

# **EKSAMENSOPPGAVE**

**Emnekode: DAT108**

**Emnenavn: Programmering og webapplikasjoner**

**Klasse: 2. klasse DATA / INF**

**Dato: 16. desember 2022**

---

Eksamensform: Skriftlig skoleeksamen (Wiseflow | FLOWlock)

Eksamenstid: 4 timer (0900-1300)

Antall eksamensoppgaver: 5

Antall sider (medregnet denne): 10

Antall vedlegg: Ingen

Tillatte hjelpemiddel: Ingen

Lærere:      Lars-Petter Helland (928 28 046) lph@hvl.no  
                 Bjarte Kileng (909 97 348) bki@hvl.no

**LYKKE TIL!**

## Oppgave 1 (20% ~ 48 minutter) – Lambda-uttrykk og strømmer

For å få full score må løsningene ikke være unødige kompliserte, og det bør brukes metodereferanser der det er mulig.

- a) Nedenfor ser du 5 ulike  $\lambda$ -uttrykk. Skriv en setning for hver av disse der de tilordnes en variabel. Det vi er ute etter er datatypen til variablene, f.eks. om det er en `Consumer<String>`, en `Function<Integer, String>`, etc. ..

```
a -> System.out.println(a)    // der a er en String
(a,b) -> a.compareTo(b)       // der a og b er String-er
a -> a * a                     // der a er et heltall
a -> a > 0                     // der a er et heltall
(a,b) -> a + b                 // der a og b er heltall
```

- b) Anta at du har metoden **utplukk**, metoden **main** og resultatet **[3, 6, 9]** av å kjøre main:

```
List<Integer> utplukk(List<Integer> liste, ??? b) {
    return liste.stream().filter(b).toList();
}

void main(...) {
    List<Integer> liste = List.of(1,2,3,4,5,6,7,8,9);
    List<Integer> resultat = utplukk(liste, ???);
    System.out.println(resultat);
}

[3, 6, 9]
```

Hva skal stå der det står ??? i parameterlisten (dvs. typen til den formelle parameteren b), og hva skal stå der det står ??? i metodekallet (dvs. den aktuelle parameteren)?

I oppgave c) og d) skal du jobbe med en liste av biler, f.eks.:

```
List<Bil> biler = List.of(
    new Bil("EK 12345", "Tesla model Y"),
    new Bil("EV 52345", "Tesla model Y"),
    ...
    new Bil("SV 12346", "Mazda 5"),
    new Bil("SU 24680", "Volvo 240"),
    new Bil("EL 24683", "Nissan Leaf"));
```

- c) Bruk streams til å lage en ny liste av alle elbilene (de som har **skilt** som starter med 'E'). Legg svaret inn i en variabel.
- d) Bruk streams til å skrive ut på skjermen (uten duplikater) alle **modell**navn på elbilene, én linje per bilmodell. Bruk gjerne svaret fra forrige spørsmål i løsningen.

I oppgave e), f) og g) skal du jobbe med beregning av strømregning for en liste med kunder. For enkelhets skyld kan vi si at data om hver kunde kun er **navn** og **forbruk** (i kWh), men vi får tenke oss at det også kunne ha vært andre data her. En liste kan f.eks. se slik ut:

```
List<Kundedata> kundeliste = List.of(
    new Kundedata("Arne", 1234),
    new Kundedata("Per", 2234),
    new Kundedata("Pål", 1000),
    new Kundedata("Emma", 4000),
    new Kundedata("Ine", 5234),
    new Kundedata("Tone", 1111));
```

- e) Bruk streams til å beregne totalforbruket (i kWh) for alle kundene i kundelisten og legg svaret inn i en variabel.

Strømselskapet ønsker å se på ulike modeller for prising, og hvordan dette slår ut på selskapets inntekter.

- f) Lag en metode `beregnTotalInntekt(...)` som tar inn en kundeliste og en funksjon for prisberegning for en enkelt kunde, og som regner ut total inntekt med dette som input.
- g) Vis hvordan `beregnTotalInntekt(...)` kan brukes med et eksempel der prisberegningen er en flat sats på 1.50 kr per kWh. (Altså slik at f.eks. Pål får en regning på 1500,-)

## Oppgave 2 (20% ~ 48 minutter) – JavaScript

- a) JavaScript i nettleser har bla. objektene **window**, **navigator**, **history** og **screen**. For hvert av disse objektene, hva kan vi bruke objektet til?

Du trenger ikke huske navn på egenskaper.

- b) JavaScript og datatyper:

- i. JavaScript betegnes som et type-svakt språk (*weakly typed, loosely typed*).

Hva betyr dette i praksis i JavaScript? Når bestemmes en variable sin type i JavaScript?

- ii. JavaScript sine 7 primitive datatyper inkluderer **string**, **number**, **boolean** og **symbol**.

Når brukes datatypen **symbol**? Demonstrer bruken av **symbol** med et kodeeksempel.

- c) En JavaScript klasse **Studentarkiv** lar foreleser holde oversikt over sine studenter. En student er registrert med *id*, *etternavn*, *fornavn* og eventuelt *tlf* som er en liste med 0 eller flere telefonnumre.

Klassen **Studentarkiv** har følgende offentlige (public) metoder:

- *nystudent(student)*: Metoden legger inn en ny student i arkivet.

