Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Erland Holmström Göteborg 8 sep 2013

Laboration 2 del 1

Anvisningar:

Ibland anges inom parentes (tex 50R) hur många rader som ungefär går åt för att skriva programmet. Du kanske behöver lite fler rader men om du har många fler rader (tex dubbelt så många, tomma och kommentarer oräknade) så är det förmodligen något fel på ditt program (men det måste inte vara så; kontakta handledare). Men det är ingen tävling i att skriva det kortaste programmet jag vill bara ge er en chans att reagera på felaktiga konstruktioner i tid. Vanligaste felet med långa program är att man har (för) många specialfall.

Programmet skall utformas enligt de principer som lärts ut i kursen och koden skall vara indenterad mm enligt stilguiden på hemsidan, se länksidan.

För att bli godkänd räcker det alltså inte med att programmet fungerar korrekt, det måste vara välstrukturerat och läsbart också!

Tänk också på att labbarna inte är "*nåt (ointressant) som måste göras*" utan är ett inlärningsmoment. Labbarnas innehåll dyker också alltid upp på tentan.

Förkunskaper:

Den här uppgiften kräver kunskap om metoder, fält, val (if), loopar (for), primitiva typen int, logiska operatorer. *Efter föreläsning 5 har vi täckt det.*

Syfte:

Att öva på sakerna ovan. Att öva på att skriva testprogram för sina program och att komma till insikt att det är ganska svårt att testa ett program för korrekthet. Det är helt enkelt svårt att komma på alla testfall.

Känt citat: Testning kan hitta fel men inte garantera att det inte finns några.

Inlämning:

Klassen Rse, din testklass TestRse och i README filen utdata från det givna testprogrammet TestRseEH (klipp och klistra resultatet från terminalfönstret men se till att det blir läsligt).

Testprogrammen jag ibland ger er (tex TestRseEH) är egentligen inte skrivna för er. Dom skrevs för att rättarna skulle kontrollera era program. Dom hittar naturligtvis en del fel i era program och istället för att ta in era program och ge retur så ger jag er alltså testprogrammen så slipper vi en returvända. Ni bör titta i testprogrammen för att få en uppfattning om hur man kan skriva dom men det är inte meningen att ni skall förstå allt som står där. Ni bör också skriva era egna testprogram innan ni tittar på mina.

När du lämnar in lab 2 så kan du lägga alla filerna i samma mapp, det är ju inte så många.

Dvs lösningen på den här labben ligger i
.../xx.lab2/Rse.java där xx är ditt labgruppsnummer.

Om du helst vill ha en undermapp för varje labb så måste den heta som klasserna dvs tex Rse. Så lösningen på den här labben ligger då i
.../xx.lab2/Rse/Rse.java

En README fil räcker för bägge uppgifterna.

Är radsummorna lika? (30R)

En matris i Java är en vektor av vektorer. Den deklareras som en vektor men med två index dvs ”`int[][] m`” är en vektor som innehåller vektorer som innehåller heltal.

Skriv en klass, `Rse`, med funktionen

```
public static boolean allRowSumsEqual(int[][] m)
```

som avgör om summan av elementen i respektive rad i en matris, `m`, är lika. Du får bara använda ett par variabler (ej extra fält) i funktionen.

Du skall skriva en hjälpfunktion, `public static int rowSum(int[] v)`, för att summera elementen i en rad. Se till att det fungerar för: tomma matriser (då är radsummorna lika, OBS att det finns två sorters tomma matriser!), matriser med en rad, matris med en rad som är tom osv. (vad innebär det att matrisen är tom? att en rad är tom?), matriser med olika längd på raderna mm.

Exempel: Första matrisens radsummor är lika, den andras är det inte.

```
1 2 1 2 1 2 -- summa = 9      1 2 3 4     -- summa = 10
4 0 0 2 3   -- summa = 9      5 6 7 8     -- summa = 26
1 1 1 1 1 4 -- summa = 9      1 1 1 1     -- summa = 4
```

När din klass är klar så skall du skriva en klass, TestRse, med en main-metod som testar din klass (läggs i separat fil). Försök att skapa relevanta testfall som täcker alla situationer dvs tomma fält, fält med ett element, fält där alla rader är lika, inga rader lika osv. *Detta skall du göra innan du tittar på min testfil.*

Här är två matriser du kan kopiera in till ditt program och testa med. Du behöver alltså inte läsa in matriserna från användaren, det blir för jobbigt att göra det hela tiden, utan ”hårdkoda” dom i programmet.(men du behöver fler testfall än dessa).

```
int [][] a = { {1, 2, 1, 2, 1, 2},
               {4, 0, 0, 2, 2, 1},
               {1, 1, 1, 1, 1, 4} };

int [][] b = { {1, 2, 3, 4},
               {5, 6, 7, 8},
               {1, 1, 1, 1} };
```

När du tycker att du är färdig så finns det en klass med ett testprogram på hemsidan du skall köra, TestRseEH.java. Det består av flera testfall med krångliga exempel på matriser som programmet skall klara (men du skall inte kika här innan du skrivet ditt eget testprogram!). Framförallt testas många specialfall som tomma matriser. Du laddar ner testprogrammet och kompilerar det som om det var din TestRse klass.

I en matris `m` är antalet rader lika med `m.length` och antalet kolumner i första raden är `m[0].length`. För utskrift kan du använda metoden `print` som också finns i testprogrammet: (studera metoden print så du förstår den...)

```
static void print(int[][] m) {
  if ( m==null ) {
    System.out.print("[ null ]");
  } else if ( m.length == 0 ) {
    System.out.print("[ - ]");
  } else {
    System.out.print("[ ");
    for ( int row=0; row<m.length; row++ ) {
      if ( m[row]==null ) {
        System.out.print("null ");
```

```
        } else {
          System.out.print("{ ");
          if ( m[row].length == 0 ) {
            System.out.print("- ");
          } else {
            for ( int col=0; col<m[row].length; col++ ) {
              System.out.print( m[row][col] + " " ) ;
            }
          }
          System.out.print("} ");
        }
      }
      System.out.print("]");
    }
  }
```

Vanliga fel:

- Ett av dom vanligaste felen är att man inte läser instruktionerna ordentligt!
  Att tex beräkna en radsumma flera gånger (se nedan) efter att man läst dessa råd om vanliga fel är ett allvarligt fel!
- Ett vanligt fel här är att man på olika sätt sparar *alla* radsummor. Man skall naturligtvis beräkna och spara den första och sedan jämföra de övriga mot den.
- Ett annat fel är att man beräknar radsummorna flera gånger – max en gång per radsumma såklart.
- ett tredje fel är att man har ett för dåligt testprogram så man tror att Rse fungerar men .... Vad skall tex resultatet vara för `int[][] nn ={{0, 0, 0, 0}, null, {}}`?
  och så (naturligtvis...) samma fel som brukar förekomma i lab 1 :-(
- Indenteringen är felaktig eller saknas.Se "Links".
- Använd max 80-100 tecken på en rad i filerna. Se "Links"
- Namn och grupp skall finnas först i *alla* filer, även källtext.
- Ni har inte gjort någon tar-boll eller gjort en felaktig tar-boll dvs inte lämnat in på korrekt sätt.
- Inga onödiga filer får skickas in tex tilde filer eller .class filer.