Innhold

Del 1: Flervalgsoppgaver

Denne delen inneholder flervalgsoppgaver. For hver oppgave er det spesifisert hvor mange poeng det er mulig å oppnå. Husk å lese hele oppgaveteksten før du svarer.

Totalt er det mulig å oppnå 16 poeng på denne delen.

Alle disse oppgavene er automatisk rettet.

1.1 DevOps (1p)

Fullfør setningen.

DevOps er en samling av praksiser og prinsipper med formål om å

bryte ned ✓ (bryte ned, bygge opp) siloer i en organisasjon.

1.2 Kotlin - Operator (2p)

Hvilken av følgende operatører i Kotlin brukes for å gi en alternativ verdi hvis en nullable variabel er null?
Velg ett alternativ:

- ?: (elvis) operatoren ✓
- !! (non-null assertion) operatoren
- :: (scope resolution) operatoren
- ?. (safe call) operatoren

1.3 Android - layout (2p)

Du ønsker å vise en **horisontal liste** med **ukjent antall elementer**.
Hvilken av følgende komponenter bør du benytte deg av?
Velg ett alternativ:

- Column
- LazyRow ✓
- LazyColumn
- Row

1.4 Funksjonelt krav (2p)

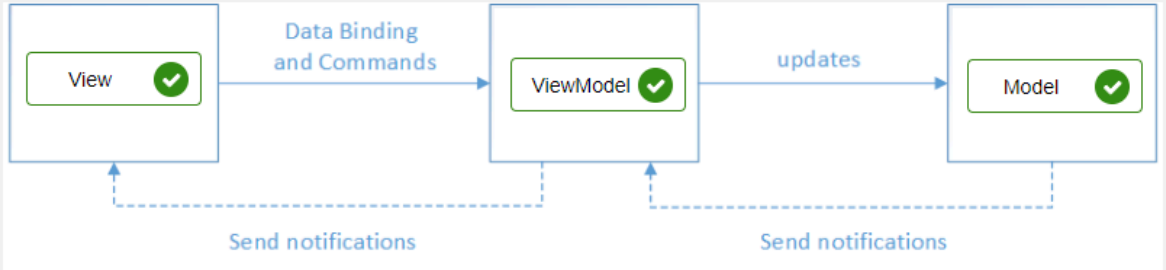
Hvilken av følgende krav er funksjonelt?
Velg ett alternativ:

- Appen skal ha grønn, gul eller rød farge basert på luftkvalitet
- Appen skal være skrevet i Kotlin, med bruk av Android Studio
- Appen skal vise et kart over målestasjoner i Norge ✓
- Appen skal håndtere minst 1000 brukere samtidig

1.5 MVVM (2p)

Identifiser og plasser følgende deler korrekt i arkitekturskissen fra MVVM:

Hjelp



(Skissen er hentet fra Microsoft)

1.6 Kotlin - filter (2p)

Se på følgende kode:

```
val list = listOf(1, 2, 3, 4, 5, 6, 7, 8, 9)
val result = list.filter{ PREDICATE }

println(result)
```

Hva må byttes ut med PREDICATE for at følgende skal skrives ut i terminalen?
>> [1, 4, 7]
Velg ett alternativ:

- it % 2 == 1
- it % 2 == 0
- it % 3 == 0
- it % 3 == 1 ✓

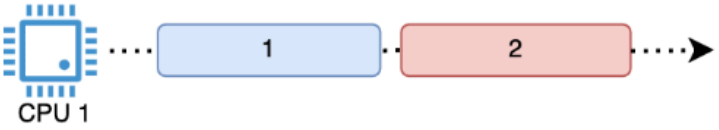
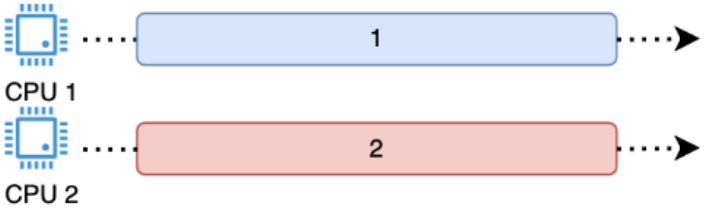
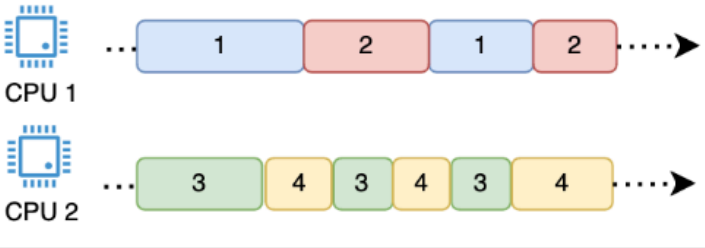
1.7 Samtidighet og parallellitet (2p)

Koble beskrivelse med riktig figur. Hver "oppgave" som gjøres av cpu-en er markert som tall i egen farge, "oppgaven" er først ferdig siste gang boksen repeteres.

