| Kandidatnummer(e)/Navn: | | | |
| --- | --- | --- | --- |
| Ida Aspenes-Isaksen  Julie Bao Thi Tran | | | |
| Dato: | Fagkode: | Studium: | Ant sider/bilag: |
| 23.05.2025 | IDATT2003 | Dataingeniør | 3 / |

|  |
| --- |
| Faglærer(e): |
| Atle Olsø  Majid Rouhani |

|  |
| --- |
| Tittel: |
| Prosjekt i IDATT2003 |

|  |
| --- |
| Sammendrag: |
| Denne rapporten er skrevet i forbindelse med et prosjekt i emnet IDATT2003. Prosjektet går ut på å utvikle stigespill i tillegg til et annet valgfritt brettspill. Den endelige løsningen inkluderer et klassisk stigespill, med noen twister, i tillegg til ludo. |

*Denne oppgaven er en besvarelse utført av student(er) ved NTNU.*

Deklarasjon om KI-hjelpemidler

Har det i utarbeidingen av denne rapporten blitt anvendt KI-baserte hjelpemidler?

|  |  |
| --- | --- |
|  | Nei |
|  |  |
|  | Ja |

Hvis *ja*: spesifiser type av verktøy og bruksområde under.

Tekst

|  |  |
| --- | --- |
|  | **Stavekontroll**. Er deler av teksten kontrollert av: *Grammarly, Ginger, Grammarbot, LanguageTool, ProWritingAid, Sapling, Trinka.ai* eller lignende verktøy? |
|  |  |
|  | **Tekstgenerering**. Er deler av teksten generert av: *ChatGPT, GrammarlyGO, Copy.AI, WordAi, WriteSonic, Jasper, Simplified, Rytr* eller lignende verktøy? |
|  |  |
|  | **Skriveassistanse**. Er en eller flere av ideene eller fremgangsmåtene i oppgaven foreslått av: *ChatGPT, Google Bard, Bing chat, YouChat* eller lignende verktøy? |

Hvis ja til anvendelse av et tekstverktøy - spesifiser bruken her:

|  |
| --- |
|  |

Kode og algoritmer

|  |  |
| --- | --- |
|  | **Programmeringsassistanse**. Er deler av koden*/*algoritmene som i) fremtrer direkte i rapporten eller ii) har blitt anvendt for produksjon av resultater slik som figurer, tabeller eller tallverdier blitt generert av: *GitHub Copilot, CodeGPT, Google Codey/Studio Bot, Replit Ghostwriter, Amazon CodeWhisperer, GPT Engineer, ChatGPT, Google Bard* eller lignende verktøy? |

Hvis *ja t*il anvendelse av et programmeringsverktøy - spesifiser bruken her:

|  |
| --- |
|  |

Bilder og figurer

|  |  |
| --- | --- |
|  | **Bildegenerering**. Er ett eller flere av bildene/figurene i rapporten blitt generert av: *Midjourney, Jasper, WriteSonic, Stability AI, Dall-E* eller lignende verktøy? |

Hvis ja til anvendelse av et bildeverktøy - spesifiser bruken her:

|  |
| --- |
|  |

Andre KI-verktøy

|  |  |
| --- | --- |
|  | **Andre KI-verktøy.** har andre typer av verktøy blitt anvendt? Hvis ja spesifiser bruken her: |

|  |
| --- |
|  |

|  |  |
| --- | --- |
|  | Jeg er kjent med NTNUs regelverk: *Det er ikke tillatt å generere besvarelse ved hjelp av kunstig intelli- gens og levere den helt eller delvis som egen besvarelse.* Jeg har derfor redegjort for all anvendelse av kunstig intelligens enten i) direkte i rapporten eller ii) i dette skjemaet |

-----------------------------------------------------------------------

Underskrift/Dato/Sted

INNHOLD

Innholdsfortegnelse

[1 Introduksjon 1](#_Toc158625712)

[1.1 Bakgrunn 1](#_Toc158625713)

[1.2 Kravspesifikasjon 1](#_Toc158625714)

[1.3 Avgrensninger 1](#_Toc158625715)

[1.4 Begreper/Ordliste 1](#_Toc158625716)

[2 Teori 2](#_Toc158625717)

[3 Metode 2](#_Toc158625718)

[3.1 Utviklingsprosess 2](#_Toc158625719)

[3.2 Verktøy 3](#_Toc158625720)

[3.3 Bruk av KI verktøy 3](#_Toc158625721)

