Laboration nr 2

Syfte

Att få förståelse för de grundläggande objektorienterade begreppen.

Redovisning

Källkoden för uppgifterna skall skickas in via Fire.

Uppgift 1

Skriv ett klass Rational för att hantera *rationella tal.* Ett rationellt tal uttrycks som bekant som kvoten mellan två heltal. Vi skriver rationella tal på formen 1/3, -1/5.

Steg1:

När man beskriver rationella tal är det bäst att förkorta bort alla gemensamma faktorer i täljaren och nämnare. För att kunna göra denna förkortning behöver vi en metod som beräknar den största gemensamma divisorn (*eng.* greatest common divisor) till två positiva heltal m och n. Börja med att skriva en sådan metod. Kalla metoden gcd, gör den till en *klassmetod* och placera den i en klass med namnet Rational.

För att beräkna största gemensamma divisor är det lämpligt att använda *Euclides algoritm* som har följande utseende:

1. Sätt m till det största av de två talen och n till det minsta av talen.
2. Dividera m med n och beteckna resten vid divisionen med r.
3. Om r = 0 så är beräkningen klar och resultatet finns i n.
4. Sätt annars m till n och n till r. Upprepa från punkt 2.

Test:

Testa metoden gcd genom att kopiera det färdiga huvudprogrammet RationalTest1 som finns på kursens hemsida till samma bibliotek som klassen Rational.

Steg 2:

För varje objekt av klassen Rational skall täljaren och nämnaren lagras som två heltal (dessa är således instansvariabler i klassen). Täljaren och nämnaren skall alltid sakna gemensamma faktorer, dvs. ett rationellt tal skall alltid vara avkortat (normaliserat) så långt det går. Nämnaren måste alltid vara större än noll. Negativa rationella tal representeras med en negativ täljare.

I detta steg skall du förse klassen Rational 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, vilken betecknar täljaren. Det aktuella talet initieras till a/1.
- En *konstruktor* med två parametrar, a och b 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 ges en felutskrift och programmet avslutas*.
- 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.
Anm. I Rational(Rational r) { … } är instansvariablerna i r synliga i kroppen { … } även om de

deklarerats som privata. En klass har access till instansvariablerna i *alla* objekt av klassen, inte bara i det aktuella objektet.

- En metod set som har två parametrar, a och b vilka betecknar täljaren och nämnaren. Om a och b saknar gemensamma faktorer sätts det aktuella talet till a/b, annars divideras först a och b med sin största gemensamma divisor. Om b är lika med noll ges en felutskrift och programmet avslutas.

- En andra version av metoden set. Denna version har bara en parameter a vilken betecknar täljaren. Det aktuella talet sätts till a/1.

- En tredje version av metoden set. Denna version har ett annat rationellt tal r som parameter. Det aktuella talet sätts 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:

Börja med att skriva den första versionen av metoden set. I denna måste du kontrollera att nämnaren inte är lika med 0. I så fall får du ge en felutskrift och avsluta programmet (System.exit(0);). Börja med att dividera de båda parametrarna a och b med den största gemensamma divisorn. Du finner lätt den största gemensamma divisorn genom att använda klassmetoden gcd från steg 1. (Tänk dock 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 skall då låta täljaren vara negativ och nämnaren positiv.

Skriv sedan de två andra versionerna av set, vilka blir mycket enkla.

Skriv nu de fyra konstruktorerna. Eftersom de kan anropa de olika versionerna av metoden set blir de alla triviala.

Metoderna getNumerator och getDenominator blir också mycket enkla och bör inte vålla dig några problem.

Test:

Testa klassen Rational genom att kopiera det färdiga huvudprogrammet RationalTest2, som finns på kursens hemsida. Kompilera och kör programmet! Läs sedan programkoden för huvudprogrammet RationalTest2 och förvissa dig om att du förstår vad detta program gör.

