Mittuniversitetet
MID SWEDEN UNIVERSITY

dt006g Java I

Laboration: Sortering

Martin Kjellqvist*

1 april 2015

1 Introduktion

Sortering är en väldigt vanlig uppgift för datorprogram. Metoderna för sortering är många och av varierande kvalitet. Ofta introducerar kurser metoder som Bubblesort och Selection sort, de är förvisso enkla att förstå men är i övrigt av usel kvalitet.

En metod som däremot är väldigt användbar är merge-sort. Den består i att upprepa en operation som kallas merge. Den operationen undersöks i labben.

2 Syfte

Du kommer i labben att använda dig av iteration och selektion med filhantering.

3 Läsanvisningar

Du ska ha färdigställt dina tidigare labbar innan du fortsätter med denna.

*E-post: martin.kjellqvist@miun.se.

4 Genomförande

Du ska skapa ett program som tar som argument ett respektive två filnamn.

Du startar ditt program med

```
\$java Sort filnamn1 [filnamn2]
```

filnamn1 och filnamn2 behandlas utav ditt program på följande vis:

```
public class CmdArgument{
  public static void main(String [] arg){
    String filnamn1 = arg[0];
  String filnamn2 = arg[1];
  }
}
```

Om namnen inte ges som argument till programmet finns inte arrayelementen `arg[0]` och/eller `arg[1]`. Skriv ditt program så att det ger ett informativt felmeddelande om filnamnen inte angetts. Testa detta innan du fortsätter.

Dela upp ditt program i lämpliga, förslagsvis statiska, metoder. Metoderna ska göra *en* väl avgränsad sak. Skicka metoderna de argument de behöver för att lösa sin uppgift.

Du kan förutsätta att filerna innehåller heltal. Du behöver inte ha felhantering för icke-numeriska värden.

1. En fil innehåller heltal. Du skall författa en funktion som avgör om heltalen i filen är i ordning eller inte. Testprogrammet ska avgöra om filen A1 nedan är sorterad eller inte. En algoritmisk beskrivning ges i avsnitt 6. Försök att komma på en metod på egen hand innan du tittar på algoritmen.

2. Två filer, A och B är sorterade. (Kontrollera detta med den funktion du författat i uppgiften ovan.) Utifrån filerna ska du skapa en fil som innehåller samtliga element i sorterad ordning. Denna operation kallas för en merge. En algoritmisk beskrivning ges i avsnitt 6. Försök att implementera den på egen hand innan du tittar på algoritmen.

   Filen A hittar då på

   `http://w3.miun.se/dt028g/attach/161/A`.

Filen B finner du på

`http://w3.miun.se/dt028g/attach/162/B`.

Filen A1 finner du på

`http://w3.miun.se/dt028g/attach/163/A1`.

5 Examination

Om du läser på campus kontrollerar du med din handledare att din lösning är rimlig innan du lämnar in uppgiften. Om du läser på distans gör du en extra koll att din lösning uppfyller kraven i uppgiften.

Då du löst uppgiften laddar du upp din källkod till inlämningslådan i lärplattformen.

6 Algoritmer

Algorithm 1 Avgör om en fil är sorterad

Input: Filen A innehåller numeriska värden
Output: Filen A är sorterad

```
a ← readValue(A)
while notEndOfFile(A) do
    b ← readValue(A)
    if a > b then
        return false
    end if
    a ← b
end while
return true
```

Referenser

Algorithm 2 Merge med två filer

Input: Filerna A och B är sorterade enligt algoritm 1

Output: Filen C innehåller samtliga värden från A och B i sorterad ordning

```
a ← readValue(A)
b ← readValue(B)
while notEndOfFile(A) And notEndOfFile(B) do ▷ Avgör vilket värde som ska skrivas till C
    if a < b then
        Skriv a till C
        a ← readValue(A)
    else
        Skriv b till C
        b ← readValue(B)
    end if
end while                    ▷ A eller B är slut, skriv klart båda
while notEndOfFile(A) do
    Skriv a till C
    a ← readValue(A)
end while
while notEndOfFile(B) do
    Skriv b till C
    b ← readValue(B)
end while
```