[4 Resultat 3](#_Toc158625722)

[4.1 Teknisk Design 3](#_Toc158625723)

[4.2 Implementasjon 3](#_Toc158625724)

[4.3 Testing 4](#_Toc158625725)

[4.4 Utrulling til sluttbruker (deployment) 4](#_Toc158625726)

[5 Drøfting 4](#_Toc158625727)

[5.1 Drøfting av løsning/design 4](#_Toc158625728)

[5.2 Drøfting av prosess 5](#_Toc158625729)

[5.3 Drøfting av bruken av KI-verktøy 5](#_Toc158625730)

[6 Konklusjon - erfaring 5](#_Toc158625731)

Figurliste

[Figur 1 Use Case diagram 3](#_Toc126231951)

[Figur 2 Klassediagram som viser... 4](#_Toc126231952)

Tabelliste

[Tabell 1 Begreper og ordliste 2](#_Toc126231737)

[Denne rapporten inneholder ferdigdefinerte **stiler** som du/dere kan benytte for de mest vanlige avsnittene. Følgende stiler er definert:

Heading 1/Overskrift 1 Overskrift på nivå 1

Heading 2/Overskrift 2 Overskrift på nivå 2

Heading 3/Overskrift 3 Overskrift på nivå 3

Brødtekst Standard tekst i et avsnitt. Benytt denne for all ”vanlig” tekst

Definition Benyttes hovedsakelig i avsnittet ”TERMINOLOGI”

References Benyttes i REFERANSER-avsnittet.

Comment Denne grønne teksten. Fjern all tekst av denne typen i rapporten.]

# Introduksjon

## Bakgrunn

[Dette er første kapitlet i den faglige rapporten. Det bør behandle bakgrunnen for oppgaven, eventuell oppdragsgiver, oppsummering av problemstillingen og/eller oppgaven som skal løses.]

Denne rapporten er skrevet ved NTNU Trondheim som en del av emnet IDATT2003, Programmering 2 V25. Hensikten med prosjektet er å vise objektorientert programmering (OOP), programvareutvikling og prosjektarbeid gjennom utvikling av brettspill med par-programmmering.

## Kravspesifikasjon

Bruk her gjerne **UML-diagrammer** som **Use-Case**, **Aktivitetsdiagram** osv for å beskrive krav til funksjonalitet (NB! Uten å dra inn **hvordan** du/dere har løst det.).

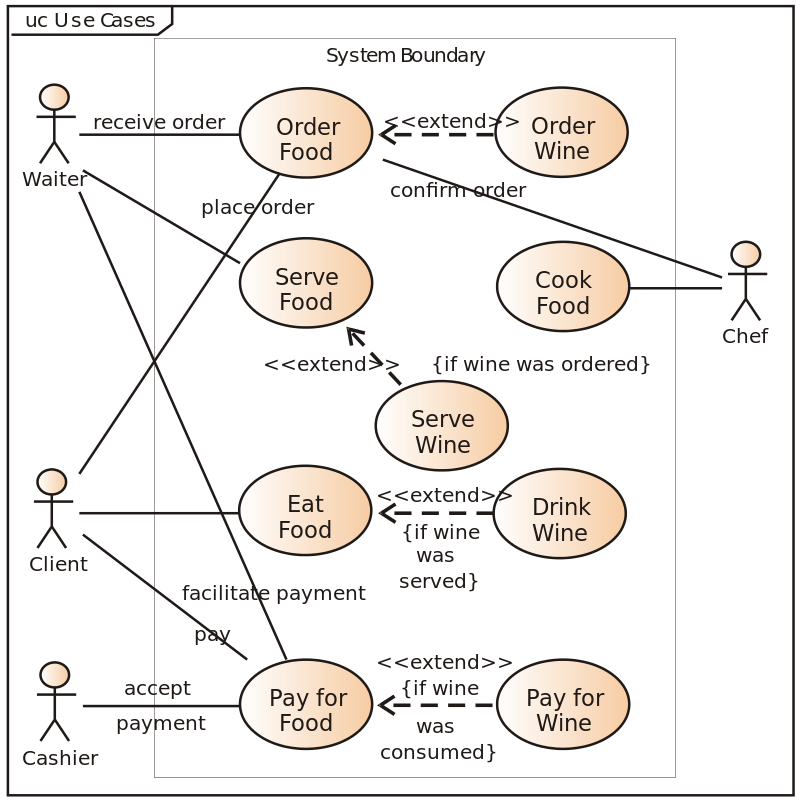
Når du senere skriver **drøfting** og **konklusjon**, må du henvise tilbake til dette kapittelet og svare på om løsningen du har levert svarer på kravspesifikasjonen].

