# String-manipulasjon og I/O i Java

INF1000 h14 - Hans Petter Taugbøl Kragset
hpkragse@ifi.uio.no

# Plan

- Viktige metoder i String
- Repetisjon av I/O
- Oppgaver og eksempler

# Viktige metoder

- charAt
- compareTo[IgnoreCase]
- contains
- endsWith
- equals[IgnoreCase]
- indexOf
- lastIndexOf
- length
- isEmpty
- replace
- split
- startsWith
- substring
- toCharArray
- to[Lower][Upper]Case
- trim

# Mal for slidene

- `signatur(type argument)`
- forklaring

- `eksempel`
- Spørsmål
- `fasit`

# String

# length

- `int length()`
- Returnerer lengden til Stringen

```
String a = "abcde";
int len = a.length();
```

- Hva er veriden av *len*?
- 5

# isEmpty

- `boolean isEmpty()`
- Returnerer true hvis Stringens lengde er 0
- **IKKE** det samme som NULL

```
String a = "";
boolean b = a.isEmpty();
```

- Hva er verdien av *b*?
- `true`

# charAt

- `char charAt(int index)`
- Returner char-verdin på en gift indeks
- Indeks varierer fra 0 til *length()* - 1
  - helt tilsvarende array-indeksering

```
String a = "abc";
char b;
b = a.charAt(2);
```

- Hva er verdien av b?

- `'c'`

# equals

- `boolean equals(Object anObject)`
- Returnerer true dersom dette objektet (*this*) og objektet *anObject* inneholder akkurat den samme sekvensen av tegn.
- Fancy måte å si "True hvis strengene er like" på.

```
String a, b;
a = "abcde";
b = "abc de";
boolean c = a.equals(b);
```

- Hva er verdien av *c*?
- `false`

# equalsIgnoreCase

- `boolean equalsIgnoreCase(String anotherString)`
- Gjør det samme som *equals*, men anser 'A' og 'a' for å være like

```
String a, b;
a = "AbCde";
b = "abcDe";
boolean c = a.equalsIgnoreCase(b);
```

- Hva er verdien av *c*?
- `true`

# compareTo
# compareToIgnoreCase

- `int compareTo(String anotherString)`
- `int compareToIgnoreCase(String str)`
- Returnerer en int basert på hvilken String som kommer leksikografisk først.
- Negativ hvis this kommer før argument-Stringen, positiv om det er motsatt, 0 hvis strengene er like (som equals).
  - IgnoreCase anser 'A' og 'a' for å være like

```
String a, b;
a = "ape";
b = "ake";
int comp = a.compareTo(b);
```

- Hva er verdien av *comp*? (positiv, negativ eller 0?)
- `positiv`

# mer om compareTo

- Brukes ofte for å sortere
- Jeg bruker alltid litt tid på å huske om det er større enn 0 eller mindre eller hva det er…
- Pass på hva som sammenlignes med hva
  - *a.compareTo(b)* er jo motsatt av *b.compareTo(a)*

# `startsWith`

- `boolean startsWith(String prefix)`
- Returnerer true hvis *this* begynner med *prefix*.

- 
```
String a, b;
a = "abcdefg";
b = "abc";
boolean c = a.startsWith(b);
```

- Hva er verdien av *c*?
- `true`

# endsWith

- `boolean endsWith(String suffix)`
- Returnerer true hvis *this* slutter med *suffix*.

```
String a, b;
a = "abcdefg";
b = "abc";
boolean c = a.endsWith(b);
```

- Hva er verdien av *c*?
- `false`

# indexOf

- `int indexOf(int ch)`
- `int indexOf(int ch, int fromIndex)`
- `int indexOf(String str)`
- `int indexOf(String str, int fromIndex)`

- Finner posisjonen til en gitt char eller String
- Hvorfor står det "int ch"?
  - *char er en slags int*

# indexOf

```
String a = "abcabc";
char b = 'c';
int i = a.indexOf(b);
int j = a.indexOf("b");
int k = a.indexOf("b", 3);
```

- Hva er verdien av *i, j og k*?

- `i = 2`
- `j = 1`
- `k = 4`

# lastIndexOf

- `int lastIndexOf(int ch)`
- `int lastIndexOf(int ch, int fromIndex)`
- `int lastIndexOf(String str)`
- `int lastIndexOf(String str, int fromIndex)`
- Finner *den siste* posisjonen til en gitt char eller String

# lastIndexOf

```
String a = "cbcbc";;
int i = a.lastIndexOf("c");
int j = a.lastIndexOf("cbc");
int k = a.lastIndexOf("c", 3);
```

- Hva er verdien av *i, j og k*?

