

## Seksjon 1

### Om eksamen

DAT108 Programmering og Webapplikasjoner, vår 2024

Tid: 4 timer

Du kan svare på engelsk eller norsk. Skriver du på norsk kan du bruke engelske faguttrykk.

Jeg har lest og forstått instruksjonene

☐ Ja

☐ Nei

# Oppgave 1 (10%) - Strømmer/Streams

**Kort om retting:**

- Teller ca. 10% (24 minutter).
- Fem deloppgaver: A, B, C, D og E.
- Hver deloppgave teller ca. 2% (4-5 minutter)
- Oppgaven rettes automatisk.
- Det er bare en **helt** rett løsning på hvert problem.
- Du blir gitt **delvis** uttelling for delvis korrekt svar:
  - Alle felt som er korrekte gir poeng.
  - Feil svar gir **IKKE** trekk.

**Oppgave A:** Velg rett svar om metoder som brukes i strømmer. Hvilken metode...

	filter	map	forEach	sorted	limit	stream	s
fjerner elementer som ikke oppfyller et predikat?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	(
transformerer strøm-elementer med en funksjon?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	(
utfører en handling på elementene og avslutter strømmen?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	(
endrer rekkefølgen til elementene i en strøm?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	(

**Oppgave B:** Du skal bruke nedtrekksmenyene til å lage følgende strøm:

Gitt en liste av desimaltall (`List<Double>` desimaltall), lag en ny liste (`List<Integer>` heltall) hvor tallene er rundet av til nærmeste heltall.

```

    ▼ heltall = desimaltall. ▼ ( )
        . ▼ ( ▼ )
        . ▼ ( );

```

**Oppgave C:** Du skal bruke nedtrekksmenyene til å lage følgende strøm:

Gitt en liste med tall (`List<Integer>` tall), lag en ny liste som inneholder primtallene i listen. Primtallene skal være sortert fra minst til størst og uttrykt som ord.

Du skal bruke de to funksjonelle kontraktene under i strømmen (anta at de alt er implementert).

```

Predicate<Integer> erPrimtall; // Et predikat som avgjør om et heltall er et p
rimtall.

```

```

Function<Integer, String> tallTilOrd; // En funksjon som sender tall (1, 2, 3, ...) t
il sitt tilhørende tallord ("en", "to", "tre", ...).

```

```

    ▼ minste = tall. ▼ ( )
        . ▼ ( ▼ )
        . ▼ ( )
        . ▼ ( ▼ )
        . ▼ ( );

```

**Oppgave D:** Hvilket tall blir skrevet ut når koden under kjøres?

```
import java.util.Arrays;
import java.util.List;

public class StreamExample {
    public static void main(String[] args) {
        List<Integer> tall = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15);

        int resultat = tall.stream()
            .filter(n -> n % 2 == 0)
            .mapToInt(n -> n / 2)
            .sorted()
            .limit(3)
            .sum();

        System.out.println(resultat);
    }
}
```

Tallet som blir skrevet ut er

**Oppgave E:** Her skal du sette sammen strømmene med den beskrivelsen som passer best.

```
String ut= in.stream().reduce  
("", (a, b) -> a + b);
```

```
Integer ut = in.stream().redu  
ce(0, (a, b) -> a + 1 );
```

```
String ut = in.stream().reduc  
e("", (a, b) -> b + a);
```

```
Integer ut= in.stream().reduc  
e(0, (a, b) -> b + a);
```

⚙ Summerer (legger sammen) alle tallene i en liste.

⚙ Konkaterener (setter sammen) alle strengene i en liste i rett rekkefølge.

⚙ Teller hvor mange tall det er i en liste.

⚙ Konkaterener (setter sammen) alle strengene i en liste i reversert rekkefølge.

**Antagelser/bemerkninger:**

Dette feltet kan stå tomt. Dersom noe er uklart i oppgavene over så kan du skrive ned eventuelle antagelsene du har gjort her. Marker kommentarene dine med oppgavenummer. Skriv for eksempel:

// Oppgave B iii).

\*Dine antagelser...\*

**B** *I* u | ≡ ≡<sub>2</sub> á

0 / 10000 Word Limit

# Oppgave 2 (10%) - Funksjonelle kontrakter

**Kort om retting:**

- Teller ca. 10% (24 minutter).
- Fem deloppgaver: A, B, C, D og E
- Hver deloppgave teller ca. 2% (4-5 minutter)
- Oppgaven rettes automatisk.
- Det er bare en **helt** rett løsning på hvert problem.
- Du blir gitt **delvis** uttelling for delvis korrekt svar:
  - Alle felt som er korrekte gir poeng.
  - Feil svar gir **IKKE** trekk.

