), vilka betecknar täljaren och nämnaren. Om `a` och `b` saknar gemensamma faktorer initieras det aktuella talet till `a/b`, annars divideras först `a` och `b` med sin största gemensamma divisor. Om `b` är lika med noll ska en exception av typen `NumberFormatException` genereras. Detta kan göras med satsen
  ```
  throw new NumberFormatException("Denominator = 0");
  ```

- En konstruktor (en s.k. kopieringskonstruktor) som har ett annat rationellt tal `r` som parameter. Det aktuella talet initieras så att det innehåller samma täljare och nämnare som `r`.

- En metod `getNumerator` som avläser och returnerar det aktuella talets täljare.

- En metod `getDenominator` som avläser och returnerar det aktuella talets nämnare.

Tips.

I den tredje konstruktorn måste du först kontrollera att nämnaren inte är lika med 0. Om så inte är fallet så ska du dividera de båda parametrarna `a` och `b` med den största gemensamma divisorn. Du finner lätt den största gemensamma divisorn genom att använda klassmetoden `sgd` från steg 1. (Men tänk på att denna metod kräver att parametrarna är större än 0, så du måste använda dig av absolutvärdena `a` och `b`.) En liten komplikation är att någon eller båda av parametrarna `a` och `b` kan vara mindre än noll. Om båda är negativa betecknar `a/b` ett positivt tal och man kan därför ersätta `a` och `b` med sina motsvarande positiva värden. Om den ena parametern är positiv och den andra negativ betecknar ett `a/b` negativt tal. Man ska då låta täljaren vara negativ och nämnaren positiv.

Test.

Testa klassen `RatNum` genom att kopiera det färdiga huvudprogrammet `RatNumTest2` till samma bibliotek som klassen `RatNum`. Kompilera och kör programmet! Läs sedan programkoden för huvudprogrammet `RatNumTest2` och förvissa dig om att du förstår vad detta program gör. Om programmet upptäcker något fel skriver det ut felets nummer. Du får då titta i koden för huvudprogrammet för att se vilken test det var som misslyckades.

Steg 3

Klassen ska nu utökas med några metoder, vilka förenklar inläsning och utskrift av rationella tal och gör det möjligt att utföra matematiska operationer på rationella tal. Följande metoder ska läggas till:

- En metod `toString` som returnerar det aktuella rationella talet som en text. Om talets täljare är `a` och dess nämnare är `b` och det gäller att `a < b` ska texten ha formen `"a/b"`,

annars ska texten ha formen `"h d/n"`, där `h` är heltalsdelen av divisionen `a/b` och `d` resten. Om det är fråga om ett negativt rationellt tal ska texten ha formen `"-h d/n"`. Om `d = 0` ska bara heltalsdelen vara med i texten. (Detta gäller för både positiva och negativa rationella tal.)

- En metod `toDouble` som returnerar ett närmevärde för det aktuella talet. Returtypen ska vara **`double`**.

- En *klassmetod* `parse` som har en parameter `s` av typen `String`. Parametern ska undersökas och om den innehåller en text som på ett korrekt sätt beskriver ett rationellt tal ska metoden `parse` skapa ett nytt rationellt tal med det angivna värdet och returnera en referens till det nya talet. (Jämför med metoden `parseInt` i standardklassen `Integer`.) Texten i parametern `s` får ha någon av fyra följande former, där `a` och `b` betecknar heltalskonstanter: `"a/b"`, `"-a/b"`, `"a/-b"` eller `"a"`. Det får inte finnas några blanka tecken. Den sista formen visar att det är tillåtet att bara ange täljaren. I så fall ska nämnaren antas vara lika med 1. Om texten i `s` ser ut på ett otillåtet sätt ska `parse` generera en exception av typen `NumberFormatException`.