Følgende **funksjonelle** krav gjelder:

* En bruker skal kunne velge et spill blant et på forhånd definerte varianter av stigespill (f.eks. Ved å lese fra fil). Med “varianter” menes f.eks. layout (antall felt) og hvor stigene er plassert.
* Bruker velger så hvor mange spillere som skal delta i spillet (**maks fem**).
* Hver spiller velger deretter én “brikke” for sin spiller (fra et utvalg av ulike forhåndsdefinerte brikker) og gir sin brikke/spiller et navn.
* Spillet starter med at det trilles **to terninger** for hver spiller, øynene summeres opp og spillerens brikke flyttes tilsvarende.
* Avhengig av hvilken type felt spilleren lander på, utføres respektiv handling:
  + Stige opp
  + Stige ned
  + Rykk tilbake til start
  + Stå over ett kast
* Når første spiller når siste rute, rute 90, er spillet over og vinneren kåres.

Følgende **ikke-funksjonelle** krav gjelder:

* Det skal benyttes verktøy som **CheckStyle** (med Google sine regler) og **SonarLint** for å sikre høy kodekvalitet.
* Det skal utføres både **positive** og **negative enhetstester** for den delen av koden som er forretnings-kritisk
* Det skal implementeres solid **unntakshåndtering** med formål om å gjøre koden mer robust.
* Prosjektet skal legges under versjonskontroll.
* Følgende versjon av Java JDK, biblioteker og Maven Plugins skal benyttes:
  + Java: versjon 21
  + Bibliotek:
* JUnit: org.junit.jupiter:junit-jupiter-engine:5.11.4
* JavaFX: org.openjfx:javafx-controls:23.0.1
* MavenPlugins:
* org.apache.maven.plugins:maven-compiler-plugin:3.13.0
* org.apache.maven.plugins:maven-surefire-plugin:3.2.5
* org.openjfx:javafx-maven-plugin:0.0.8



Figur 1 Use Case diagram

## Avgrensninger

For begge nivåer gjelder følgende avgrensninger av oppgaven:

* Spillet skal spilles av alle spillere på samme datamaskin, det vil si ingen nettverksbasert spill.
* Spillet skal ha et **grafisk brukergrensesnitt** utviklet med biblioteket **JavaFX** uten bruk av FXML

## Begreper/Ordliste

[Når man utvikler programvare for en kunde, er det viktig å etablere en felles forståelse for begreper/terminologi/ord som benyttes **av kunden**. Det er derfor svært vanlig å lage en "ordliste" og/eller en "Domene modell". Denne ordlisten er også et svært godt utgangspunkt for å finne frem til hvilke mulige **klasser** det kan være aktuelt å implementere i løsningen når denne skal utvikles. Bruk tid på denne slik at du har en god forståelse for begrepene.

Her kan man også bruke klasse-diagrammer for å illustrere hvordan **begreper** henger sammen. NB! Klasser i dette diagrammet er **ikke** klassene du har valgt å implementere i prosjektet.]

| Begrep (Norsk) | Begrep (Engelsk) | Beskrivelse |
| --- | --- | --- |
| Spill | Game | Ansvarlig for å sette opp selve spillet: sette opp brettet, sette opp terninger, registrere spillerne, osv. |
| Terning | Die | Representerer én enkelt terning. |
| Terninger | Dice | Representerer samlingen av alle terninger. |
| Spillbrett | Board | Representerer selve spillbrettet bestående av en samling av felt. |
| Spiller | Player | Representerer én spiller i spillet. |
| Felt | Tile | Representerer ett felt på spillbrettet. |
| Aksjon | Action | Representerer ulike typer aksjoner som skal kunne utføres på en spiller som lander på ett gitt felt. |

Tabell 1 Begreper og ordliste

# Teori (påbegynt)

[Her beskriver du de **teoriene** og **beste praksiser** som dere har benyttet ved utvikling av en programvare, som kobling, cohesion, SOLID osv. Beskriv **hvilke teorier** og IKKE **hvordan** dere har brukt teoriene. Det hører hjemme i resultat og diskusjon.

Her må dere referere til **kilder** (som legges til i kapittelet **Referanser**).

Når du senere skal skrive drøfting og konklusjon, må du henvise tilbake til dette kapittelet og svare på om løsningen du har levert er løst i henhold til disse teoriene/beste praksisene.