**Oppgave A:** Hvilken funksjonell kontrakt brukes for å...

	BiFunction	Predicate	UnaryOperator	Consumer	Supplier	Bi
A ta inn en verdi og returnere en verdi av samme type?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
B evaluere om en betingelse er oppfylt?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
C produsere en verdi uten et argument?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
D utføre en handling uten å returnere noe?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	

**Oppgave B:** Gi rett type til variablene b1, b2 og b3 under:

// i)

b1 = x -> x < 0;

// ii)

b2 = x -> x\*x ;

// iii)

b3 = x -> Double.toString(x).toUpperCase().length();

**Oppgave C:** Velg det lambdauttrykket passer med typen til variabelene c1, c2 og c3.

// i)

Predicate<Integer> c1 =  ;

// ii)

Function<Fjelltur, String> c2 =  ;

// iii)

Comparator<Fjelltur> c3 =  ;**Informasjon om Fjelltur:**

FELTVARIABLER: Klassen Fjelltur har feltvariablene gitt under.

- String navn
- Integer høyde
- Integer lengde
- String gradering // Vanskelighetsgradene (fra lettest til vanskeligst) er "GRØNN", "BLÅ", "RØD" og "SVART".

KONSTRUKTØR: Fjelltur har en konstruktør som tar inn navn, høyde, lengde og gradering.

METODER: Hver feltvariabel har tilhørende set'ere (void) og get'ere (ingen argumenter). Fjelltur har også en compareTo metode som sammenligner fjellturer utifra vanskelighetsgrad:

- tur1.compareTo(tur2) = -1 hvis og bare hvis tur1 er lettere en tur2.
- tur1.compareTo(tur2) = 1 hvis og bare hvis tur2 er lettere en tur1.
- tur1.compareTo(tur2) = 0 hvis og bare hvis tur1 og tur2 har samme gradering.



**Oppgave D:** Velg rett lambdauttrykk og type til variablene d1, d2 og d3 som passer med beskrivelsen av input og output.

```
// i)
// Inn: En fjelltur (Fjelltur).
// Ut: En streng som inneholder navn og vanskelighetsgrad på turen.
```

d1 =  ;

```
// ii)
// Inn: To fjellturer (Fjelltur).
// Ut: Høydeforskjellen (differansen) på de to fjellturene
// Hint: Metoden skal brukes i en strøm til sortering av Fjellturer etter stigning.
```

d2 =  ;

```
// iii)
// Inn: En fjelltur (Fjelltur) og et lambdauttrykk som estimerer hvor lang tid en fjelltur tar.
// Ut: true dersom turen tar mindre enn 5.0 (timer) OG den ikke har vanskelighetsgrad "SVART", ellers false.
```

d3 =  ;

**Oppgave E:** Er kodelinjene under gyldig? (I.e. er syntaks korrekt OG er uttrykket gitt rett funksjonell kontrakt)?

		Gyldig	Ugyldig
A	<div>Predicate&lt;Fjelltur&gt; lambda1 = t -&gt; t.getHoyde() &gt; 200 0;</div>	<input type="radio"/>	<input type="radio"/>
B	<div>Supplier&lt;Fjelltur&gt; lambda2 = () -&gt; new Fjelltur("Ulri ken", 643, 2000, "BLÅ");</div>	<input type="radio"/>	<input type="radio"/>
C	<div>Consumer&lt;Fjelltur&gt; lambda3 = t -&gt; t.setLengde(t.getLe ngde() + 100);</div>	<input type="radio"/>	<input type="radio"/>
D	<div>Function&lt;Fjelltur, Fjelltur&gt; lambda4 = (t1, t2) -&gt; t 1.getNavn().equals(t2.getNavn());</div>	<input type="radio"/>	<input type="radio"/>
E	<div>BiConsumer&lt;Fjelltur, Integer&gt; lambda5 = (t, x) -&gt; t.s etLengde(t.getLengde() + x);</div>	<input type="radio"/>	<input type="radio"/>
F	<div>Consumer&lt;Fjelltur&gt; lambda6 = t -&gt; System.out.println ("Navn: " + t.getNavn());</div>	<input type="radio"/>	<input type="radio"/>

Antagelser/bemerkninger:

Dette feltet kan stå tomt. Dersom noe er uklart i oppgavene over så kan du skrive ned eventuelle antagelsene du har gjort her. Marker kommentarene dine med oppgavenummer. Skriv for eksempel:

// Oppgave B iii).

