Användning av datastrukturer

Algoritmer och datastrukturer

Obligatorisk Laboration nr 1

Syfte

Att träna användning av Javas standardbibliotek för datastrukturer och algoritmer.

Litteratur

Weiss kap. 6, Skansholm 17

Utvecklingsmiljö

BlueJ eller eclipse rekommenderas. En kort introduktion till eclipse finns på kurshemsidan.

Färdig programkod

Given programkod som behövs i uppgifterna finns på kurshemsidan under fliken **Laborationer**.

Genomförande och redovisning

Uppgifterna är obligatoriska.

- Redovisning sker i grupper om två personer – inte färre, inte fler.
- Varje grupp skall självständigt utarbeta sin egen lösning.
  Samarbete mellan grupperna om övergripande principer är tillåtet, men inte plagiering.

Gå till kurshemsidan och vidare till **Laborationer->Hur man använder Fire-systemet**.

- Om du inte registrerat labbgrupp i Fire än: Gör laboration 0.0 innan du fortsätter!
- Redovisa filerna `AngloTrainer.java`, `WordLists.java` med dina lösningar, samt `README.txt` med allmänna kommentarer om lösningen. Bifoga även källtextfiler med eventuella komparatorklasser.
- Senaste redovisningsdag, se Fire.

Kodningsråd

Använd om möjligt Javas förenklade for-loop för genomlöpning av listor och mängder. Detta förenklar koden och gör den betydligt mer lättläst än med iteratorer, särskilt i uppgift 2.

Uppgift 1

Konstruera ett program som låter användaren bilda olika ord ur slumpmässiga bokstavsföljder. Orden skall kontrolleras mot en ordlista. För varje bokstavsföljd skall dessutom programmet självt presentera alla möjliga ord i ordlistan som kan bildas ur uppsättningen. En dialog med programmet kan se ut så här (understruken text skrivs av användaren):

```
> anglotrainer
21654 words loaded from dictionary.txt
The random letters are: byota
Try to build as many words from these letters as you can!
by
ok!
to
ok!
at
ok!
bat
ok!
tab
ok!
bot
Your suggestion was not found in the dictionary.
I found:
at
bat
bay
boa
boat
boy
by
oat
tab
tao
to
toy
```

Krav

- Programmet skall skriva ut en följd *S* av lika många slumpmässigt valda bokstäver i intervallet a-z som ordlistans längsta ord.
- Användaren får skriva in förslag på engelska ord som kan bildas enligt reglerna
  1. Endast bokstäver i *S* får användas.
  2. Varje bokstavsförekomst i *S* får användas högst en gång i det bildade ordet. Finns det t.ex. två förekomster av bokstaven t så får det bildade ordet innehålla högst två t.
- Föreslagna ord som uppfyller 1 och 2 samt finns i ordlistan godkänns och användaren får fortsätta.
- Användaren skall när som helst kunna avbryta dialogen genom att trycka ctrl-Z (ctrl-D i Linux) varvid programmets egna förslag presenteras.
- Om användaren föreslår ett ord som inte är korrekt bildat enligt reglerna 1 och 2, eller om ordet inte finns i ordlistan, avbryts användardialogen och programmets egna förslag (utan duplikat) presenteras i bokstavsordning. De ord som skrivs ut av programmet skall givetvis uppfylla kraven ovan. *OBS! Lämpligt meddelande skall skrivas ut så att användaren får veta varför ett ord underkänns – skilj alltså fallet med ett felbildat ord m.a.p. reglerna 1 och 2 från fallet med ett korrekt bildat ord som saknas i ordlistan.*
- Javas klasser och algoritmer skall användas så långt som möjligt.
- Längden på den egenutvecklade programkoden bör knappast överstiga drygt en A4-sida.

- Dela in din lösningsalgoritm i lämpliga metoder. Komplettera dessutom givna metodstubbar.
- För att använda metoden `includes` som beskrivs nedan behöver du kunna sortera tecknen i en sträng. Implementera därför metoden

  ```
  private String sort(String s)
  ```

  Tips: uttnytta klassmetoden `java.util.Arrays.sort`.

Några användbara klasser och algoritmer

Nedan följer några klasser och algoritmer som är användbara för att lösa uppgiften.

Datastrukturer

`String`
: Standardklass för stränghantering.

