

Innhold

Innhold	0
Del 1: Oppgaver som rettes automatisk (22p)	1
Del 2: Diverse pensum	1
2.1 Diverse kort svar fra pensum (18p)	1
2.2 String? (6p)	3
2.3 Partition (5p)	3
2.4 Refleksjon - teamarbeid (8p)	4
Del 3: Caseoppgave	6
3.1 Prosjektoppstart (27p)	6
3.2 Representasjon av fremmed art (7p)	7
3.4 Bug (7p)	8

Del 1: Oppgaver som rettes automatisk

Denne delen består av **tolv** deloppgaver og gir maks 22 poeng. Logg inn i Inspira for å se poengfordeling og fasit for de ulike oppgavene.

Del 2: Diverse pensum

Denne delen består av fire deloppgaver og dekker diverse deler fra pensum. På denne delen er det mulig å oppnå totalt 37 poeng.

2.1 Diverse kort svar fra pensum (18p)

Denne oppgaven består av små oppgaver der kandidaten **maks skal bruke fire setninger** på å besvare hver av dem. Dersom kandidatene bruker mer enn fire setninger, eller svarer feil gis det 0 poeng. Her skal det mye til å få 1 av 2, siden som regel har kandidaten riktig eller feil - lite i mellom.

a) Hva er forskjellen på parprogrammering og mobbprogrammering?

Eksempel på svar:

Parprogrammering gjøres med kun to utviklere, mens mobbprogrammering gjøres med hele teamet. I mobbprogrammering bytter en hyppigere på hvem som er navigatører og sjåfører.

b) Hva er den magiske trekanten (magic triangle)?

Eksempel på svar:

Den magiske trekanten består av kost, tid og kvalitet. Trekanten forsøker å illustrere at det er krevende å lage et system som leverer på alle de tre punktene samtidig. F.eks. er det vanskelig å levere på tid og kost, uten at det går utover kvaliteten.

c) Hva er et case-studie?

Eksempel på svar:

Case-studier legger vekt på å studere fenomener i sine naturlige omgivelser. Teorier bør styre datainnsamling og analyse.

d) Forklar forskjellen på en gruppe og et team innen systemutvikling.

Eksempel på svar:

En gruppe er en samling av personer som jobber individuelt, med en leder som styrer gruppen og koordinerer individuelle bidrag, ofte med lav felles forpliktelse. **Et team** er en samling av individer som styrer seg selv og deler et felles oppdrag med ulike målsetninger. Team-medlemmene er forpliktet til hverandre og til teamets målsetninger.

e) Hva er WCAG, og hvorfor er det viktig?

Eksempel på svar:

WCAG er internasjonale retningslinjer for universell utforming og beskriver krav til

blant annet struktur, kontrast, teksting, tastaturnavigasjon og alternative tekster. WCAG er viktig fordi det sikrer at flest mulig, også personer med funksjonsnedsettelser, kan bruke digitale tjenester og fordi det ligger til grunn for lovkrav.

f) Forklar følgende to begreper innen git: commit og branch.

Eksempel på svar:

En **commit** er en lagret "snapshot" av endringene dine i et git-repo, med melding og historikk, slik at du kan gå tilbake eller sammenligne. En **branch** er en egen utviklingslinje i prosjektet, som lar deg utvikle og teste endringer isolert fra andre før de eventuelt flettes inn i f.eks. `main`.

g) Forklar noen forskjeller på dataformatene XML og JSON

Eksempel på svar:

XML bruker nestede tagger og er mer tungt og ordrik, mens JSON er lettere og mer kompakt. XML har støtte for attributter og avanserte skjema (f.eks. XSD), mens JSON typisk er enklere strukturert. JSON er i dag mest brukt i web-API-er og lett å bruke i kode, mens XML oftere finnes i eldre eller dokumenttunge systemer.

h) Hva er en code smell?

Eksempel på svar:

En code smell er et tegn på at koden kan være dårlig strukturert eller vanskelig å vedlikeholde, selv om den fungerer. Eksempler er duplisert kode, altfor store klasser eller funksjoner, og utydelige navn. Code smells brukes som hint om at det kan være lurt å refaktorere.

i) Hvordan vurderes en architecture "trade-off"?

Eksempel på svar:

En arkitektur-"trade-off" vurderes ved å veie fordeler og ulemper ved ulike løsninger opp mot hverandre. Man ser på hvordan hver løsning scorer på kvalitetsegenskaper som ytelse, sikkerhet, skalerbarhet, kostnad og kompleksitet. Valget blir den løsningen som best oppfyller de viktigste kravene, gitt begrensninger i tid, budsjett og risiko.