\*Dine antagelser...\*

**B**

*I*

U

☰

☰<sup>1</sup><sub>2</sub>

á

0 / 10000 Word Limit

## Seksjon 4

# Oppgave 3 (10%) - JavaScript teorispørsmål

I disse flervalgsoppgavene er ett og kun ett av utsagnene korrekt. En oppgave gir kun poeng hvis det korrekte utsagnet er valgt.

## a) (1%)

HTML taggen SCRIPT kjenner et attributt DEFER.

```
<SCRIPT DEFER SRC="js/konkurranseskontroller.js"></SCRIPT>
```

Hva er konsekvensene av attributtet defer?

- ☐ Attributtet var i bruk i tidligere nettlesere, men er nå merket som *deprecated*.
- ☐ Hvis ikke dokumentet i *src* attributtet finnes skal setningen ignoreres.
- ☐ Nettleser skal bruke en kompatibilitetsmodus for gammel JavaScript når koden kjøres.
- ☐ JavaScript-koden kan lastes parallelt med at web dokumentet lastes. Når koden er ferdig lastet kan koden kjøres umiddelbart.
- ☐ JavaScript-koden kan lastes parallelt med at web dokumentet lastes. Koden skal kjøres først når web dokumentet er ferdig parset.
- ☐ JavaScript-koden kan lastes parallelt med at web dokumentet lastes. Koden skal kjøres først når webdokumentet fjernes fra nettleser sin hukommelse.

## b) (1%)

Koden nedenfor finnes i HEAD delen av et HTML dokument:

```
<SCRIPT>
  const h1repr = document.getElementsByTagName("h1")[0];
  console.log(h1repr.tagName);
</SCRIPT>
```

Dokumentet har en BODY tagg med flere tagger H1.

Når koden kjøres produseres denne feilmeldingen:

*Uncaught TypeError: h1repr is undefined*

Hva er årsaken til denne feilmeldingen?

- ☐ Taggen *SCRIPT* er ulovlig i dokumentet sin *HEAD*. Den må legges i *BODY*.
- ☐ Taggen mangler et påkrevd attributt *DEFER*.
- ☐ Når JavaScript-koden kjøres fra HEAD av dokumentet finnes ikke ennå H1 taggene i nettleser sin hukommelse.
- ☐ Navnet på taggen må skrives med store bokstaver, dvs. ikke *h1*, men *H1*.
- ☐ Første *H1*-element har indeks 1. Riktig er derfor *[1]*, ikke *[0]*.
- ☐ JavaScript-kode må lastes fra eksterne dokumenter. Det er ikke lov med JavaScript-kode inne i HTML dokumentet.

## c) (1%)

JavaScript betegnes som et type-svakt språk. Hva betyr det?

- ☐ Variabler kan deklarereres med en datatype, men dette er ikke påkrevd.
- ☐ Variabler har en type som bestemmes når variabelen brukes.
- ☐ Variabler har en type som bestemmes når variabelen tilordnes verdi.
- ☐ JavaScript er et språk som ikke har datatyper.
- ☐ JavaScript er et språk med få datatyper.
- ☐ JavaScript er et språk med kun svake datatyper.

## d) (1%)

JavaScript kan kjøres i en modus som kalles *strict mode*. Hva innebærer *strict mode*?

- ☐ I *strict mode* tillates kun enkle JavaScript-konstruksjoner som også blir forstått av gamle nettlesere.
- ☐ *Strict mode* er en versjon av JavaScript laget for Internet Explorer 6.
- ☐ *Strict mode* er en funksjonell utgave av JavaScript.
- ☐ *Strict mode* angir at koden skal tolkes som *TypeScript*.
- ☐ Med *strict mode* vil bla. bruk av variabler som ikke er deklartert gi JavaScript feil.
- ☐ Med *strict mode* vil bla. bruk av **class** gi JavaScript feil.
- ☐ Med *strict mode* tillates bla. bruk av **public** og **private** sammen med **class**.