`HashSet<`*elementtyp*`>`
: Generisk oordnad mängd baserad på hash-teknik.

`TreeSet<`*elementtyp*`>`
: Generisk ordnad mängd baserad på binärt sökträd.

`java.util.Random`
: Används för generering av pseudoslumptal.

Algoritmer

`java.util.Arrays.sort()`
: Sorteringsfunktion för fält.

`public boolean includes(String a,String b)`
: Testar inklusionsrelation mellan två strängar. Ej standardmetod. Bifogas.
Låt #(x,s) = antalet förekomster av tecknet x i strängen s.
`includes(a,b)` returnerar `true` om, och endast om, för varje tecken x i `b`, #(x,`b`) ≤ #(x,`a`); annars `false`. Ett nödvändigt villkor före anrop av metoden är att teckenföljderna i båda strängarna är sorterade lexikografiskt (bokstavsordning). En testmetod som demonstrerar `includes` finns i den givna programkoden.

Engelsk ordlista

En liten engelsk ordlista finns i filen `dictionary.txt`, en längre på http://www-01.sil.org/linguistics/wordlists/english/. Placera filen med ordlistan i projektets rotkatalog, om du använder eclipse är det katalogen som innehåller katalogerna `src` och `bin`.

Programkod

Given programkod, ordlista m.m. finns på kursens hemsida. Ett programskelett finns i `AngloTrainer.java`. Filen innehåller bl.a. metoden `includes` ovan.

Uppgift 2

Konstruera ett program som producerar tre olika typer av ordlistor från en textfil. Den första listan skall vara en alfabetisk förteckning över alla ord som förekommer i filen. Efter varje ord skall skrivas ut hur många förekomster som hittades i texten. I nästa lista skall orden (utan duplikat) skrivas ut i baklängesordning (se nedan) [1]. Den tredje listan skall gruppera orden efter avtagande frekvens. Inom varje frekvensgrupp skall orden ordnas alfabetiskt. De tre listorna skall skrivas ut i var sin textfil.

Exempel
Om man ger följande avsnitt ur filen `provtext.txt` som indata till programmet

```
provtext.txt

The way to teach Java as a first programming language is
still uncharted. Those of us who have gone through a change
of first programming language more than once know that the
one guiding beacon is not to assume that the pattern of
teaching a new language will follow that of the old. Java is
no exception to this rule.
...
```

så skall utdatafilerna (avkortat) innehålla:

Alfabetisk ordlista

```
alfaSorted.txt

a     3
as     1
assume     1
beacon     1
change     1
exception     1
first     2
follow     1
gone     1
guiding     1
have     1
is     3
java     2
know     1
language     3
...
```

Baklängesordlista

```
backwardsSorted.txt
                    a
                 java
            uncharted
                  old
                once
            language
              change
                 the
                rule
              assume
                 one
                gone
                more
               those
                have
                  of
             guiding
            teaching
          programming
```

Frekvensordlista

```
frequencySorted.txt

    4:
       of
       the
    3:
       a
       is
       language
       that
       to
    2:
       first
       java
       programming
    1:
       as
       assume
       beacon
       change
       exception
    ...
```

Några användbara standardklasser

Javas standardklasser för mängder och mappar är användbara för att lösa problemet. Några tips:

- Eftersom `Set` och `Map` är generiska finns inget som hindrar att mängder kan vara element i mappar och tvärtom.
- Frekvensordlistan skall sorteras efter avtagande frekvenser. Utnyttja att `TreeMap` har en konstruktor som tar ett objekt av typen `Comparator` som parameter. Definiera en lämplig

[1] Det är inte så konstigt som det låter, det finns faktiskt en officiell svensk baklängesordlista.

subklass till den och utnyttja möjligheten att jämföra som du vill!
Ex. Denna jämförare ger vanlig lexikografisk ordning:

```
public class StringComparator implements Comparator<String> {
    public int compare(String s1,String s2) {
        return s1.compareTo(s2);
    }
}
```

- Metoderna `keySet`, `values` och `entrySet` är användbara vid iterering i mappar.

Programkod

Given programkod m.m. finns på kursens hemsida. Ett programskelett finns i `WordLists.java`. Filen innehåller bl.a. en färdig metod för inläsning av ord.

Exempeltext

En kort engelsk text finns i filen `provtext.txt`.

Lycka till!