

Opgave i Java-programmering

Denne opgave går ud på at skrive en klasse, som hedder *StringTokenizer*.

Klassen har til formål at opdele tekststrengene i mindre dele efter nærmere angivne kriterier.

Ideen er, at tekststrengene kan opfattes som udelukkende bestående af *tokens* og *delimiters*. Hvis vi ser på følgende tekststreng:

"Gid du var i Skanderborg og blev der, kære Peter."

kan man opfatte den således, at den består af ti tokens (ordene), og at der indgår tre forskellige delimiters: " " (blank), "," (komma) og "." (punktum).

Mængderne af forskellige tegn, som kan indgå i tokens og delimiters er med andre ord disjunkte: en delimiter kan ikke indgå i et token, og et tegn som indgår i et token kan ikke samtidig være en delimiter.

StringTokenizer har følgende attributter:

```
String  tekst;      //tekst, der skal opdeles i tokens
String  delim;      //streng som indeholder de aktuelle delimiters
int     indeks;      //pegepind til aktuelt tegn i tekst
```

Opgave 1

Du skal nu implementere klassens constructors, som har følgende signaturer:

```
StringTokenizer(String source);
StringTokenizer(String source, String delimiters);
```

som kan initialisere attributterne således:

tekst : initialiseres med *source*
delim : sættes til parameteren *del* eller til strengen " \n " (linjeskift, blank), hvis parameteren mangler
indeks : sættes til 0 og udpeger dermed det første tegn i *tekst*

Opgave 2

Metoden

```
boolean isDelimiter(char tegn);
```

skal returnere *true*, hvis *tegn* er en delimiter, dvs. indeholdt i strengen *delim*, og ellers returneres *false*.

Opgave 3

Der er flere tokens i *tekst*, hvis *indeks* ikke har passeret sidste tegn i *tekst* og der findes mindst ét tegn efter *indeks*, som ikke er en delimiter.

Skriv metoden

```
boolean hasMoreTokens();
```

som gennem returværdien kan vise, om der er flere tokens tilbage.

Opgave 4

Nu skal du implementere klassens to centrale metoder:

```
String nextToken();  
String nextToken(String delimiters);
```

Den første returnerer næste token ud fra de delimiter tegn, der findes i *delim*.

Den anden opdaterer først attribut *delim*, således at den indeholder parameteren *del* og returnerer derefter næste token med udgangspunkt i den nye værdi af *delim*. *delim* forbliver uændret indtil næste kald af denne metode med en ny delimiter-streng.

Næste token findes ved at tage udgangspunkt i pegepinden *indeks*' placering, således at:

1. Hvis der ikke er flere tokens returneres en tom streng
2. Alle delimiters før næste token overspringes.
3. Alle tegn herefter, der ikke er en delimiter, opsamles i det token, der skal returneres
4. Når en delimiter herefter mødes, eller sidste tegn er passeret, standser indeks ved dette tegn og token returneres

Eksempler:

1.
tekst = "Gid du var i Skanderborg og blev der, kære Peter."
delim = " ,. " (blank, komma, punktum)
indeks = 6 (ved den blanke mellem "du" og "var")
Der returneres "var" og *indeks* får værdien 10.

2.
tekst og *delim* som i eksempel 1
indeks = 36 (ved kommaet)
Der returneres "kaere" og *indeks* får værdien 43.

3.
tekst og *delim* som i eksempel 1
indeks = 49 (ved punktummet)
Der returneres "" (tom streng) og *indeks* får værdien 50 (sidste tegn er passeret).

Eksemplerne gælder for begge versioner af `nextToken` efter eventuel opdatering af attribut *delim*.

Opgave 5.

Metoden

```
int countTokens();
```

returnerer antallet af tokens, der er tilbage i strengen med de aktuelle delimiters, der findes i attribut *delim*.

Implementér metoden.