# Programtext

**import** java.lang.Math;

**public** **class** RatNum **implements** Cloneable{

//STEG 1

**public** **static** **int** sgd(**int** m, **int** n){

//ger felmeddelande om m eller n = 0

**if**(m==0||n==0){

**throw** **new** IllegalArgumentException();

}

//Euklides algoritm

**while**(**true**){

**int** r=m%n;

**if**(r==0){

**return** Math.*abs*(n);

}

**else**{

m=n;

n=r;

}

}

}

//STEG 2

**int** taljare;

**int** namnare;

**public** RatNum(){

taljare = 0;

namnare = 1;

}

**public** RatNum(**int** a){

taljare = a;

namnare = 1;

}

**public** RatNum(**int** a, **int** b){

**if**(b==0){

**throw** **new** NumberFormatException("Denominator=0");

}

//ser till att nämnaren är positiv och byter tecken på a om tecken på b byts

**if**(b<0){

a=-a;

b=-b;

}

//initierar täljaren till 0 och nämnaren till b om a=0 och b!=0

**if**(a==0){

taljare=a;

namnare=b;

}

//initierar täljaren till a och nämnaren till b om sgd(a,b) == 0

**else** **if**(*sgd*(a,b)==0){

taljare=a;

namnare=b;

}

//initierar täljaren till a delat med den största gemensamma faktorn och nämnaren till b delat med den största gemensamma faktorn

**else**{

taljare = a/*sgd*(a,b);

namnare = b/*sgd*(a,b);

}

}

//returnar täljaren

**public** **int** getNumerator(){

**return** taljare;

}

//returnar nämnaren

**public** **int** getDenominator(){

**return** namnare;

}

//En metod som skriver ut täljaren och nämnaren för tester

**public** **void** printit(){

System.***out***.println(taljare + ", " + namnare);

}

//"Kopieringskonstruktor"

**public** RatNum(RatNum r){

taljare=r.getNumerator();

namnare=r.getDenominator();

}

//STEG 3

//Returnerar det aktuella talet som en text med formen "a/b".

**public** String toString(){

**if**(taljare==0){

**return** "0";

}

**if**(Math.*abs*(taljare)>=namnare){

**if**(namnare==1){

**return** Integer.*toString*(taljare);

}

**if**(taljare<0){

**int** m=-taljare;

**int** f=m%namnare;

**int** h=(m-f)/namnare;

**int** k=m-namnare\*h;

**return** "-" + h + " " + k + "/" + namnare;

}

**else**{

**int** f = taljare%namnare;

**int** h=(taljare-f)/namnare;

**int** k=taljare-namnare\*h;

**return** h + " " + k + "/" + namnare;

}

}

**else**{

**return** taljare+"/" + namnare;

}

}

//Returnerar ett närmevärde för det aktuella talet

**public** **double** toDouble(){

**return** (**double**)taljare/namnare;

}

//Kontrollerar om det är skrivet på korrekt sätt och i så fall skapar ett rationellt tal

**public** **static** RatNum parse(String s){

//Kontrollerar om parametern är på formen "a"

**if**(s.matches("[0-9]+")){

**int** d = Integer.*parseInt*(s);

**return** **new** RatNum(d);

}

//Kontrollerar om parametern är på formen "-a"

**else** **if**(s.substring(0,1).equals("-") && s.substring(1,s.length()).matches("[0-9]+")){

**int** d = Integer.*parseInt*(s);

**return** **new** RatNum(d);

}

//Kontrollerar om / existerar, delar i så fall upp parametern i två delar: det som kommer före / och det som kommer efter. Om parametern är på formen "-a/-b" ges en NumberFormatException, annars görs det som kommer före / och det som kommer efter / om till två integers. Om det inte är rätt form så kommer en NumberFormatException ges.

**else** **if** (s.indexOf('/')!=-1){

**int** m=s.indexOf('/');

**if**(s.substring(0,1).equals("-") && s.substring(m+1,m+2).equals("-")){

**throw** **new** NumberFormatException();

}

String x = s.substring(0,m);

String y = s.substring(m+1,s.length());

**int** x1=Integer.*parseInt*(x);

**int** y1=Integer.*parseInt*(y);

**return** **new** RatNum(x1,y1);

}

**else**{

**throw** **new** NumberFormatException();

}

}

//Initierar det nya talet om det är skrivet på korrekt sätt, annars skickar felmeddelande

**public** RatNum(String s){

taljare=*parse*(s).getNumerator();

namnare=*parse*(s).getDenominator();

}

//Försöker använda metoden clone, annars visar ett felmeddelande

**public** Object clone(){

**try** {

**return** **super**.clone();

}

**catch** (CloneNotSupportedException e) {

**throw** **new** InternalError();

}

}

//Jämför det aktuella talet med ett annat rationellt tal

**public** **boolean** equals(RatNum r){

**return**(taljare==r.getNumerator() && namnare==r.getDenominator());

}

//Jämför det aktuella talet med ett annat tal. Om det aktuella talet är mindre än r returneras true annars false