2.2 String? (6p)

Se for deg at følgende variabel er definert: `val text = String?`

a) Hva er grunnen til at du ikke kan kalle `text.length` direkte? (1p)

Du kan ikke kalle `text.length` direkte fordi `text` er av typen nullable String.

b) Skriv et kort kodeeksempel som bruker smart cast til å kunne kalle `length` på `text` (2p)

Smart cast etter en null-sjekk gjør at kompilatoren behandler `text` som ikke-null inne i blokken, eksempel:

```
if (text != null) {  
    val len = text.length // smart cast til String  
}
```

For å smart caste må det gjøres en if-check, when-uttrykk eller let.

c) Skriv et kort kodeeksempel som bruker enten non-null assertion-operatoren eller Elvis-operator til å kalle `length` på `text`. Forklar forskjellen på disse to operatorene, og hvorfor du valgte den ene over den andre i eksempelet ditt. (3p)

NON-NULL ASSERTION

```
val len: Int = text!!.length
```

ELVIS

```
val len: Int = text?.length ?: 0
```

Her kan kandidaten velge det ene eller andre, så lenge begrunnelsen er bra KAN det gis full pott. Men generelt vil Elvis-operator være mest hensiktsmessig siden non-null assertion kan kaste en `NullPointerException`, noe Elvis-operator ikke gjør siden den definerer en ved at en trygg "fallback-verdi".

2.3 Partition (5p)

```
data class Student(val navn: String, val poeng: Int)
```

```
val studenter = listOf(  
    Student("Ola", 42),  
    Student("Kari", 87),  
    Student("Anne", 65),  
    Student("Per", 31)  
)
```

```
fun main() {
```

```

...

print("Bestått: ")
println(bestått)
print("Ikke bestått: ")
println(ikkeBestått)
}

```

Skriv kode som bruker partition der det står ... slik at utskriften blir:

Bestått: [Student(navn=Kari, poeng=87), Student(navn=Anne, poeng=65)]

Ikke bestått: [Student(navn=Ola, poeng=42), Student(navn=Per, poeng=31)]

En student har bestått med 43 poeng eller mer. Programmet skal gi riktig output også for andre lister med andre studentobjekter (5p)

Forslag til løsning:

```

fun main() {
    val (bestatt, ikkeBestatt) = studenter.partition { it.poeng >= 43 }

    print("Bestått: ")
    println(bestatt)
    print("Ikke bestått: ")
    println(ikkeBestatt)
}

```

Dersom partition ikke brukes skal det trekkes kraftig i denne oppgaven.

2.4 Refleksjon - teamarbeid

Beskriv minst to smidige praksiser dere benyttet dere av i teamarbeidet i IN2000 og hvordan dere gjorde endringer av disse praksisene underveis i teamarbeidet. NB: Ikke nevne navn/teamnummer i besvarelsen! (8p)

Dette er en refleksjonsoppgave, dermed er det viktigste at studentene klarer å reflektere, fremfor å kun beskrive. En god besvarelse vil klare å henvise både til pensum (primært Sommerville kap 3 og forelesningen om teamarbeid), og relatere til eksempler, for de to (eller fler) smidige praksiser som beskrives. Det MÅ ikke være egne erfaringer fra prosjektarbeidet, men det må være eksempler og begrunnelser i noen form.

- Mulige praksiser som kan trekkes inn.
- Daily stand up/daglige møter
- Retrospektiv møter
- Sprint planning møter/Sprint review møter
- Fasilitering av arbeidsprosessen

- Fokus på gode og effektive møter
- Bruk av digitale kanaler for skriftlig kommunikasjon
- Parprogrammering som samarbeidsverktøy
- Bruk av product backlog (gir oversikt)
- Bruk av flytverktøy (e.g. ulike boards)
- Bruk av teamkontrakt

Andre praksiser som ikke listes her kan også være aktuelle, det viktigste er god refleksjon rundt **hvorfor** det er gode praksiser. Noen vil nok nevne Scrum, Scrumban eller Kanban etc., det er også greit, selv om det er noe mer overordnet (mer rammeverk og prosessmodell) enn smidige praksiser.

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 41 poeng på denne delen.

3.1 Prosjektoppstart

a) Beskriv interessenter og hvilken interesse de har av systemet, Beskriv også hvilke av interessentene som er primære aktører og hvilke som er sekundære aktører. Begrunn svaret. (5p)

Bør vite hva en interessent er, og hva og hva primær og sekundær aktør er. Greit å ha med minst en primær og en sekundær aktør, men gjerne flere.

Eks:

Interessenter:

- Bruker (turgåere, ...) (primæraktør)
- Selskapet som har utviklet systemet (sekundær aktør)
- Forskere (biologer etc.)
- Myndigheter som er opptatt av miljø, gårdsbruk, planter og lignende.