- `i = 4`
- `j = 2`
- `k = 2`

# substring

- `String substring(int beginIndex)`
- `String substring(int beginIndex, int endIndex)`

- Returnerer en String som er en del av *this.*

- *`String a = "abcdefgh";`*
- *`string b = a.substring(4, 6);`*

- Hva er verdien av *b*?

- `"ef"`

# Mer om substring

- `String substring(int beginIndex)`
- *Fra og med beginIndex,* til slutten av Stringen

- `String substring(int beginIndex, int endIndex)`
- *Fra og med beginIndex,* til *og ikke med endIndex.*
- Dette gjør at vi kan vite lengden på Stringen
  - `a.substring(4, 6);`
  - gir en resultatstreng med lengde 2 (4 - 6).

# replace

- `String replace(char oldChar, char newChar)`
- Returnerer en ny String der alle forekomster av charen *oldChar* er byttet ut med charen *newChar*.

```
String a = "abcdefg";
String b = a.replace('c', 'x');
```

- Hva er verdien av *b*?
- `"abxdefg"`

# Mer om replace

- `String replace(CharSequence target, CharSequence replacement)`
- CharSequence er det samme som String
- Bytter altså ut en substring, og ikke bare en og en char.

```
String a = "abc";
String b = a.replace("b", "kalkun");
```

- Hva er verdien av *b*?

- `"akalkunc"`

# contains

- `boolean contains(CharSequence s)`
- Returnerer true hvis *this* inneholder *Stringen s*.
  - CharSequence er noe rare greier, tenk på det som en String.

```
String a = "abcdefg";
b = "bc";
boolean c = a.contains(b);
```

- Hva er verdien av *c*?
- `true`

# split

- `String[] split(String regex)`
- Returnerer en String-array
- Splitter opp *this* og setter del for del inn i arrayen

```
String a = "ab:cd:ef:g";
String[] b = a.split(":");
```

- Hva er verdien av *b*?
- `{"ab", "cd", "ef", "g"}`

# Mer om split

- `String[] split(String regex)`
- Vi kan splitte med hva som helst
- De som vil kan lese mer om regex for å lære mer om det
- Mastering Regular Expressions - lån på biblioteket.

# toLowerCase
# toUpperCase

- `String toLowerCase()`
- `String toUpperCase()`
- Returnerer en String der alle tegn er store eller små

```
String a, b;
a = "abcdefg";
b = "abc";
a = a.toUpperCase();
b = b.toLowerCase();
```

- Hva er verdien av *a* og *b*?

- `"ABCDEFG"`
  `"abc"`

# trim

- `String trim()`
- Returnerer en ny String der blanke tegn i starten og på slutten er fjernet

```
String a, b;
a = "      ab c d  ";
String b = a.trim();
```

- Hva er verdien av *b*?
- `"ab c d"`

# `toCharArray`

- `char[] toCharArray()`
- Returnerer en char-array som inneholder alle tegnene fra *this*.

```
String a = "ab def";
char[] c = a.toCharArray();
```

- Hva er verdien av *c*?

- `{'a', 'b', ' ', 'd', 'e', 'f'}`

# String som returtype

- Returnerer et *nytt* String-objekt
- f eks:

```
String abc, sub;
abc = "abcdefg";
sub = abc.substring(2);
```

- *sub* er nå et *nytt* String-objekt laget av substring-metoden i String-objektet *abc*

- Det er ingen metoder i String som forandrer en String
  - Alle returnerer en ny String!

# Fremgangsmåte

- Hvilke metoder *kan* funke?
- Finn den metoden som gjør mest
- Kombiner metoder
- Tenk på rekkefølgen!

- Ta vare på returverdier
- Tenk på hva som kan gå galt
- Flere eksempler kommer

I/O

# I/O - plan

- Scanner
  - Fra terminal
  - Fra fil
- PrintWriter
  - Til fil

# Scanner

- `import java.util.Scanner;`
- Metodene vi trenger:
- `hasNextLine()`
- `nextLine()`

# Innlesing fra terminal

- `Scanner sc = new Scanner(System.in);`
- "System.in" er terminalen, der vi leser fra
- `String a = sc.nextLine();`
- *a* inneholder nå alt som ble skrevet i terminalen til *men ikke med* linjeskiftet på slutten (Enter)
- Linjeskiftet forsvinner!