Parametre:

- Inn-parameter: **object**
- Returverdi: **boolean** eller verdien *null*

Inn-parameter *student* er et objekt med egenskaper *id*, *etternavn*, *fornavn* og eventuelt også *tlf*:

- *id* er et heltall av type **number** som er unikt for hver student,
- *etternavn* og *fornavn* er av type **string**,
- *tlf* er en forekomst av **Array** med telefonnumre, der hvert telefonnummer er en **string**.

Dersom formatet på inn-parameter *student* er feil eller en student med gitt id allerede finnes blir ikke arkivet oppdatert.

- Returverdi er *null* hvis formatet på inn-parameter *student* er feil.
- Hvis student med gitt id allerede finnes i arkivet er returverdien *false*.
- Hvis arkivet blir oppdatert med en ny student returneres verdien *true*.

- *hartelefon(id, telefonnummer)*: Metoden sjekker om student er registrert med et gitt telefonnummer.

Parametre:

- Inn-parametre: **number**, **string**
- Returverdi: **boolean** eller verdien *null*

Metoden returnerer verdien *null* hvis det ikke finnes noen student med den angitte id. Returverdi er *true* hvis studenten er registrert med det gitte telefonnummeret, ellers *false*.

- *nytelefon(id, nummer)*: Metoden legger til et nytt telefonnummer for student.

Parametre:

- Inn-parametre: **number, string**
- Returverdi: **boolean** eller verdien *null*

Metoden returnerer verdien *null* hvis det ikke finnes noen student med den angitte id. Returverdi er *false* hvis studenten allerede er registrert med det gitte telefonnummeret. Returverdi er *true* hvis det nye telefonnummeret ble lagt til for studenten.

- *eksporterdata()*: Metoden returnerer en tekst med alt innhold i arkivet. Semikolon skiller feltene, og linjeslutt skiller studentene.

Kodeeksempelet nedenfor demonstrerer bruk av **Studentarkiv**:

```
const arkiv = new Studentarkiv();
const ole = arkiv.nystudent({
  id: 101,
  etternavn: "Olsen",
  fornavn: "Ole",
  tlf: ["112 23 344", "323 22 323"]
});
const anne = arkiv.nystudent(
  { id:106,etternavn:"Annesen",fornavn: "Anne" }
);
const oletelefon = arkiv.hartelefon(101, "112 23 344");
const annetelefon = arkiv.hartelefon(106, "767 44 333");
const arkivdata = arkiv.eksporterdata();
```

Etter at koden over har kjørt har konstantene følgende verdier:

- Konstantene *ole*, *anne* og *oletelefon* er *true*.
- Konstanten *annetelefon* er *false*,
- Konstanten *arkivdata* inneholder følgende tekst:

```
101;Olsen;Ole;112 23 344;323 22 323
106;Annesen;Anne
```

**Oppgave:** Skriv JavaScript-koden for **Studentarkiv** i samsvar med teksten over.

**Hjelp (se neste side)**

**Hjelp:**

- Både **Map**, **Set** og **Array** har en metode *forEach*.
- **Map** og **Set** sin metode *values* returnerer en iterator over alle verdier.
- **Map** sin metode *keys* returnerer en iterator over alle nøkler.
- **Array** sin statiske metode *from*, og også spre (*spread*) operator kan kopiere et itererbart objekt til en **Array**.
- **Map** har bla. metoder *has*, *set*, *get* og *delete*.
- **Set** har bla. metoder *has*, *add* og *delete*.
- Operator *typeof* returnerer en **string** som angir operanden sin type.
- Operator *instanceof* tester om operand er en forekomst av en klasse, også via arv.

### Oppgave 3 (10% ~ 24 minutter) – Tråder

Vi ønsker å lage et program som beregner store Fibonacci-tall, f.eks. `fib(50)`. Du trenger ikke å tenke på hvordan utregningen gjøres. Poenget er at utregningen kan ta litt tid ( $O(2^n)$ ) hvis vi bruker en enkel/naiv algoritme.

For å ha en følelse av at programmet gjør noe mens beregningen foregår skal det skrives ut en prikk hvert sekund så lenge utregningen foregår. Når beregningen er ferdig skal svaret skrives ut på skjermen og programmet avslutte.

Eksempel på beregning og utskrift av `fib(48)`, som tok ca. 30 sekunder:

```
Beregner fib(48): .....  
Svar: 4807526976
```

Vi kan anta at vi har en metode **`long fib(int n)`** som vi kan gjøre et metodekall til for å få utført selve beregningen. (Det er denne metoden som bruker lang tid.)

Du skal skrive `main()`, som skal gjøre følgende:

- Tallet vi skal bruke i kjøringen (f.eks. 50) kan være en hardkodet konstant.
- Det skal opprettes og startes en tråd som gjør beregningen og skriver ut svaret.
- Det skal skrives ut en prikk hvert sekund så lenge den andre tråden **`isAlive()`** (som er en metode i `Thread`-klassen).

For å få full score må løsningen ikke være unødvendig komplisert.

### Oppgave 4 (10% ~ 24 minutter) – Passord

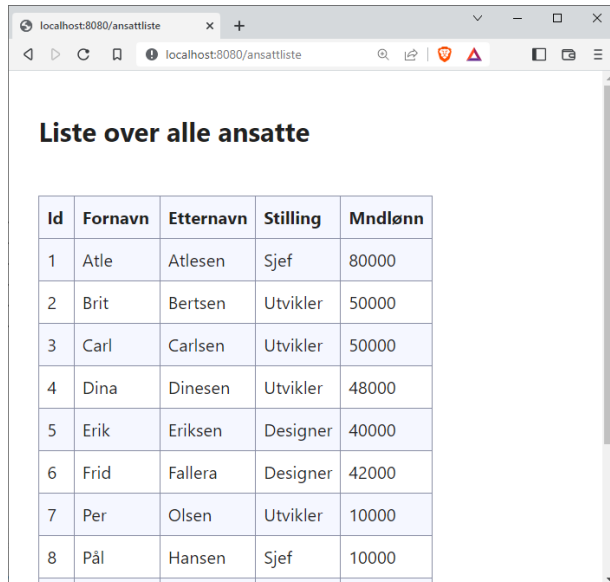
Korrekt håndtering og lagring av brukeres passord er helt nødvendig av sikkerhets- og personvern hensyn.