## e) (1%)

Operatoren "===":

- ☐ Brukes for å sjekke om to variabler har lik type.
- ☐ Brukes for å sjekke om to variabler har samme type og lik verdi.
- ☐ Brukes for å sjekke om to variabler har samme verdi uansett type.
- ☐ Brukes for å sjekke om alle elementer i to lister har samme verdi.
- ☐ Brukes for å sjekke for likhet av ikke-primitive datatyper.
- ☐ Brukes for å sjekke for likhet av primitive datatyper.

## f) (1%)

JavaScript-koden under er brukt i en web-applikasjon:

```
const person = { id: 121, navn: "Ole" };  
  
let {id:studnr,navn:fornavn} = person;
```

Hvilket av utsagnene under er korrekt?

- ☐ Variabel *studnr* får verdi 121 og variabel *fornavn* får verdi *Ole*.
- ☐ Variabel *id:studnr* får verdi 121 og variabel *navn:fornavn* får verdi *Ole*.
- ☐ Variabel *id:studnr* tilordnes objektet **person**, mens variablene *navn:fornavn* blir *undefined*.
- ☐ Variabel *id* får verdi 121 og variabel *navn* får verdi *Ole*.
- ☐ Koden vil feile da objektet **person** må være på andre siden av likhetstegnet ved tilordning.
- ☐ Koden vil feile da kolon, dvs. tegnet ":" ikke er lov i variabelnavn.
- ☐ Objektet **person** får verdi *{id:studnr,navn:fornavn}*.

## g) (1%)

JavaScript-koden under er brukt i en web-applikasjon:

```
const f = x => 2*x
```

Hvilket utsagn er korrekt?

- ☐ Verdien til *x* tilordnes *f*, som som så tilordnes variabelen  $2 \cdot x$ .
- ☐ Verdien til  $2 \cdot x$  tilordnes til variabelen *x*, og *f* settes så lik *x*.
- ☐ Konstanten *f* settes lik maksimalverdien av *x* og  $2 \cdot x$ .
- ☐ Konstanten *f* settes lik maksimalverdien av *x* og 2.
- ☐ Konstanten *f* er en funksjon med parameter *x* som returnerer  $2 \cdot x$ .
- ☐ Så lenge *x* er mindre enn 1 doubles verdien. Første verdi større enn 1 tilordnes så til konstanten *f*.

## h) (1%)

JavaScript-kode i en web-applikasjon inneholder følgende konstanter med data for studenter:

```
const ole = {'id':5, 'navn': 'Ole'};  
const anne = {'id':7, 'navn': 'Anne'};  
const gro = {'id':9, 'navn': 'Gro'};  
const hans = {'id':11, 'navn': 'Hans'};
```

Parameter *id* er unik og identifiserer studenten. Hvilken datastruktur er mest egnet, og hvorfor?

- ☐ Bruk en **Array**. Det gjør det enklest å legge til og fjerne elementer.
- ☐ Bruk en **Map**, med nøkkel *id*. Da vil hver student kun forekomme en gang.
- ☐ Bruk et **Set**. Da vil hvert element kun forekomme en gang.
- ☐ Bruk et JavaScript **JSON** objekt da data er formatert som **JSON**.
- ☐ Bruk et **RegExp** objekt da parameter *navn* forekommer regelmessig.
- ☐ Bruk **Symbol** da *id* er en unik verdi.

## i) (1%)

JavaScript-koden under er brukt i en web-applikasjon:

```
const studentInfo = `${student.navn} har id ${student.id}`;
```

Hvilket utsagn er korrekt?

- ☐ Konstanten *studentInfo* inneholder teksten *\${student.navn} har id \${student.id}*.
- ☐ Nøkkelordet *const* må endres til *let* eller *var* da objektet *student* ikke er konstant.
- ☐ Konstanthavnet *studentInfo* må endres til *student* da verdien fylles fra et objekt *student*.
- ☐ Konstruksjonene *\${}* vil erstattes med siste element fra listen *student* som har parametre *navn* og *id*.
- ☐ Når *studentInfo* benyttes vil *\${}* konstruksjonene bli erstattet med verdier fra objektet *student*.
- ☐ Når *studentInfo* opprettes vil *\${}* konstruksjonene bli erstattet med verdier fra objektet *student*.