Betydning av ordene samtidig (concurrent) og parallell er lik som i forelesningen som omhandler parallellitet.

Dra beskrivelsen til boksen til venstre for figuren den hører til.

Hjelp

Ikke parallell og ikke sam g	
Samtidig, men ikke para	
Parallell, men ikke samti	
Parallell og samtidig	

Del 2: Spørsmål fra pensum

Denne delen består av 5 deloppgaver og gir maks 25 poeng

2.1 Mobb-programmering (7p)

a) Beskriv konseptet Mobb-programmering. I besvarelsen skal du inkludere en beskrivelse av de to rollene som brukes i Mobb-programmering. (3p)

Det viktigste å få med her er en grei beskrivelse av Mobb-programmernig, nemlig at hele teamet (flere folk) sitter sammen om å løse et problem, i samme rom, samtidig. For å få uttelling er det viktig å nevne at det kun er **en datamaskin/ett tastatur** som går på rundgang, gjerne med en "timer" slik at rullering skjer hyppig. Besvarelsen skal også inneholde en forklaring av rollene "sjåfør" (den som har tastaturet og gjør det de andre sier) og "navigatør" (som forteller sjåfør hva som skal skrives). Alltid én sjåfør og flere navigatører.

b) Beskriv to fordeler og to ulemper med Mobb-programmering. (4p)

Det gis 1p for hver fordel og ulempe som kandidaten nevner.

Fordel	Ulempe
• Mindre behov for detaljplanlegging • Mindre venting (positiv type venting der du kan lære) • Mindre forstyrrelser • Kontinuerlig kunnskapsdeling • Felles forståelse av problem og løsning • Fellesskap gir trygghet • Teamet tåler at enkeltmedlemmer slutter	• Stor overgang fra å sitte alene med et problem, ikke alle synes overgangen er enkel • Ulike personligheter og behov • Mister personlig eierskap til koden (kan være viktig for noen) • Kan være vanskelig å finne plassen sin i en «erfaren» mobb • Noen vil kanskje aldri trives i en mobb

Det gis også poeng for andre fordeler / ulemper så lenge de gir mening.
Fordeler/ulemper skal **beskrives**. Maks 2p dersom ved oppramsing.

2.2 Sikkerhet (6p)

Nevn tre av sikkerhetskomponentene i Android-utvikling og forklar hvordan disse kan håndteres.

- Input-validering
- Kryptografi
- Autentisering og Autorisering
- Nettverkskommunikasjon
- Interaksjon med Mobilplattformen
- Kodekvalitet og Exploit-Mitigation
- Anti-Tampering og Anti-Reversing

For opptil tre sikkerhetskomponenter:

- 1p for hver nevnte sikkerhetskomponent.
- 1p for en kort forklaring for den. Se slides fra forelesning om dette.

2.3 Universell utforming/Universal design (5p)

Hvordan kan vi lage mer tilgjengelige apper? Hvorfor er dette viktig?

Et godt svar er å henvise til rammeverk og standarder som finnes for dette, som f.eks.

- WCAG
- DIGDIR
- WAD

Dersom en kandidat går teknisk til verks med å nevne spesifikke komponenter i Android som legger til rette for universell utforming, er det viktig at kandidaten beskriver **hvordan** komponentene er med på å lage mer tilgjengelige apper - ikke bare ramse den opp.

Svaret skal i tillegg inneholde en begrunnelse på hvorfor dette er viktig. Dersom denne delen av oppgaven ikke besvares, skal 2 poeng trekkes.

2.4 Git (5p)

a) **Forklar noen fordeler med å benytte seg av flere brancher, fremfor kun en main-branch i teamarbeid. (2p)**

- Flere utviklere kan samarbeide parallelt uten å forstyrre hverandre, kan altså eksperimentere i egen branch uten å påvirke produksjon / andre.
- Sporbarhet. Hver branch kan representere hver sin brukerhistorie, slik at det som utvikles lett kan korreleres til backloggen.

Andre rimelige fordeler gir også poeng.

b) **Forklar kort hvilke utfordringer som kan oppstå med mange isolerte brancher som har lang levetid. Hvordan kan dette håndteres? (3p)**

- Det kan være vanskelig å ha oversikt over endringer.
- Flere utviklere kan utvikle på en utdatert branch dersom ikke endringer regelmessig hentes ned i branchen.
- Å flette koden sammen (merge) vil kunne føre til konflikt (git conflict) og denne må løses.