**public** **boolean** lessThan(RatNum r){

**return**(taljare\*r.getDenominator()<r.getNumerator()\*namnare);

}

//returnerar ett nytt rationellt tal som är summan av det aktuella talet och ett annat tal

**public** RatNum add(RatNum r){

**int** m=r.getNumerator();

**int** k=r.getDenominator();

**return** **new** RatNum((taljare\*k + m\*namnare),(namnare\*k));

}

//returnerar ett nytt rationellt tal som är differensen av det aktuella talet och ett annat tal

**public** RatNum sub(RatNum r){

**int** m=r.getNumerator();

**int** k=r.getDenominator();

**return** **new** RatNum((taljare\*k - m\*namnare),(namnare\*k));

}

//returnerar ett nytt rationellt tal som är produkten av det aktuella talet och ett annat tal

**public** RatNum mul(RatNum r){

**int** m=r.getNumerator();

**int** k=r.getDenominator();

**return** **new** RatNum((taljare\*m),(k\*namnare));

}

//returnerar ett nytt rationellt tal som är kvoten av det aktuella talet och ett annat tal

**public** RatNum div(RatNum r){

**int** m=r.getNumerator();

**int** k=r.getDenominator();

**return** **new** RatNum((taljare\*k),(namnare\*m));

}

**public** **static** **void** main(String[] args){

String x = "a";

System.***out***.println(x.indexOf("abcd"));

}

}

# Utskrift

> 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

# Kommentar

|  |  |
| --- | --- |
| Rad: | Funktion: |
| 8-13 | Testar equals |
|  | Testar clone |
| 21-25 | Testar toDouble genom att skapa ett antal nya RatNums på formen (j/i), subtraktera (double)(j/i) från dessa och kolla om absolutbeloppet av det erhållna värdet är ungefär lika med noll (felmarginal på 1.0e-13) |
| 28-29 | Kör testet RatNumTest2 |
| 30 | Skapar en ny BufferedReader, *in* refererar till denna |
| 31 | Skriver ut text |
| 32 | Påbörjar while-loopen |
| 33 | Skriver ut ”> ” i början av varje rad och skriver ut allt som är skrivet i System.out (tömmer bufferten genom *flush*) |
| 34 | Skapar en string *s*, som består av det som användaren har skrivit |
| 35-36 | Om användaren inte skrev något så försöker programmet igen. |
| 37 | Skapar en ny scanner sc med stringen *s* som parameter. Denna scanner är indelad i tokens, som är avdelade med mellanslag i stringen *s*. |
| 38 | Deklarerar en array *a* som består av strings och har 3 element |
| 39 | Deklarerar en int *i* |
| 40-41 | Lägger till tokens till *a* ifall de existerar. För varje element som läggs till blir i=i+1, där ’startvärdet’ för i är 0 och maxvärdet är 3. |
| 42 | Startar en if-loop om i>0 (alltså om åtminstone ett element har lagts till till arrayen *a.* |
| 43 | Skriver ut stringen *s*, en tabulator och en pil |
| 44-45 | Kollar att i=3 (alltså 3 tokens har lagts till till arrayen *a*) och att *sc* inte har en till token (alltså stringen *s* inte består av mer än 3 element). Om något av de två vilkoren inte är sanna skrivs ”Felaktigt uttryck!” ut. |
| 46-47 | Annars försök att: |
| 48 | Gör en ny RatNum *r1* av den första tokenen i arrayen *a*. |
| 49 | Gör en ny String *op* den andra tokenen i *a*. |
| 50 | Gör en ny char av det första tecknet i *op*. |
| 51 | Gör en ny RatNum *r2* av den tredje tokenen i arrayen *a*. |
| 52-53 | Om *op* inte har längden 1 (alltså inte består av bara ett tecken) eller *c* inte tillhör ”+-\*/=<” så skriver programmet ut ”Felaktig operator!”. |
| 54 | Annars: |
| 55 | Skapar en ny RatNum *res* = null. |
| 56-63 | Använder metoden add, sub, mul eller div på *r1* och *r2* och sparar resultatet i *res* (programmet använder den metoden som behövs beroende på *c*)*.* |
| 64-67 | Använder metoden equals eller lessthan och skriver ut resultatet (även här använder programmet den metoden som behövs beronde på *c*). |
| 68-72 | Om add sub mul eller div användes (alltså *c* är +, -, \* eller /) så skrivs ”Fel i add, sub, mul eller div” ut om *res*==null. Annars skrivs *res* ut. |
| 73-74 | Avslutar else från rad 54 och try från rad 46-47. |
| 75-77 | Om en NumberFormatException *e* gavs i try så skriver programmet ut ”NumberFormatException: ” + e.getMessage(). |
| 78 | Avslutar if loopen från rad 44. |
| 79-81 | Avslutar programmet. |