- En konstruktor med en parameter av typen `String`. Om parametern innehåller en text som på ett korrekt sätt beskriver ett rationellt tal ska det nya talet initieras, annars ska en exception av typen `NumberFormatException` genereras.
  Tips: Använd dig av metoden `parse` samt den kopieringskonstruktor du skrev tidigare.

- En parameterlös metod `clone` som skapar en kopia av det aktuella talet. Som returvärde ger metoden en referens till kopian. Obs. För att denna metod ska följa mönstret i Java är returtypen inte `RatNum`, utan `Object` (som är en superklass till alla klasser).

- En metod `equals` som har ett annat rationellt tal `r` som parameter vilket jämförs med det aktuella talet. Om de två talen är lika returneras **`true`** annars **`false`**. (Om `r =` **`null`** returneras värdet **`false`**.)

- En metod `lessThan` som har ett annat rationellt tal `r` som parameter vilket jämförs med det aktuella talet. Om det aktuella talet är mindre än `r` returneras **`true`** annars **`false`**
- En metod `add` som har ett annat rationellt tal `r` som parameter. Som resultat ges ett nytt rationellt tal som innehåller summan av det aktuella talet och `r`.

- En metod `sub` som har ett annat rationellt tal `r` som parameter. Som resultat ges ett nytt rationellt tal som innehåller skillnaden mellan det aktuella talet och `r`.

- En metod `mul` som har ett annat rationellt tal `r` som parameter. Som resultat ges ett nytt rationellt tal som innehåller produkten av det aktuella talet och `r`.

- En metod `div` som har ett annat rationellt tal `r` som parameter. Som resultat ges ett nytt rationellt tal som innehåller kvoten mellan det aktuella talet och `r`.

Tips.

I metoderna `add`, `sub`, `mul` och `div` använder du förstås dina matematiska kunskaper. Uttrycket `a/b+c/d` kan t.ex. skrivas om som `(ad+bc)/bd`. Använd en lämplig konstruktor när du skapar det tal som ska innehålla resultatet av den matematiska operationen. Då kommer automatiskt alla gemensamma faktorer att divideras bort.

I metoden `parse` har du stor nytta av några av metoderna `indexOf` och `substring` i klassen `String`. Ett litet tips: Om parametern `s` till `parse` inte innehåller någon nämnare kan du själv lägga till texten `"/1"` sist i `s`. På det sättet behöver du inte konstruera någon speciell kod för detta fall. Använd metoden `Integer.parseInt` för att avkoda täljaren respektive nämnaren. Denna genererar som du vet själv en exception av typen `NumberFormatException` om texten inte innehåller ett heltal.

Test.

När klassen `RatNum` är klar ska den testas tillsammans med det färdiga huvudprogrammet `RatNumTest3`. Studera detta program i detalj och sätt dig in i hur det fungerar! När man kör det färdiga huvudprogrammet tillsammans med din klass `RatNum` kan det se ut på följande sätt:

```
Skriv uttryck med formen a/b ? c/d, där ? är något av tecknen + - * / = <
> 1/3 + 1/5
1/3 + 1/5       --> 8/15
> 2/3 * 2/5
2/3 * 2/5       --> 4/15
> 1/3 - 2/5
1/3 - 2/5       --> -1/15
> 2/3 / 2/5
2/3 / 2/5       --> 1 2/3
> -2/3 - 2/5
-2/3 - 2/5      --> -1 1/15
> 2/11 < 1/5
2/11 < 1/5      --> true
> 3/15 = 1/5
3/15 = 1/5      --> true
> 5 / 2/3
5 / 2/3 --> 7 1/2
> 5/9 * 2
5/9 * 2 --> 1 1/9
>
```

För att underlätta testningen av klassen `RatNum` finns det en färdig fil `indata.txt` som du kan använda om du vill. Du kör den genom att i ett textfönster ge kommandot

```
java RatNumTest3 < indata.txt
```

Utskriften som programmet då ger kan jämföras med "facit" som finns i filen `utdata.txt`.