Steg 3:

Klassen skall 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 skall läggas till:

- En metod toString som returnerar det aktuella talet som en text med formen "a/b".
(Som extrauppgift kan du senare, när resten av klassen är klar, utöka metoden toString så att den presenterar resultatet med formen k r/b, där k är kvoten i heltalsdivisionen a/b och r resten. Om det är fråga om ett negativt tal skall man i detta fall presentera resultatet med formen –k r/b.) Ex. 11/4 skall presenteras som 2 3/4.

- En *klassmetod* parse som har en parameter s av typen String. Parametern skall undersökas och om den innehåller en text som på ett korrekt sätt beskriver ett rationellt tal skall 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.) Om texten i s ser ut på ett otillåtet sätt skall parse inte skapa något nytt rationellt tal utan avbryta programmet (System.exit(0);). 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. Här gäller att *b inte får vara lika med 0*. Den sista formen visar att det är tillåtet att bara ange täljaren. I så fall skall nämnaren antas vara lika med 1.

- 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 skall följa mönstret i Java är returtypen inte Rational, utan Object (förklaringen till detta får ni senare i kursen).

- 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**.

- 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 rationell 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 rationell 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 rationell 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 rationell 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 skall innehålla resultatet av den matematiska operationen. Då kommer automatiskt alla gemensamma faktorer att divideras bort. (Det är förstås inte tillåtet att använda flyttal vid uträkningarna!)

I metoden parse har du stor nytta av några av de metoder som finns i klassen String. Ett litet knep: 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. När Integer.parseInt får en parameter som inte kan översättas till ett heltal kastar den en *exception* av typen NumberFormatException. För att hantera eventuella exceptions skall ni använda en **try-catch**-sats, inom vilken anropen av Integer.parseInt läggs. Ex.

```
int x;
try { x = Integer.parseInt(s); }
catch (NumberFormatException e) { return null; }
```

Exceptions kommer att behandlas utförligt senare i kursen.

Test:

När klassen Rational är klar skall den testas tillsammans med det färdiga huvudprogrammet RationalTest3 som kan lagddas ner från kursens hemsida. *Studera detta program i detalj och sätt dig in i hur det fungerar!* När man kör det färdiga huvudprogrammet tillsammans med din klass Rational kan det se ut på följande sätt:

Skriv uttryck på 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 Rational 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.

Uppgift 2

En fotbollstabell har följande utseende:

Kalmar FF	16	11	2	3	37-16	35
Elfsborg	16	10	4	2	22-5	24
Helsingborg	16	9	4	3	29-18	31
IFK Göteborg	16	8	5	3	24-14	29

Kolumnerna i tabeller innehåller i tur och ordning följande information: lagets namn, antal spelade matcher, antal vunna matcher, antal oavgjorda matcher, antal förlorade matcher, gjorda respektive insläppta mål samt antal poäng.

Några engelska fotbollstermer

Fotbollslag	football team
Fotbollsserie	football league
Vinst	win
Oavgjort	draw
Förlust	defeat
Göra mål	score a goal
Släppa in mål	concede a goal
Målskillnad	goal difference

I denna uppgift skall du skapa klasser för att handha en fotbollsserie. Använd konsekvent *engelska* vid namngivning av klasser, variabler och metoder!

Steg 1:

Börja med att skriva en klass FootballTeam som beskriver ett fotbollslag med den information som är nödvändig för att skapa en tabell enligt ovan.