Beskriv i tekniske termer hvordan brukeres valgte passord bør prosesseres og lagres (i database) ved f.eks. oppretting av ny brukerkonto på en tjeneste. Fortell også hvordan de ulike skrittene i prosessen er med på å beskytte brukerens passord mot cracking.

## Oppgave 5 (40% ~ 96 minutter) – Web backend med Spring MVC

Du skal lage deler av en applikasjon for å holde orden på ansatte i en bedrift. I hovedsak er det to ulike brukstilfeller vi skal se på:

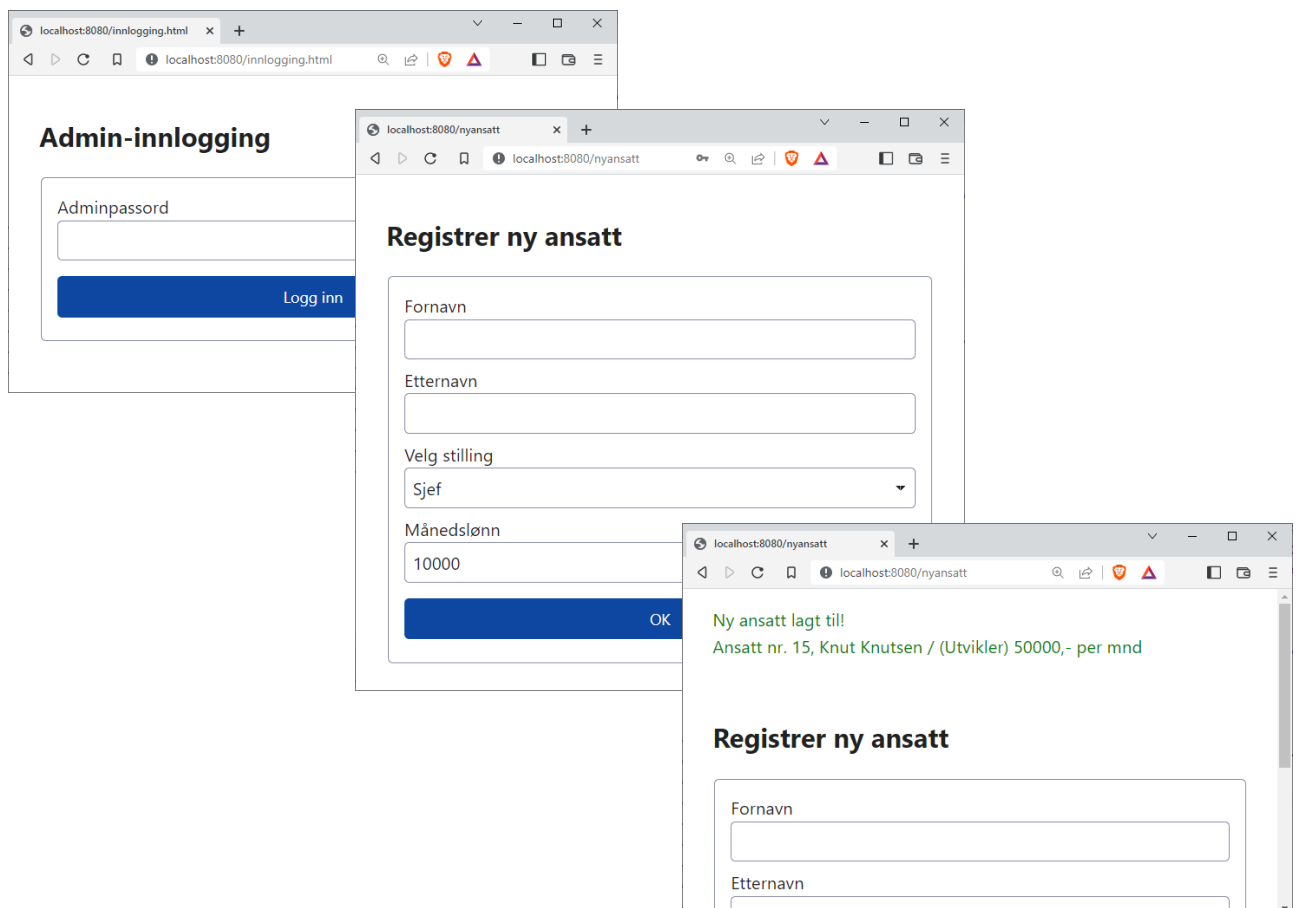
### 1. Få se en oversikt over alle ansatte



The screenshot shows a web browser window with the address bar at localhost:8080/ansattliste. The page title is "Liste over alle ansatte". Below the title is a table with 5 columns: Id, Fornavn, Etternavn, Stilling, and Mndlønn. The table contains 8 rows of employee data.