# Innlesing av tall

- `Scanner sc = new Scanner(System.in);`
  `String a = sc.nextLine();`
- Hva gjør vi for å hente ut tall fra *Stringen a?*

- `Integer.parseInt(String s)`
- Henter int-verdien ut av en String
- `int b = Integer.parseInt(a);`

- `Double.parseDouble(String s);`
- Gjør det samme for komma-tall

# Innlesing fra fil

```
import java.util.Scanner;
import java.io.File;

File f = new File("filnavn.txt");
Scanner sc = new Scanner(f);
```

- Nå er *sc* en Scanner som leser fra filen "filnavn.txt"

- Vi kan bruke følgende metoder:
  - `boolean hasNextLine()`
    - Returnerer true hvis det fins flere linjer igjen i filen
    - *Ingenting blir lest*
  - `String nextLine()`
    - Leser og returnerer neste linje av filen
  - Vi må huske noe spesielt også:
    - `throws Exception`

# Innlesing fra fil forts.

```
import java.util.Scanner;
import java.io.File;

File f = new File("filnavn.txt");
Scanner sc = new Scanner(f);

while(sc.hasNextLine()) {
  System.out.println(sc.nextLine());
}
```

# PrintWriter

- `import java.io.PrintWriter;`
- Metodene vi trenger:
- `println()`
- `print()`
- `close()`

# Utskrift til fil

```
import java.util.PrintWriter;

String a = "filnavn.txt";
PrintWriter pw = new PrintWriter(a);
```

- Nå er *pw* en PrintWriter som skriver til filen "filnavn.txt"
- For å skrive til fil bruker vi
  - `pw.println("Srevet til fil");`
- Hvis vi ikke vil ha med linjeskift på slutten kan vi bruke
  - `pw.print("Uten linjeskift");`
  - Vi må huske noe spesielt også:
    - `throws Exception`

# Utskrift til fil forts.

```
import java.util.PrintWriter;

  String a = "filnavn.txt";
  PrintWriter pw = new PrintWriter(a);

int i = 0;
while(i < 10) {
   pw.println("Verdien til i: " + i);
   i++;
}
pw.close();
```

# Eksempel

# Eksempel

- Vi har en fil med et litt rart format, og vi ønsker å plukke ut en del av innholdet.
- Filen er som følger:

```
HEI     DIN     KALKUN    HVORDAN
HAR_    DU_     DET   _   ?   _JEG_  HAR  _
DET_BRA   _!
```

- Vi ønsker å lese ut setningen som står der, uten alle "_" og ekstra mellomrom. Hva kan vi gjøre?

1. Hvilke metoder *kan* funke?
2. Finn den metoden som gjør mest
3. Kombiner metoder
4. Tenk på rekkefølgen!

# Eksempel

- `HEI _ DIN _ KALKUN_ HVORDAN_ HAR_ DU_ DET _ ? _JEG_ HAR _ DET_BRA _!`
- substring, split, indexOf, lastIndexOf, toCharArray…. (mer?)
- `split("_");`
- Da har vi en array med ord, men mellomrommene er der enda…
- `trim();`
- Fullstendig kode på neste slide

```
import java.util.PrintWriter;
import java.util.Scanner;
import java.io.File;

public class StringManip {
  public static void main(String[] args) {

    Scanner sc = new Scanner(new File("filnavn.txt"));
    String tekst = sc.nextLine();

    String[] ord = tekst.split("_");
    String setning;

    for(int i = 0; i < ord.length; i++) {
      setning += ord[i].trim() + " ";
    }

    PrintWriter pw = new PrintWriter("filnavn.txt");
    pw.println(setning);
    pw.close();
  }
}
```

- Output:

```
$> HEI DIN KALKUN HVORDAN HAR DU DET ? JEG HAR DET BRA !
```

- Spørsmål?

# Andre løsningsforslag?

- Fra salen?
- Vi kunne brukt andre fremgangsmåter!
- Eksempel:

```
• int ind1 = indexOf('_');
• ord[i] = line.substring(0, ind1);
• ord[i] = ord[i].trim();
• line = line.substring(ind1 + 1);
```

  - Dette kan gå i løkke
- Bruk kreativiteten deres!

# Oppgaver!!!

Hvis du leser dette på nettet skal det ligge en fil "Repkurs-oppgaver.pdf" ved siden av som inneholder oppgaver. Hvis ikke, send mail (se første slide).