Hvordan håndtere: etablere gode rutiner og retningslinjer. Følge en git strategi og få hele teamet til å forstå og følge denne.

2.5 Deserialisering (2p)

Så lenge forklaringen er riktig gis det full pott på denne oppgaven. Forklaringen bør inneholde noe om at det er prosessen å gjøre om fra et gitt dataformat til objekter. Ellers gir det 0p.

Del 3: Caseoppgave

Denne delen inneholder caseoppgaven. For hver oppgave i delen er det spesifisert hvor mye poeng det er mulig å oppnå.

Totalt er det mulig å oppnå 44 poeng på denne delen.

3.1 Prosjektoppstart (14p)

Beskriv de mest sentrale aktørene for dette systemet og hvilke som er primære aktører og hvilke som er sekundære aktører. Begrunn svaret. (4p)

Eksempler:

- Kunder (primær)
- Banktjenester (sekundær)
- Ansatte i Børekekspressen (sekundær)

Her er det ingen fasit, men kandidatene bør begrunne valg av aktører og hvorfor de er primære/sekundære.

Hvilke(n) forskningsmetode(r) kan være fornuftig i oppstartsfasen av prosjektet? Begrunn svaret ditt. (4p)

Her er det ingen fasit, det vi er ute etter her er forståelse av de ulike forskningsmetodene, satt i kontekst av at dette er under en prosjektoppstart. Viktigste er en kort begrunnelse på hvorfor oppgitt forskningsmetode(r) egner seg.

Hvordan kan vi inkludere tverrfaglighet og brukersentrert design i startfasen av et slikt prosjekt? Beskriv noen utfordringer rundt tverrfaglighet i "smidige" team. (6p)

På denne oppgaven ønsker vi drøfting / tanker fra forelesningen om Smidig utvikling og design. Her er det naturlig å f.eks. nevne gjensidig læring og at en jobber mer på tvers av spesialitetene sine og planlegger sammen. Konseptet sprint 0 kan også nevnes.

Potensielle utfordringer:

- Mye ventetid
- Usikkerhet rundt hva de andre holder på med
- Lite forståelse for hva de andre holder på med
- En jobber bare med det en selv kan fra før

3.2 Diagram (10p)

a) Lag en brukerhistorie for appen til Børekekspressen. (2p)

For å få poeng på denne oppgaven skal kandidaten skrive en brukerhistorie, dvs. beskrive et krav på formen: "*Som [ROLLE], ønsker jeg [FUNKSJON] for å oppnå [NYTTEVERDI]*".

Brukerhistorien trenger ikke å være "perfekt" definert, men skal inkludere rolle, funksjon og nytteverdi. Dersom svaret ikke er en brukerhistorie, eller ligner på en brukerhistorie, skal det gis 0 poeng.

b) Modeller et UML diagram som visualiserer denne brukerhistorien. Du kan selv velge hvilket UML diagram du ønsker, men inkluder hvorfor du valgte diagrammet og vær tydelig på formålet med diagrammet. (8p)

Besvarelsen bør:

- Ha med en beskrivelse av hvorfor det spesifikke diagrammet ble valgt.
- Ha med et diagram som representerer den definerte brukerhistorien fra forrige deloppgave.
- Korrekt syntaks på valgt diagram.
- Ha med noe forklarende tekst og antagelser.

3.3 Implementasjonsoppgave (10p)

a) Definer en enum som viser de ulike Børek-variantene (fetaost, spinat, kylling, sopp). Kotlin skal benyttes som programmeringsspråk. (2p)

2 poeng

```
enum class Borek {
    Spinat,
    Fetaost,
    Kylling,
    Sopp
}
```

1 poeng

```
enum Borek {
    Spinat,
    Fetaost,
    Kylling,
    Sopp
}
```

Det gis poeng for enum-er med steketid definert i enum-en, så lenge den er definert riktig. Alle andre svar gir 0 poeng.

b) (...) Kort oppsummert skal metoden sjekke hvilke børek som er klare til å tas ut av ovnen basert på type og steketid. Metoden skal så returnere en liste av disse børek. (8p)

Løsningsforslag:

- https://pl.kotl.in/_qCa7TblJ (steketid definert i logikken)
- https://pl.kotl.in/YiDZ6bf-Z (steketid definert i enum)

Det er mulig å få full pott på denne oppgaven selv om den ikke er løst på måten som er vist i lenken over. Det bedømmes på:

- Riktig deklarasjon av funksjon (riktig parameter og returtype)
- Riktig logikk (at funksjonen gjør det den skal)

- “Kotlin”-idiomati, f.eks
    - Riktig bruk av operasjoner på Collections
    - Bruk av ”when” i stedet for “if”
    - Returnere hele uttrykket direkte

Det er mulig å få inntil 2 poeng på hele deloppgaven dersom metoden er implementert som pseudokode / i et annet programmeringsspråk.