Primæraktør: Bruker

Sekundæraktør: Selskapet som har utviklet mobilappen (f.eks. ønsker å hente ut), utviklerne (vedlikeholder systemet).

b) Lag et UML aktivitetsdiagram som viser funksjonsflyt fra bruker åpner appen til registreringen fullføres, inkludert beslutninger og minst en alternativ flyt. (9p)

Her må kandidaten skjønne hva en aktivitetsdiagram er (flyt/prosess), med start og slutt symbol, aktivitet (rektangel) og diamant («if then else»). Bør ha minst en alternativ flyt og diamantsymbol for full pott.

c) Lag et UML tilstandsdiagram som viser hvilke tilstander systemet er i fra bruker åpner appen til registreringen fullføres (9p)

Her må kandidaten skjønne hva en tilstandsdiagram er (hendelsesorientert), med start og slutt symbol, vise at systemet går fra en tilstand til en annen, og også en fordel å beskrive hva som gjelder i en tilstand (bruke do/).

d) Beskriv hovedfokus og perspektiver på de to diagramtypene aktivitetsdiagram og tilstandsdiagram (4p)

Aktivitetsdiagram fokuserer på dataflyt og prosessen (input-aktivitet-output), er mer kundenært enn mange andre diagrammer og brukes ofte i samarbeid med kunden for å vise forretningsprosesser og arbeidsprosesser. Tilstandsdiagram viser hvordan systemet reagerer på interne og eksterne hendelser. Baseres på antagelsen av at systemet har et

endelig antall tilstander og at hendelser (stimuli) fører til at systemet går fra en tilstand til en annen.

3.2 Representasjon av fremmed art

a) Basert på casebeskrivelsen, definer et JSON-objekt for å representere en fremmedart. (3p)

For eksempel:

```
{
  "id": "IN2000",
  "artNavn": "IFI Rose",
  "lokasjon": {
    "latitude": 59.912730,
    "longitude": 10.746090
  },
  "timestamp": "2019-05-28T14:23:10Z",
  "bildeUrl": "https://example.com/xxx.jpg",
  "aiScore": 87,
  "status": "fremmed",
  "manuellVerifikasjon": false
}
```

Kommentar:

Det viktigste med denne oppgaven er ikke nødvendigvis å ha med ALLE feltene fra casebeskrivelsen, men bør så klart ha noen av dem. Videre bør kandidaten noenlunde ha definert riktig JSON struktur som JSON array/object som rotelement, og fnutter før og etter komma for hver verdi.

b) Basert på JSON-objektet du definerte i forrige deloppgave (a), definer en data class for å representere JSON-objektet. (3p)

```
data class FremmedArtFunn(
    val id: String,
    val artNavn: String,
    val lokasjon: Lokasjon,
    val timestamp: String,
    val bildeUrl: String,
    val aiScore: Int,
    val status: String,
    val manuellVerifikasjon: Boolean
)
data class Lokasjon(
    val latitude: Double,
    val longitude: Double
)
```

Kommentar:

I denne oppgaven er det viktigste at data classen er definert riktig i henhold til JSON-objektet som ble besvart som deloppgave a. At det ikke er lagt til/fjernet felter. Videre er viktig at en data class er definert på riktig måte.

c) Hva kalles prosessen som tar en JSON string (eller fil) og gjør den om til et objekt/datastruktur? (1p)

Svar: Deserialisering

3.4 Bug (7p)

Se for deg at følgende feil blir rapport inn fra en bruker:

Jeg opplever å miste informasjon fra appen. Mer konkret, etter jeg har tatt bilde av en plante, og venter på svar fra AI tjenesten, kan jeg rotere skjermen. Da henger appen veldig lenge, og så får jeg timeout som svar etter noen minutter. Feilen oppstår ikke dersom jeg ikke roterer skjermen.

a) Se for deg at du er utvikler og skal fikse den innrapporterte buggen. Synes du problemet er godt nok definert? Begrunn svaret. (2p)

Her er det ingen fasit så lenge kandidaten begrunner hvorfor buggen er godt nok / ikke godt nok definert. Så si at den er god nok/ikke god nok uten begrunnelse gir 0 poeng.

b) Hva kan være grunnen til at feilen oppstår? Foreslå en løsning på problemet. (5p)

Her kan det være flere grunner, så lenge kandidaten ramser et reelt problem og foreslår en løsning er det godt nok.

Eksempler:

- Tilstand (state) mistes ved rotasjon
- Asynkront kall er bundet til feil livssyklus (f.eks. Composable i stedet for ViewModel)
- Blokkerende arbeid eller feil trådbruk

Med tilhørende løsning på problem.

At AI API-et er nede kan være en grunn, men sannsynligvis er det en intern feil i appen, så slike kan gi noen poeng, men ikke full pott.