Praktisk information

Til *lektion 6, 11. oktober*, vil vi benytte kapitel 10 fra *Savitch & Mock: "Java - An Introduction to Problem Solving & Programming", 7th Global Edition.*

Den nye pdf og kodeeksempler ligger på BlackBoard under **Værktøjer og Litteratur->File I/O, Streams, Exceptions og TCP**.

Emner til lektion 5 (27/9)

Opsamling på *Arv, Abstrakte klasser og Polymorphism*.

Interfaces og Polymorphism.

Forberedelse til lektionen

Fra kap 9:

- Afsnit 9.1 er en ganske kort introduktion til begrebet *Polymorphism.* Læs det både før i kigger på det øvrige stof og bagefter. Giver det mere mening efter at de øvrige afsnit er behandlet?
- Eksperimenter med eksemplet i afsnit 9.2 (Listing 9.1 – 9.7). Det er et stort eksempel, så der skal bruges noget tid på det:
    - Studer hvordan klassen *Staff* indeholder et array af den abstrakte klasse *StaffMember*, som udfyldes af instancer af de konkrete klasser, der arver fra *StaffMember*, og hvordan den abstrakte metode *double pay()* implementers forskelligt i de konkrete klasser.
    - Bemærk brugen af *cast* i slutningen af constructoren i *Staff* , fx
      `((Hourly) staffList[3]).addHours(40);`.
      Hvorfor er det nødvendigt af *cast'*e til *Hourly*?
    - Prøv at forstå brugen af polymorphism i `void payDay()`.
- Afsnit 9.3 og 9.4 er en alt for kort beskrivelse af *Interfaces* og de viste eksempler (Listing 9.8 – 9.10) er efter min mening ikke særlig gode. Læs afsnittene igennem så vi kan bruge ret lang tid på at snakke om interfaces og programmere et par opgaver under lektionen.

Implementer videre på arve-hierarkiet i den obligatoriske opgave.

Bearbejdelse af dagens emner

Diskussion af erfaringerne med *Arv og Polymorphism* på baggrund af *Shape* opgaven.
Gennemgang af kapitel 9.1 og 9.2, så i får styr på begreberne *Late Binding* og *Polymorphism*.

Så kigger vi grundigt på *interface* og ser noget af det, som det kan bruges til, fx en opgave i klassen (fra eksamen 1. sem IT/SE E12):

Opgave 3. Polymorphism 30 points

Udleveret kode: `opg3_poly.Playable.java` *(interface),*
`opg3_poly.PolymorphismTester.java` *(Skelet til driver klasse)*. **Se opg3_poly.zip.**

`Playable.java` er et interface, som erklærer en enkelt metode, `public void play()`.

Opg. 3a (15 points)

Der skal implementeres 3 forskellige klasser, som implementerer dette interface. De tre klasser skal hver repræsentere *"noget man kan spille"* eller *"noget som kan spille"*, fx *Skak, Tennis* og *Klaver*.

De tre klasser skal ikke implementeres, så de rent faktisk kan spille. Implementationerne af `play()`-metoden skal blot skrive en besked på `System.out`, fx *"I'm playing Chess"*.

Opg. 3b (15 points)

Færdiggør implementationen af `public static void main()`-metoden i `PolymorphismTester`, så én ny instance af hver af de tre *Playable*-klasser tilføjes til den erklærede *ArrayList*.

I et loop skal `play()`-metoden i hvert element i listen kaldes.

Eksempel på udskrift:

```
I'm playing Piano!
I'm playing Tennis!
I'm playing Chess!
```

Hvis der er tid, tager vi *PP9.5* og *PP9.1*.

Og hvordan kan det så indgå i Matador-spillet?

Emner til lektion 6 (Tirsdag 4/10)