Id	Fornavn	Etternavn	Stilling	Mndlønn
1	Atle	Atlesen	Sjef	80000
2	Brit	Bertsen	Utvikler	50000
3	Carl	Carlsen	Utvikler	50000
4	Dina	Dinesen	Utvikler	48000
5	Erik	Eriksen	Designer	40000
6	Frid	Fallera	Designer	42000
7	Per	Olsen	Utvikler	10000
8	Pål	Hansen	Sjef	10000

### 2. Registrere en ny ansatt. Denne funksjonen krever at man er 'innlogget som admin'.



The image shows three overlapping browser windows. The leftmost window is titled "Admin-innlogging" and has a form with a label "Adminpassord" and a "Logg inn" button. The middle window is titled "Registrer ny ansatt" and has a form with fields for "Fornavn", "Etternavn", "Velg stilling" (a dropdown menu with "Sjef" selected), and "Månedslønn" (with "10000" entered), and an "OK" button. The rightmost window shows a success message: "Ny ansatt lagt til! Ansatt nr. 15, Knut Knutsen / (Utvikler) 50000,- per mnd", followed by the "Registrer ny ansatt" form.



Litt mer detaljer om hvordan applikasjonen skal virke:

- Ansatte er lagret i databasen i en tabell kalt **ansatt** med attributter tilsv. det første skjermbildet.
- **Ansatt.id** genereres automatisk av databasen.
- Adminpassordet er oppgitt som **app.adminPassord** i **application.properties**
- Det er ingen direkte utlogging. I stedet blir admin logget ut automatisk etter et antall sekunder med inaktivitet, oppgitt i **app.adminTimeout** i **application.properties**
- Feil admin-passord gir innlogginsskjemaet på nytt uten noe mer info.
- Forsøk på å registrere en ansatt uten å være innlogget som admin gir innlogginsskjemaet på nytt uten noe mer info.
- Drop-down-listen der man velger stilling skal være generert dynamisk ut fra hvilke stillinger som finnes i databasen.
- Ugyldige data ved registrering av ny ansatt skal gi registreringsskjemaet tilbake med en (rød) melding (øverst) om at data ikke er gyldige. Dette er ikke vist på skjermbildene over.
- Vellykket registrering skal gi registreringsskjemaet tilbake med en (grønn) melding (øverst) om at ny ansatt er registrert, se skjermbilde.

Forslag til URL-er, Controllere og Views:

- **/ansattliste, /nyansatt og /innlogging**
- Én controller for henting av ansattlisten
- Én controller for henting av registreringsskjema og for registrering / lagring av nyansatt
- Én controller for admininnlogging
- **ansattliste.jsp, nyansattskjema.jsp og innlogging.html** (den siste kan være statisk)

Hjelpeklasser som kan brukes i controllerne:

- **@Service AnsattDbService** (som er knyttet til databasen) med metodene
  - List<Ansatt> finnAlleAnsatte()
  - List<String> finnStillinger()
  - Ansatt registrerNyAnsatt(String fornavn, String etternavn, String stilling, Integer lønn)
- **@Service LogginnService** med metodene
  - void loggInn(HttpServletRequest request)
  - erLoggetInn(HttpServletRequest request)

For å få full score må du løse oppgaven på best mulig måte i hht. det som er gjennomgått i kurset, dvs. **Spring MVC / Spring Boot, Model-View-Controller, Post-Redirect-Get, EL og JSTL** i JSP-ene, **robusthet, ufarliggjøring** av brukerinnt, **god bruk av hjelpeklasser, elegant kode**, osv ...

Oppgaver:

- a) Skriv controlleren for henting av ansattlisten
- b) Skriv ansattliste.jsp
- c) Skriv controlleren med GET-mapping-metoden som har ansvar for henting av registreringsskjema. (POST-mappingen i denne controlleren kommer i et senere delspørsmål.)
- d) Bruk skjelettet nedenfor for nyansattskjema.jsp og fullfør denne. Dvs. fyll inn det som mangler, merket med **?1? ... ?7?**.

```
<html>
<body>
  <c:if test="${not empty nyansatt}"> //Satt ved registrering
    <p style="color:green;"> ?1? </p>
  </c:if>
  <c:if test="${not empty feilmelding}"> //Satt ved feil i registrering
    <p style="color:red;"> ?2? </p>
  </c:if>

  <h3>Registrer ny ansatt</h3>
  <form method="?3?">
    Fornavn <input type="?4?" name="fornavn"><br>
    Etternavn <input type="?5?" name="etternavn"><br>
    Velg stilling <select name="stilling">
      ?6? //Tips: Syntaks for et select-alternativ er
      //<option value="..."></option>
    </select><br>
    Månedslønn <input type="?7?" name="mndlonn"
      min="10000" max="99999" value="10000" ><br>
    <input type="submit" value="OK">
  </form>
</body>
</html>
```

- e) Fullfør controlleren fra c) med POST-mapping-metoden som har ansvar for registrering / lagring av nyansatt. Hver request-parameter mottas separat, og må valideres. Hvis ikke alle data er gyldige er det greit nok med en generell feilmelding, se skjelett for oppgave d).

For å få full score må du også skrive en hjelpeklasse AnsattValidator som har ansvar for input-valideringen. (Hvis du ikke rekker dette kan du bruke den som om den finnes.)