- Definiera nödvändiga instansvariabler i klassen.
- Skriv en konstruktor för att skapa ett fotbollslag. Inga matcher är spelade när laget skapas.
- Skriv accessmetoder (*int getSomething()* ...) för att avläsa aktuell information om laget, dvs lagets namn, antal spelade matcher, antal vunna matcher, antal förlorade matcher, antal oavgjorda matcher, antalet gjorda mål, antalet insläppta mål och antalet poäng.
- Skriv en metod

  public void registerMatch(**int** scoredGoals, **int** concededGoals)

  som uppdaterar berörda instansvariabler. Vid vinst erhålls 3 poäng, vid oavgjort 1 poäng och vid förlust 0 poäng.
- Skriv en metod toString för att kunna få en utskrift på formen:

  Kalmar FF 16 11 2 3 37-16 35
- Skriv en metod

  public int goalDifference()

  som returnerar differansen mellan gjorda och insläppta mål.
- Skriv en en metod

  public int compareTo(FootballTeam otherTeam)

  som jämför det aktuella lagets resultat med resultatet för laget otherTeam. Metoden skall returnera värdet 1 om det aktuella laget har bättre resultat än det andra laget, värdet 0 om resultaten är lika och värdet -1 om det andra laget har bättre resultat. Vid jämförelsen är i första hand lagens poäng avgörande, i andra hand lagens målskillnad och i tredje hand flest gjorda mål.
- Skriv ett program för att testa klassen.

Steg 2:

Skriv en klass FootballLeague som representerar en fotbollsserie. För varje fotbollsserie skall följande uppgifter lagras:

- namn (t.ex Allsvenskan, Premier League, Div 4 Västra Götaland)
- en lista av lagen som ingår i serien, dvs en lista med referenser till objekt av klassen FootballTeam

Listan med lagen implementeras lämpligen som ett fält.

Klassen FootballLeague skall innehålla

- en konstruktor med två parametrar; namnet på serien och en lista av namnen på lagen som ingår i serien.
- en metod

  public FootballTeam getTeam(String name)

  som returnerar en referens till det angivna laget om detta lag finns i serien, annars returneras **null**.
- en metod

  public boolean registerMatch(String name1, String name2, **int** scored1, **int** scored2)

  som uppdaterar *serietabellen* med resultatet från en match. Parametrarna name1 och name2 anger vilka lag som spelat matchen och parametrarna scored1 och scored2 anger hur många mål respektive lag gjort. Metoden returnerar **true** om resultatet av matchen kan registreras, dvs. att båda lagen spelar i serien, annars returnerar metoden **false**.
- en metod

  public void inputResult()

  som läser matchresultat från en dialogruta och registrerar dessa i *serietabellen*. Ge matchresultatet t på formen:

  Djurgården;IFK Göteborg;0;5

  Vid avkodningen används lämpligen metoden String.split som returnerar ett fält av strängar.
  Ex: `"a+b + c".split("+")` ger fältet `{"a","b "," c"}`.
- en metod

  public void sortTable()

  som sorterar lagen i serien efter hur de ligger till i serien för tillfället. Utnyttja metoden java.util.Arrays.sort för att utföra sorteringen.
- en metod

  public String toString()

  som returnerar den aktuella (sorterade) serietabellen i textform.
- en metod

  public void printTable()

  som skriver ut den aktuella (sorterade) serietabellen på skärmen.
- en main-metod som testar klassen.

Steg 3: (Frivillig utvidgning. Innehåller bl.a. filhantering som ännu inte tagits upp i kursen)

Utöka klassen med:

- en metod

  public int readResultFile(String fileName)

  som läser en textfil med matchresultat och registrerar dessa i *serietabellen*. Filen innehåller ett okänt antal rader på formen:

  IFK Göteborg;Djurgården;5;0
  AIK;GAIS;0;4
  IFK Norrköping;Helsingborg;1;3

  Filen kan innehålla matchresultat från olika serier. Endast matchresultat som tillhör den aktuella serien, dvs båda lagen spelar i serien, registreras. Metoden returnerar antalet matcher som blivit registrerade.

- Skriv två metoder

  public static boolean saveLeague(FootballLeague league, String fileName)
  public static FootballLeague loadLeague(String fileName)

som lagrar ner den angivna serien till en fil, respektive läser in en serie från en fil. För att kunna göra detta måste klasserna FootballLeague och FootballTeam implementera gränssnittet Serializable. Metoderna skall hantera *exceptions* på ett lämpligt sätt.