## Del 3: Spørsmål

Her skal dere besvare noen spørsmål. Dere vil bli vurdert på hvorvidt dere evner å

reflektere rundt egne valg, forståelsen dere viser for objektorientert programmering og

pensum i emnet. Dere skal besvare følgende spørsmål:

1.Hvilke deler av pensum i emnet dekkes i prosjektet, og på hvilken måte? (For

eksempel bruk av arv, interface, delegering osv.)

2.Dersom deler av pensum ikke er dekket i prosjektet deres, hvordan kunne dere

brukt disse delene av pensum i appen?

3.Hvordan forholder koden deres seg til Model-View-Controller-prinsippet? (Merk:

det er ikke nødvendig at koden er helt perfekt i forhold til Model-View-Controller

standarder for å få full uttelling på dette spørsmålet. Det er mulig (og bra) å

reflektere rundt svakheter i egen kode)

4.Hvordan har dere gått frem når dere skulle teste appen deres, og hvorfor har

dere valgt de testene dere har? Har dere testet alle deler av koden? Hvis ikke,

hvordan har dere prioritert hvilke deler som testes og ikke? (Her er tanken at

dere skal reflektere rundt egen bruk av tester)

5.Har dere møtt på noen utfordringer i løpet av prosjektet? Hva ville dere gjort

annerledes en annen gang?

## Del 1 - Beskrivelse

Appen vår består av et sjakkspill med grafisk brukergrensesnitt med mulighet for å lagre et pågående parti, som en senere kan laste inn og fortsette på. Flytting av brikker skjer ved å klikke på en brikke og klikke på feltet en vil flytte den til. Vi forhåndsberegner alle lovlige trekk, og de blir vist som sirkler på brettet etter en har klikket på en brikke. Algoritmen vår tar hensyn til alle regler, og den avslutter spillet både ved sjakk matt og uavgjort ved patt. Ved bondeforvandling skriver en inn den brikken en vil erstatte bonden med i et tekstfelt.

## Del 2

1.Hvilke deler av pensum i emnet dekkes i prosjektet, og på hvilken måte? (For

eksempel bruk av arv, interface, delegering osv.)

* Abstrakte klasser (Piece) -> gikk vekk fra arv
* Iterator (BoardTilesIterator)
* Comparator (lambda uttrykk)
* Lambda streams

I sjakk er det naturlig å uttrykke brikkene, brettet og rutene på brettet som objekter. Originalt hadde vi alle disse klassene, men vi valgte å droppe brettet på grunn av at vi støtte på problemer med innkapsling.

Brikkene har en overordnet abstrakt klasse Piece som de individuelle brikkeklassene arver fra. Grunnen til at vi valgte å gjøre Piece til en abstrakt klasse, er fordi dette ikke er en egen brikke som eksisterer i spillet, men det generelle konseptet om en brikke. I tillegg til at subklassene arver metoder, så er en annen fordel med at en bruker ikke kan konstruere et Piece objekt at konstruktøren til Piece kan ha skjulte parametere som blir fylt ut med forhåndsbestemte verdier i konstruktøren til brikkeklassene. Piece klassen vår inneholder en ID som blir brukt i kontrolleren til å hente riktig bilde for å vise på brettet.