Valideringsregler:

- fornavn og etternavn inneholder minst 2 tegn hver
- månedslønn må være mellom 0 og 100 000
- valgt stilling må finnes i databasen
- Hvis ikke alle data er gyldige er det greit nok med en generell feilmelding, se d)

# EKSAMENSOPPGÅVE

**Emnekode: DAT108**

**Emnenamn: Programmering og webapplikasjoner**

**Klasse: 2. klasse DATA / INF**

**Dato: 16. desember 2022**

---

Eksamensform: Skriftleg skuleeksamen (Wiseflow | FLOWlock)

Eksamenstid: 4 timer (0900-1300)

Tal på eksamensoppgåver: 5

Tal på sider (medrekna denne): 10

Tal på vedlegg: Ingen

Tillatte hjelpemiddel: Ingen

Lærarar:   Lars-Petter Helland (928 28 046) lph@hvl.no  
              Bjarte Kileng (909 97 348) bki@hvl.no

**LYKKE TIL!**

## Oppg ve 1 (20% ~ 48 minutt) – Lambda-uttrykk og straumer

For      full score m  l ysingane ikkje vera un dig kompliserte, og det b r brukast metodereferansar der det er mogleg.

- a) Nedanfor ser du 5 ulike  $\lambda$ -uttrykk. Skriv ei setning for kvar av desse der dei blir tilordna ein variabel. Det me er ute etter er datatypen til variablane, t.d. om det er ein `Consumer<String>`, ein `Function<Integer, String>`, etc. ..

```
a -> System.out.println(a)    // der a er en String
(a,b) -> a.compareTo(b)       // der a og b er String-er
a -> a * a                     // der a er et heltall
a -> a >                       // der a er et heltall
(a,b) -> a + b                 // der a og b er heltall
```

- b) Anta at du har metoden **utplukk**, metoden **main** og resultatet **[3, 6, 9]** av   k yre main:

```
List<Integer> utplukk(List<Integer> liste, ??? b) {
    return liste.stream().filter(b).toList();
}

void main(...) {
    List<Integer> liste = List.of(1,2,3,4,5,6,7,8,9);
    List<Integer> resultat = utplukk(liste, ???);
    System.out.println(resultat);
}

[3, 6, 9]
```

Kva skal st  der det st r **???** i parameterlista (dvs. typen til den formelle parameteren `b`), og kva skal st  der det st r **???** i metodekallet (dvs. den aktuelle parameteren)?

I oppg ve c) og d) skal du jobbe med ei liste av biler, t.d.:

```
List<Bil> biler = List.of(
    new Bil("EK 12345", "Tesla model Y"),
    new Bil("EV 52345", "Tesla model Y"),
    ...
    new Bil("SV 12346", "Mazda 5"),
    new Bil("SU 2468 ", "Volvo 24 "),
    new Bil("EL 24683", "Nissan Leaf"));
```

- c) Bruk streams til   lage ei ny liste av alle elbilane (dei som har **skilt** som startar med 'E'). Legg svaret inn i ein variabel.
- d) Bruk streams til   skrive ut p  skjermen (utan duplikat) alle **modellnamn** p  elbilane,  i linje per bilmodell. Bruk gjerne svaret fr  f rre sp rsm l i l ysinga.

I oppgåve e), f) og g) skal du jobbe med berekning av straumrekning for ei liste med kundar. For enkelheits skuld kan vi seie at data om kvar kunde berre er **namn** og **forbruk** (i kWh), men vi får tenkje oss at det også kunne ha vore andre data her. Ei liste kan t.d. sjå slik ut:

```
List<Kundedata> kundeliste = List.of(
    new Kundedata("Arne", 1234),
    new Kundedata("Per", 2234),
    new Kundedata("Pål", 1000),
    new Kundedata("Emma", 4000),
    new Kundedata("Ine", 5234),
    new Kundedata("Tone", 1111));
```

- e) Bruk streams til å berekne totalforbruket (i kWh) for alle kundane i kundelista og legg svaret inn i ein variabel.

Strømselskapet ønskjer å sjå på ulike modellar for prising, og korleis dette slår ut på inntektene til selskapet.

- f) Lag ein metode `beregnTotalInntekt(...)` som tek inn ei kundeliste og ein funksjon for prisberekning for ein enkelt kunde, og som reknar ut total inntekt med dette som input.
- g) Vis korleis `beregnTotalInntekt(...)` kan brukast med eit døme der prisberekninga er ein flat sats på 1.50 kr per kWh. (Altså slik at t.d. Pål får ei rekning på 1500,-)

## Oppgåve 2 (20% ~ 48 minutt) – JavaScript

- a) JavaScript i nettleser har bla. objekta **window**, **navigator**, **history** og **screen**. For kvart av desse objekta, til kva kan vi nytte objektet?

Du treng ikkje hugse namn på eigenskap.

- b) JavaScript og datatypar:

- i. JavaScript verte omtala som eit type-svakt språk (*weakly typed, loosely typed*).

Kva tyder dette i praksis i JavaScript? Når verte ein variable sin type fastsett i JavaScript?

- ii. JavaScript sine 7 primitive datatypar inkluderer **string**, **number**, **boolean** og **symbol**.

Når verte datatypen **symbol** nytta? Demonstrer bruken av **symbol** med eit døme med kode.

- c) Ei JavaScript klasse **Studentarkiv** lar forelesar halde oversikt over sine studentar. Ein student er registrert med *id*, *etternavn*, *fornavn* og eventuelt *tlf* som er ei liste med 0 eller fleire telefonnumre.

Klassa **Studentarkiv** har følgjande offentlege (public) metodar:

- *nystudent(student)*: Metoden legg inn ein ny student i arkivet.

