Høgskolen i Agder
Avdeling for økonomi- og samfunnsfag

E K S A M E N

Emnekode:	**IS-207**
Emnenavn:	**Algoritmer og datastrukturer**
Dato:	22. mai 2007
Varighet:	0900-1300
Antall sider inkl. forside:	5
Målform:	Norsk
Tillatte hjelpemidler:	Alle trykte og skrevne
Merknader:	Les gjennom hele oppgaven før du begynner å svare. Klassen java.io.File må brukes for å løse oppgaven, nødvendige utdrag av dokumentasjonen finnes i vedlegg 1. Du kan selvfølgelig bruke andre klasser fra standardbiblioteket også.

Innledning

Unix-kommandoen ”du” (forkortelse for disk usage) summerer opp og skriver ut forbruk av diskplass per katalog. For hver katalog er utskriften summen av diskforbruk for alle filer og kataloger i katalogen.

Et eksempel: Anta katalogstruktur som vist i figuren.

- katalogen *IS-207* inneholder katalogene *Fagstoff*, *Innlevering* og *Planer og pensum*, men ingen filer.
    - Katalogen *Fagstoff* inneholder en fil på 75 byte, og katalogene *Eksempler* og *Forelesninger*
        - Katalogen *Eksempler* inneholder 4 filer på til sammen 2000 byte
        - Katalogen *Forelesninger* inneholder 15 filer på til sammen 3500 byte
    - Katalogen *Innlevering* inneholder 40 filer på til sammen 35000 byte
    - Katalogen *Planer og pensum* inneholder 2 filer på til sammen 200 byte

Kommandoen ”du IS-207” vil skrive ut:

```
2000   IS-207\Fagstoff\Eksempler
3500   IS-207\Fagstoff\Forelesninger
5575   IS-207\Fagstoff
35000  IS-207\Innlevering
200    IS-207\Planer og pensum
46275  IS-207
```

Oppgave 1 (teller 20 + 5 + 5 + 10%)

a) Skriv et program i Java som skriver ut bruk av diskplass på samme måte som ”du”. Du kan ta utgangspunkt i koden i vedlegg 2.
Du trenger klassen java.io.File. Javadoc for de metodene du trenger i File finnes i vedlegg 1.

b) Hva kan du si om tidsforbruket til programmet ditt? Anta at den tidkrevende delen av programmet er å finne størrelsen til hver fil.
(File.length() krever et oppslag på disken. Tiden det tar er konstant.)

c) Hva slags struktur danner File-objektene, og hva slags traversering av strukturen er det programmet ditt gjør?

d) I Unix filsystemer kan man legge inn en symbolsk lenke til en fil (snarveier i Windows er noe lignende) som et alternativt navn på filen. Man kan også lage symbolske lenker til kataloger. I praksis betyr det at samme fil eller katalog kan ligge lagret i flere kataloger, eller flere ganger i samme katalog under forskjellige navn. Hvilken effekt vil dette kunne ha på resultatet til programmet ditt? Hva ville skje i eksempelet over hvis vi la inn en symbolsk link til *IS-207* i *Forelesninger*?

Oppgave 2 (teller 10+30+20%)

Hvis vi skal rydde på disken kan det være interessant å finne de største filene som ligger i en katalog eller en av underkatalogene. Kommandoen ”du” skriver bare ut hvor mye diskplass alle filene i en katalog bruker. Det vi ønsker oss er en utskrift av hvor mye plass hver fil bruker, sortert på størrelse, noe slikt:

```
2035   IS-207\Innlevering\Diger.zip
2011   IS-207\Eksempler\StortEksempel.jar
1978   IS-207\Innlevering\Stor.jar
...
```

a) Du trenger en datastruktur hvor du kan lagre alle filene for å lage en slik utskrift. Hvilken datastruktur vil du bruke, og hvorfor?

b) Skriv et nytt program som skriver ut en liste av alle filer sorter på størrelse.

c) Hva kan du si om tidsforbruk og minneforbruk for denne versjonen av programmet nå?

Vedlegg 1: java.io.File

public class **File**
extends Object
implements Serializable, Comparable<File>
An abstract representation of file and directory pathnames.

public **File**(String pathname)

Creates a new File instance by converting the given pathname string into an abstract pathname. If the given string is the empty string, then the result is the empty abstract pathname.

Parameters:
pathname - A pathname string
Throws:
NullPointerException - If the pathname argument is null

public boolean **isDirectory**()

Tests whether the file denoted by this abstract pathname is a directory.

Returns:
true if and only if the file denoted by this abstract pathname exists *and* is a directory; false otherwise

public boolean **isFile**()

Tests whether the file denoted by this abstract pathname is a normal file. A file is *normal* if it is not a directory and, in addition, satisfies other system-dependent criteria. Any non-directory file created by a Java application is guaranteed to be a normal file.

Returns:
true if and only if the file denoted by this abstract pathname exists *and* is a normal file; false otherwise

public long **length**()

Returns the length of the file denoted by this abstract pathname. The return value is unspecified if this pathname denotes a directory.

Returns:
The length, in bytes, of the file denoted by this abstract pathname, or 0L if the file does not exist. Some operating systems may return 0L for pathnames denoting system-dependent entities such as devices or pipes.

public File[] **listFiles**()

Returns an array of abstract pathnames denoting the files in the directory denoted by this abstract pathname.

If this abstract pathname does not denote a directory, then this method returns null. Otherwise an array of File objects is returned, one for each file or directory in the directory. Pathnames denoting the directory itself and the directory's parent directory are not included in the result.

There is no guarantee that the name strings in the resulting array will appear in any specific order; they are not, in particular, guaranteed to appear in alphabetical order.

Returns:
An array of abstract pathnames denoting the files and directories in the directory denoted by this abstract pathname. The array will be empty if the directory is empty. Returns null if this abstract pathname does not denote a directory, or if an I/O error occurs.

public String **toString**()

Returns the pathname string of this abstract pathname. This is just the string returned by the getPath() method.

Overrides:
toString in class Object
Returns:
The string form of this abstract pathname

Vedlegg 2: DiskUsage.java

```
import java.io.File;

/** Utgangspunkt for Java-program med samme funksjonalitet som Unix-kommandoen du */
public class DiskUsage {
   // dine metoder her

   public static void main(String[] args) {
      if (args.length == 1) {
         File f = new File(args[0]);

         // din kode her
      }
      else System.out.println("Usage: java DiskUsage <directory name>");
   }
```