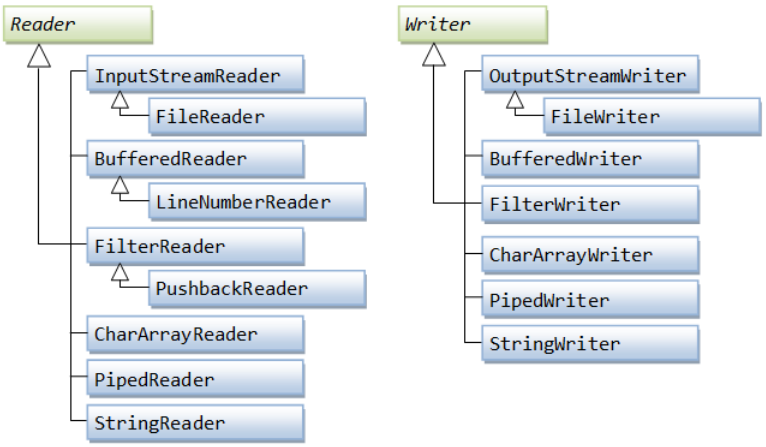
File I/O, Streams og Exceptions

Forberedelse til næste lektion

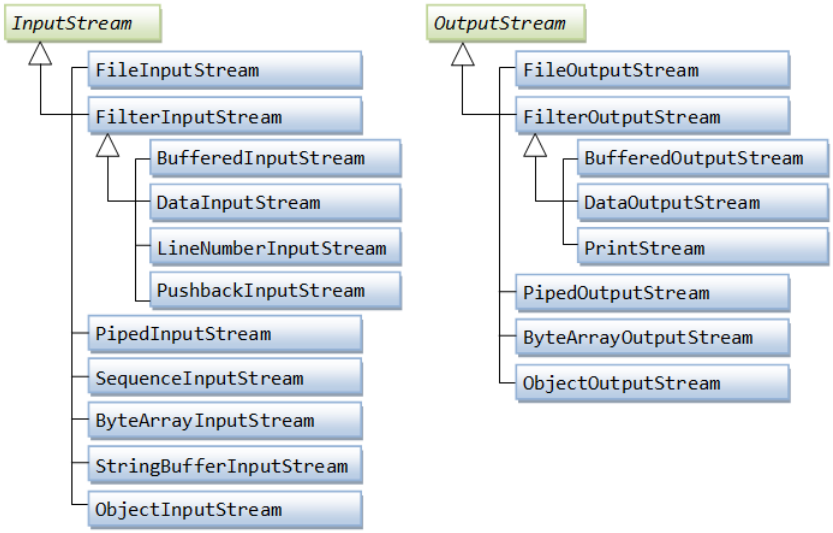
Som supplement til Savitch & Mock, kap 10, er her 2 illustrationer, som viser nogle vigtige *Streams* og *Readers/Writers*:

Character based I/O:

NB*: java.util.Scanner* og *java.io.PrintWriter* mangler. Dem skal vi også snakke om.

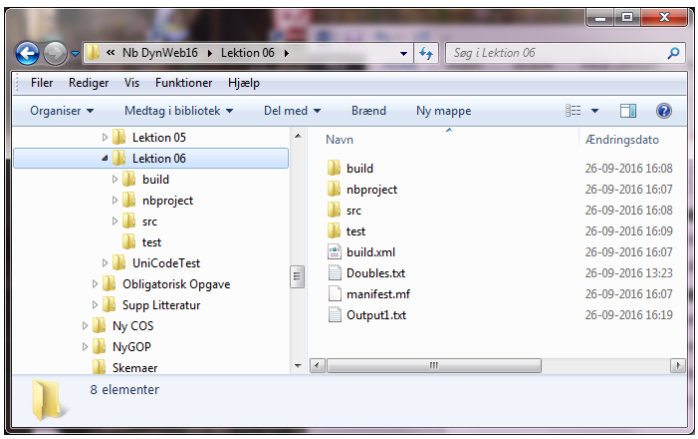


Byte based I/O:



Forbered følgende fra Savitch & Mock, kap 10.

- **Afsnit 10.1** gennemlæses med henblik på forståelsen af forskellene mellem *Tekst*- og *Binære*-filer.
- **Afsnit 10.2** studeres grundigt, inkl. Afprøvning af eksemplerne i NetBeans. Løs diverse *Self-Test Questions*, efterhånden som i arbejder jer igennem stoffet (resultaterne kan tjekkes fra side 849).
    - **Opgave 1** (baseret på *Listing 10.1* side 780 kombineret med *RECAP*-boksen side 785): Skriv et lille program, som kan skrive linjer læst fra keyboardet i en fil. Det skal være muligt at skrive videre i filen, selvom programmet har været lukket.
    - **Opgave 2** (variation af *Practice Programs 1* fra side 843)**:** Filen *Doubles.txt* indeholder 20 decimaltal. Skriv et lille program, som gennemlæser filen og konverterer hvert tal til en double. Når filen er læst udskrives summen af tallene, det mindste tal, det største tal, samt gennemsnittet af tallene til skærmen.
- **Afsnit 10.3** Læs siderne 789 – 795.
    - Bemærk at NetBeans benyttet den mappe der indeholder *src*-mappen, som *default directory*, hvorfor filer placeret her kan tilgås udelukkende med filnavnet.



    - Bemærk også at java kan håndtere UNIX path separatoren '/' også på Windows maskiner (så man kan slippe for "\\").

- **Studer** afsnittet *CASE STUDY, Processing a Comma-separated Value FileI*, side796 - 798 meget grundigt (det skal vi bruge i Matadorspillet). Specielt *Listing 10.4* er vigtig.
  NB: Linjen *String[] ary = line.split(",");* virker med andre skilletegn end ',. Fx vil *line.split("\t");* benytte tabulator karakteren som skilletegn.
- **Opgave 3** (lille del af en tidligere eksamensopgave for 2. sem IT/SE):
  - Hent filerne *Islands punktum.txt* og *Islands komma.txt* og kopier dem til default mappen i det NetBeansprojekt i aktuelt arbejder med.
  - Filerne indeholder forskellige oplysninger om danske beboede øer. Filernes oplysninger er de samme, bortset fra om der benyttes komma eller punktum som decimaltegn. Hver linje repræsenterer én ø med følgende data (adskilt af mellemrum):

    Navn Omkreds Areal Addresser Adr/km2 af datatyperne *(String, double, double, int, int).*

  - Hent zip-filen *erso_opg3.zip* og pak den ud i projektets *Source Packages (src)*. Indeholdet er en java *package* med klasserne:
    - *DanishIsland*: Objekter af denne klasse repræsenterer én ø. Studer koden, den er færdigimplementeret.
    - *DanishIslandFileReader*: Kodeskelet til læsning af ø-tekst fil. Studer koden og implementer det der mangler fra linje 42. I main() metoden kan der vælges om der skal benyttes komma eller punktum som decimaltegn på double-værdierne.
- **Afsnit 10.4** og **10.5** venter vi med til en senere lejlighed.
- **Afsnit 10.6** skal vi benytte under lektion 8.