Parameterar:

- Inn-parameter: **object**
- Returverde: **boolean** eller verde *null*

Inn-parameter *student* er eit objekt med eigenskapar *id*, *etternavn*, *fornavn* og eventuelt også *tlf*:

- *id* er eit heiltal av type **number** som er unikt for kvar student,
- *etternavn* og *fornavn* er av type **string**,
- *tlf* er ein førekomst av **Array** med telefonnumre, der kvart telefonnummer er ein **string**.

Dersom formatet på inn-parameter *student* er feil eller ein student med gjeve id allereie finnes blir ikkje arkivet oppdatert.

- Returverde er *null* om formatet på inn-parameter *student* er feil.
- Om student med gjeve id allereie finnes i arkivet er returverde *false*.
- Om arkivet blir oppdatert med ein ny student verte verde *true* returnert.

- *harteleson(id, telefonnummer)*: Metoden sjekkar om student er registrert med eit gjeve telefonnummer.

Parameterar:

- Inn-parameterar: **number**, **string**
- Returverde: **boolean** eller verde *null*

Metoden returnerer *verde null* om det ikkje finnes nokon student med den gjeve id. Returverde er *true* om studenten er registrert med det gjeve telefonnummeret, ellers *false*.

- *nytelefon(id, nummer)*: Metoden legg til eit nytt telefonnummer for student.

Parameterar:

- Inn-parameterar: **number, string**
- Returverde: **boolean** eller *verde null*

Metoden returnerer *verde null* om det ikkje finnes nokon student med den gjeve id. Returverde er *false* om studenten allereie er registrert med det gjeve telefonnummeret. Returverde er *true* om det nye telefonnummeret blei lagt til for studenten.

- *eksporterdata()*: Metoden returnerer ein tekst med alt innhald i arkivet. Semikolon skjelar felte, og linjeslutt skjelar studentane.

Døme med kode nedanfor demonstrerer bruk av **Studentarkiv**:

```
const arkiv = new Studentarkiv();
const ole = arkiv.nystudent({
  id: 101,
  etternavn: "Olsen",
  fornavn: "Ole",
  tlf: ["112 23 344", "323 22 323"]
});
const anne = arkiv.nystudent(
  { id:106,etternavn:"Annesen",fornavn: "Anne" }
);
const oletelefon = arkiv.hartelefon(101, "112 23 344");
const annetelefon = arkiv.hartelefon(106, "767 44 333");
const arkivdata = arkiv.eksporterdata();
```

Etter at koden over har kjørd har konstantane følgjande verde:

- Konstantane *ole, anne og oletelefon* er *true*.
- Konstanten *annetelefon* er *false*,
- Konstanten *arkivdata* inneheld følgjande tekst:

```
101;Olsen;Ole;112 23 344;323 22 323
106;Annesen;Anne
```

**Oppgåve:** Skriv JavaScriptkoden for **Studentarkiv** i høve med teksten over.

**Hjelp (sjå neste side).**

**Hjelp:**

- Både **Map**, **Set** og **Array** har ein metode *forEach*.
- **Map** og **Set** sin metode *values* returnerer ein iterator over alle verdi.
- **Map** sin metode *keys* returnerer ein iterator over alle nøklar.
- **Array** sin statiske metode *from*, og også spreie (*spread*) operator kan kopiere eit itererbart objekt til ein **Array**.
- **Map** har bla. metodar *has*, *set*, *get* og *delete*.
- **Set** har bla. metodar *has*, *add* og *delete*.
- Operator *typeof* returnerer ein **string** som angir operanden sin type.
- Operator *instanceof* testar om operand er ein førekomst av ei klasse, også via arv.



### Oppgave 3 (10% ~ 24 minutt) – Trådar

Me ønskjer å laga eit program som bereknar store Fibonacci-tal, t.d. `fib(50)`. Du treng ikkje å tenkja på korleis utrekninga blir gjort. Poenget er at utrekninga kan ta litt tid ( $O(2^n)$ ) viss me bruker ein enkel/naiv algoritme.

For å ha ei kjensle av at programmet gjer noko medan berekninga går føre seg skal det skrivast ut ein prikk kvart sekund så lenge utrekninga går føre seg. Når berekninga er ferdig skal svaret skrivast ut på skjermen og programmet avslutte.

Døme på berekning og utskrift av `fib(48)`, som tok ca. 30 sekund:

```
Beregner fib(48): .....  
Svar: 4807526976
```

Me kan rekna med at me har ein metode **`long fib(int n)`** som me kan gjera eit metodekall til for å få utført sjølve berekninga. (Det er denne metoden som bruker lang tid.)

Du skal skrive `main()`, som skal gjere følgjande:

- Talet me skal bruka i køyringa (t.d. 50) kan vera ein hardkoda konstant.
- Det skal opprettast og startast ein tråd som gjer berekninga og skriv ut svaret.
- Det skal skrivast ut ein prikk kvart sekund så lenge den andre tråden **`isAlive()`** (som er ein metode i Thread-klassen).

For å få full score må løysninga ikkje vere unødvendig komplisert.