## j) (1%)

JavaScript-koden under er brukt i en web-applikasjon:

```
function lagInnhold(element,content) {  
    element.innerHTML = content;  
}
```

Innholdet i parameteren *content* kan være basert blant annet på input fra bruker.

Hvilket utsagn er korrekt?

- ☐ Koden er uproblematisk da nettleser kun vil tillate bruker å legge inn ufarlig kode i dokumentet.
- ☐ Koden er uproblematisk da nettleser vil ufarliggjøre all kode før det legges inn i dokumentet.
- ☐ Koden er farlig hvis *element* representerer et *SCRIPT* element, men ufarlig for elementer som *P* og *DIV*.
- ☐ Koden lar bruker injisere HTML og CSS i dokumentet, men er likevel nokså ufarlig da JavaScript kode i bruker-input ikke vil bli utført av nettleser.
- ☐ Koden gjør applikasjonen sårbar for angrep, men kan ufarliggjøres ved å erstatte *innerHTML* med *outerHTML*.
- ☐ Koden gjør applikasjonen sårbar for angrep og må aldri brukes. Problemet er det samme med *outerHTML*.
- ☐ Koden vil feile hvis *content* er en ren tekststreng uten verken HTML eller CSS.

## Antagelser/bemerkninger:

Dette feltet kan stå tomt. Dersom noe er uklart i oppgavene over så kan du skrive ned eventuelle antagelsene du har gjort her. Marker kommentarene dine med oppgavenummer. Skriv for eksempel:

// Oppgave b).

\*Dine antagelser...\*

**B**   *I*   U   |   ☰   ☷   á

0 / 10000 Word Limit



## Seksjon 5

# Oppgave 4 (15%) - Tråder og trådsikkerhet

Se oppgave på papir. For å svare, velger du «Administrer vedlegg». Så

Nytt vedlegg

Kode

Kodespråk - Java

Da får du åpnet en ny fane der du skriver koden.

For å sikre automatisk lagring, følg denne oppskriften

1. Start med å gi vedlegget et navn. For eksempel "Oppgave 4"
2. Klikk lagre. Da aktiverer du automatisk lagring.
3. Start å skrive kode

Når du har svart, går du tilbake til hovedfanen.

**B** *I* U | ☰ ☷ á

0 / 10000 Word Limit

## Seksjon 6

# Oppgave 5 (15%) - JavaScript program

Se oppgave på papir. For å svare, velger du «Administrer vedlegg». Så

Nytt vedlegg

Kode

Kodespråk - JavaScript

Da får du åpnet en ny fane der du skriver koden.

For å sikre automatisk lagring, følg denne oppskriften

1. Start med å gi vedlegget et navn. For eksempel "Oppgave 5"
2. Klikk lagre. Da aktiverer du automatisk lagring.
3. Start å skrive kode

Når du har svart, går du tilbake til hovedfanen.

**B** *I* U |   á

10000 Word Limit

## Kode (som du kan klippe og lime inn i besvarelsen din)

I oppgaven skal dere fylle inn den manglede koden i en JavaScript-klasse **KonkurranseKontroller**. Se oppgavetekst på papir for detaljer.

```
class KonkurranseKontroller {
  #liste = [];

  // Legg til her eventuelle flere private felt

  constructor(felt1ref,felt2ref) {
    this.#navnelement =
      felt1ref.querySelector("input[type='text']");
    this.#tidelement =
      felt1ref.querySelector("input[type='number']");
    const regbt = felt1ref.querySelector("button");
    regbt.addEventListener("click",
      () => { this.#regdeltager() }
    );

    // Legg inn kode her for felt2
  }

  #regdeltager() {
    // Legg inn kode her for å registrere deltager
  }

  // Legg til metod(er) for å kunne søke på deltager
}

const felt1 = document.getElementById("registrering");
const felt2 = document.getElementById("resultat");
new KonkurranseKontroller(felt1,felt2);
```

## Seksjon 7

# Oppgave 6 (40%) - Web backend med Spring MVC

Se oppgave på papir. For å svare, velger du «Administrer vedlegg». Så

Nytt vedlegg

Kode

Kodespråk - Java

Da får du åpnet en ny fane der du skriver koden.

For å sikre automatisk lagring, følg denne oppskriften

1. Start med å gi vedlegget et navn. For eksempel "Oppgave 6"
2. Klikk lagre. Da aktiverer du automatisk lagring.
3. Start å skrive kode

Når du har svart, går du tilbake til hovedfanen.

</>