3.4 Feil implementasjon

Denne oppgaven er relatert til initialisering av skjermer med data, spesielt det at ViewModel-en trenger å få vite en børek sin ID for å kunne hente data om riktig børek.

1. Hvorfor blir metoden `loadBorek` kalt mer enn én gang når detaljvisningskjermen åpnes? (3p)

Grunnen til dette er at kallet ***viewmodel.initialize*** vil skje hver gang skjermkoden rekomponeres. Skjermen vil rekomponeres hver gang UIstate oppdaterer seg.

2. Forklar hvorfor det bare blir ett kall på `loadBorek` med “knapp-løsningen”. Hva skiller den løsningen fra den originale versjonen? (3p)

I knapp-løsningen er kallet puttet inn knappen sin onClick event-handler (som ikke en del av Composable-skopet som blir kalt på nytt ved komposisjoner). Dette gjør at kallet kun gjøres når knappen trykkes på.

3. Forklar hvordan du kan løse dette på en bedre måte. Forklar hvorfor måten du foreslår fungerer og er bedre enn løsningen med knapp. (4p)

- Det å gjøre initialize-metoden idempotent (binær operasjon) vil gi full uttelling. Dette kan f.eks. gjøres med å introdusere en boolsk variabel i ViewModel-en som brukes for å sjekke om initialize-metoden har blitt kalt på fra før. Hvis den er kalt på før kalles ikke loadBorek på nytt.

```
private var initializeCalled = false

// Funksjonen er idempotent så lenge den kun blir kalt fra UI-tråden
@MainThread
fun initialize(borekId: String) {
    if (initializeCalled) return
    initializeCalled = true

    loadBorek(borekId = borekId)
}
```

- Det å benytte seg av SavedStateHandle sammen med å erstatte initialize-metoden for å gi ViewModelen ID-en er også en løsning som gir full uttelling.

Løsninger som gir noe uttelling:

- Caching i form av lagring i en database eller lignende vil kunne gi opptil 2 poeng. Dette er ikke en løsning som fikser problemet som er beskrevet med antall kall på ViewModel-metoden loadBorek. Men, det er en løsning som kan potensielt redusere antall api-kall.
- Putte initialize-kallet inn i en LaunchedEffect vil kunne gi opptil 2 poeng. Dette gjør at brukeren ikke trenger å trykke på en knapp, men launchedEffect er egentlig ment for å kjøre suspend-funksjoner trygt i composables.

Uavhengig av foreslått løsning er det viktig med en begrunnelse på hvorfor løsningen er bedre enn løsningen med knapp.For flere detaljer, se Android-dokumentasjonen.

Del 4: Refleksjon

Denne delen inneholder refleksjonsoppgaver.

Totalt er det mulig å oppnå 15 poeng på denne delen.

4.1 Teamarbeid - refleksjon (15p)

Felles for alle refleksjonsoppgavene er at refleksjonene skal inneholde begrunnelser og ikke bare oppramsing av diverse tanker.

a) Reflekter rundt hva du mener er en "god" og en "dårlig" teamarbeider. (6p)

I tillegg til generelt om refleksjon (se over), så skal svaret inneholde:

- Tanker rundt hva en “god” teamarbeider er og hvorfor.
- Tanker rundt hva en “dårlig” teamarbeider er og hvorfor.

Det gir lite uttelling å si at “dårlig” er **kun** det motsatte av hva “god” er.

b) Se for deg at du er teamleder i et team der ett teammedlem er mye borte og viser lite interesse til å bidra. Reflekter hvordan du ville gått frem for å forsøke og løse dette problemet? (6p)

I tillegg til generelt om refleksjon (se over), så skal svaret inneholde:

- Hvordan gå frem for å løse problemet.
- Ulike metoder for dette.

c) I løpet av prosjektarbeidet har dere hatt muligheten til å gjennomføre teamaktiviteter. Hvilken eller hvilke teamaktivitet(er) mener du ga deg mest utbytte? Begrunn svaret ditt. (3p)

Her gis det poeng så lenge kandidaten beskriver en eller flere teamaktiviteter som de mener ga mest utbytte. Videre skal svaret inneholde en begrunnelse på dette.

Det skal mye til for å **ikke** få full pott på denne.