### Oppgave 4 (10% ~ 24 minutt) – Passord

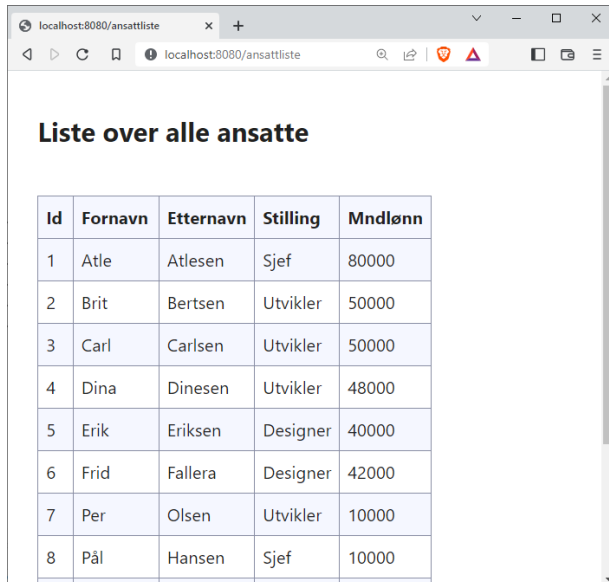
Korrekt handtering og lagring av brukarars passord er heilt nødvendig av tryggleiks- og personvernomsyn.

Beskriv i tekniske termar korleis brukarars valde passord bør prosesserast og lagrast (i database) ved t.d. oppretting av ny brukarkonto på ei teneste. Fortel også korleis dei ulike stega i prosessen er med på å verna passordet til brukaren mot cracking.

## Oppgave 5 (40% ~ 96 minutt) – Web backend med Spring MVC

Du skal laga delar av ein applikasjon for å halda orden på tilsette i ei bedrift. I hovudsak er det to ulike brukstilfelle me skal sjå på:

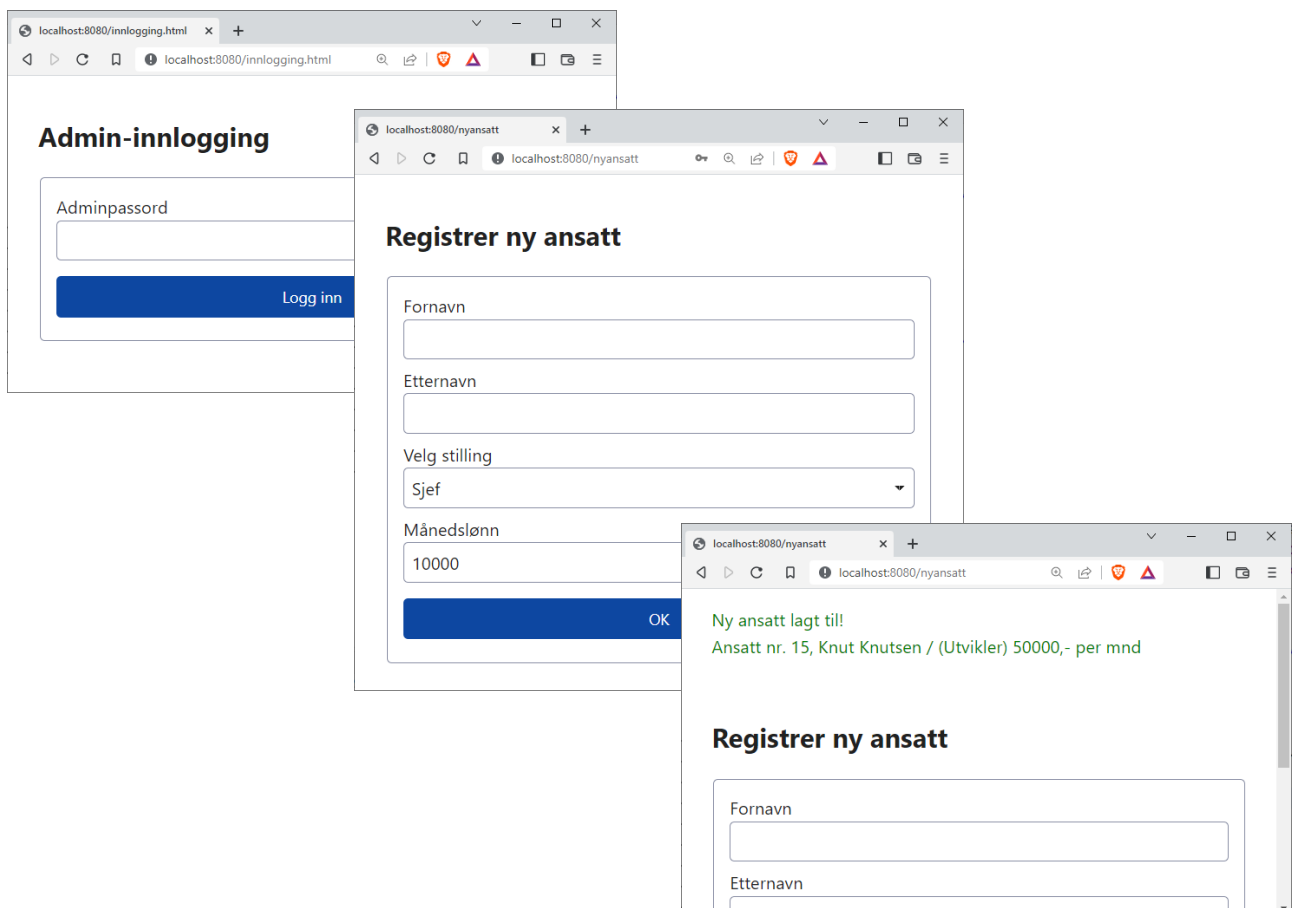
### 1. Få sjå ei oversikt over alle tilsette



The screenshot shows a web browser window with the address bar at localhost:8080/ansattliste. The page title is "Liste over alle ansatte". Below the title is a table with 5 columns: Id, Fornavn, Etternavn, Stilling, and Mndlønn. The table contains 8 rows of employee data.

Id	Fornavn	Etternavn	Stilling	Mndlønn
1	Atle	Atlesen	Sjef	80000
2	Brit	Bertsen	Utvikler	50000
3	Carl	Carlsen	Utvikler	50000
4	Dina	Dinesen	Utvikler	48000
5	Erik	Eriksen	Designer	40000
6	Frid	Fallera	Designer	42000
7	Per	Olsen	Utvikler	10000
8	Pål	Hansen	Sjef	10000

### 2. Registrere ein ny tilsett. Denne funksjonen krev at ein er 'innlogga som admin'.



The image shows three overlapping browser windows. The leftmost window is titled "Admin-innlogging" and has a form with a label "Adminpassord", an input field, and a blue "Logg inn" button. The middle window is titled "Registrer ny ansatt" and has a form with labels "Fornavn", "Etternavn", "Velg stilling", and "Månedslønn", corresponding input fields, a dropdown menu with "Sjef" selected, and a blue "OK" button. The rightmost window shows a success message "Ny ansatt lagt til!" and "Ansatt nr. 15, Knut Knutsen / (Utvikler) 50000,- per mnd" above another "Registrer ny ansatt" form.

Litt meir detaljar om korleis applikasjonen skal verke:

- Tilsette er lagra i databasen i ein tabell kalla **ansatt** med attributt tilsv. det første skjermbiletet.
- **Ansatt.id** blir generert automatisk av databasen.
- Adminpassordet er oppgitt som **app.adminPassord** i **application.properties**
- Det er ingen direkte utlogging. I staden blir admin logga ut automatisk etter eit tal sekunder med inaktivitet, oppgitt i **app.adminTimeout** i **application.properties**
- Feil admin-passord gir innlogginsskjemaet på nytt utan meir info.
- Forsøk på å registrere ein tilsett utan å vere innlogga som admin gir innlogginsskjemaet på nytt utan meir info.
- Drop-down-listen der ein vel stilling skal vere generert dynamisk ut frå kva stillingar som finst i databasen.
- Ugyldige data ved registrering av ny tilsett skal gi registreringsskjemaet tilbake med ein (raud) melding (øvt) om at data ikkje er gyldige. Dette er ikkje vist på skjermbileta over.
- Vellykka registrering skal gi registreringsskjemaet tilbake med ei (grøn) melding (øvt) om at ny tilsett er registrert, sjå skjermbilete.

Forslag til URL-er, Controllere og Views:

- **/ansattliste, /nyansatt og /innlogging**
- Éin controller for henting av tilsettlista
- Éin controller for henting av registreringsskjema og for registrering / lagring av nyttilsett
- Éin controller for admininnlogging
- **ansattliste.jsp, nyansattskjema.jsp og innlogging.html** (den siste kan vere statisk)

Hjelpeklasser som kan brukast i controllerane:

- **@Service AnsattDbService** (som er knytt til databasen) med metodane
  - List<Ansatt> finnAlleAnsatte()
  - List<String> finnStillingar()
  - Ansatt registrerNyAnsatt(String fornavn, String etternavn, String stilling, Integer lønn)
- **@Service LogginnService** med metodane
  - void loggInn(HttpServletRequest request)
  - erLoggetInn(HttpServletRequest request)

For å få full score må du løyse oppgåva på best mogleg måte i hht. det som er gjennomgått i kurset, dvs. **Spring MVC / Spring Boot, Model-View-Controller, Post-Redirect-Get, EL og JSTL** i JSP-ene, **robusthet, ufarlegging** av brukarinput, **god bruk av hjelpeklassar, elegant kode**, osv ...

Oppgåver:

- Skriv controlleren for henting av tilsettlista
- Skriv ansattliste.jsp
- Skriv controlleren med GET-mapping-metoden som har ansvar for henting av registreringsskjema. (POST-mappingen i denne controlleren kjem i eit seinare delspørsmål.)
- Bruk skjelettet nedanfor for nyansattskjema.jsp og fullfør denne. Dvs. fyll inn det som manglar, merka med ?1? ... ?7?.

```
<html>
<body>
  <c:if test="${not empty nyansatt}"> //Satt ved registrering
    <p style="color:green;"> ?1? </p>
  </c:if>
  <c:if test="${not empty feilmelding}"> //Satt ved feil i registrering
    <p style="color:red;"> ?2? </p>
  </c:if>

  <h3>Registrer ny ansatt</h3>
  <form method="?3?">
    Fornavn <input type="?4?" name="fornavn"><br>
    Etternavn <input type="?5?" name="etternavn"><br>
    Velg stilling <select name="stilling">
      ?6? //Tips: Syntaks for et select-alternativ er
      //<option value="..."></option>
    </select><br>
    Månedslønn <input type="?7?" name="mndlonn"
      min="10000" max="99999" value="10000" ><br>
    <input type="submit" value="OK">
  </form>
</body>
</html>
```

- Fullfør controlleren fra c) med POST-mapping-metoden som har ansvar for registrering / lagring av nytilsett. Kvar request-parameter blir motteken separat, og må validerast. Viss ikkje alle data er gyldige er det greitt nok med en generell feilmelding, sjå skjelett for oppgåve d).

For å få full score må du også skrive en hjelpeklasse AnsattValidator som har ansvar for input-valideringa. (Viss du ikkje rekk dette kan du bruke den som om ho finst.)

Valideringsreglar:

- fornamn og etternamn inneheld minst 2 teikn kvar
- månadslønn må vere mellom 0 og 100 000
- vald stilling må finnast i databasen
- Viss ikkje alle data er gyldige er det greitt nok med ei generell feilmelding, sjå d)