Unngå å beskrive om grunnleggende teorier som «hva er OOP», «Programmeringsspråket Java» osv.

Skriv **kun teoripunkter som du faktisk har benyttet i ditt design, og som du faktisk diskuterer i diskusjonen** 😉]

**Coupling** refererer til den direkte avhengigheten mellom klasser, altså hvordan endringer i klasse A kan påvirke klasse B (Jain, 2024). Det finnes to typer kobling: “tight coupling” og “loose coupling”. Tight coupling innebærer at endringer i klasse A krever tilsvarende endringer i klasse B. loose coupling, derimot, betyr at klassene stort sett er uavhengige av hverandre slik at endringer i én klasse ikke påvirker den andre (Horstmann, 2021, s. 203)

**Cohesion** er et prinsipp som handler om å designe klasser slik at de har et enkelt og klart formål. Jo mer fokusert og målrettet en klasse er, desto høyere kohesjon har den. Klasser med høy kohesjon er enklere å vedlikeholde og trenger sjeldent endirnger, sammenlignet med klasser med lav kohesjon. Dessuten er de ofte mer gjenbrukbare enn andre klasser (Jain, 2022)

**Robust kode** går ut på å utvikle et program som forutser potensielle problemer og håndterer dem effektivt (Bishop & Elliot, 2013)

**Factory design mønster** er et kreativt designmønster som definerer et grensesnitt for å lage et objekt, men lar subklasser implementere sine egne fabrikkmetoder for å lage spesifikke objekttyper. Dette mønsteret fremmer “loose coupling” mellom objektskaperen og objektene selv. (geeksforgeeks, 2024)

**Observer design mønster** er et atferdsdesignmønster som definerer en én-til-mange-avhengigheter mellom ulike objekter. Det går ut på at når ett objekt (subjektet) endrer tilstand, blir alle dets pårørende (observatørene) varslet og automatisk oppdatert. (geeksforgeeks, 2025)

# Metode

[I dette kapittelet skal dere beskrive hva som skal til for å kunne reprodusere resultatet dere har fått. I programvareutvikling koker det som oftest ned til **prosess/metodikk, og verktøy**.]

## Utviklingsprosess (påbegynt)

[I dette kapittelet skal du fortelle hvilken prosess du/dere planla å følge. Den skal dekke prosessmodellen, hvorfor den ble valgt og hvordan den ble implementert.

Har dere jobbet i gruppe, så si noe om hvordan dere planla å organisere arbeidet (hvor ofte planla dere å møtes å jobbe?)

Hvordan planlagt prosess faktisk fungerte og hvilke endringer dere eventuelt gjorde skal beskrives i resultat-kapittelet og drøftes under Drøfting.

I en typisk mappe-oppgave der dere har jobbet gjennom flere stadier/deler av prosjektet, og fått tilbud om tilbakemelding, bør dette beskrives som en del av prosessen (altså **at** dere jobbet i f.eks. 3 deler med muntlig tilbakemelding etter hver del).]

Prosessmodellen som ble planlagt å følge var en iterativ prosessmodell, da denne til enhver tid tillater fleksibilitet og tilpasning gjennom prosjektets ulike deler. Arbeidet ble delt inn i flere ulike issues på GitHub, hvor vi valgte arbeidsoppgaver ut ifra hva vi følte vi mestret best. For hver fullførte issue sørget vi for å gjennomgå, diskutere og teste ut de nye endringene i felleskap. Gruppen planlagte å møtes ukentlig for å jobbe sammen og diskutere arbeidsoppgavene.

Brukertesting startet etter hvert som vi hadde utviklet en brukbar og fungerende MVP. Med tilbakemeldingene fra brukertestingen ble koden tilpasset, før den så ble testet igjen.

## Verktøy

|  |  |  |
| --- | --- | --- |
| **Navn** | **Versjon** | **Benyttet til** |
| IntelliJ IDEA | 2024.2.4 | Utviklingsverktøy |
| GitHub |  | Versjonskontroll og prosjektplanlegging |
| Messenger |  | Kommunikasjon |

# Resultat

## Teknisk Design

[Kapittelet om teknisk design beskriver det store bildet av valgt løsning. For et programvareutviklingsprosjekt vil dette vanligvis inneholde systemarkitekturen (klient-server, sky, databaser, tjenester, desktop-applikasjon osv.); både hvordan det ble løst, og, enda viktigere, hvorfor denne arkitektur ble valgt]

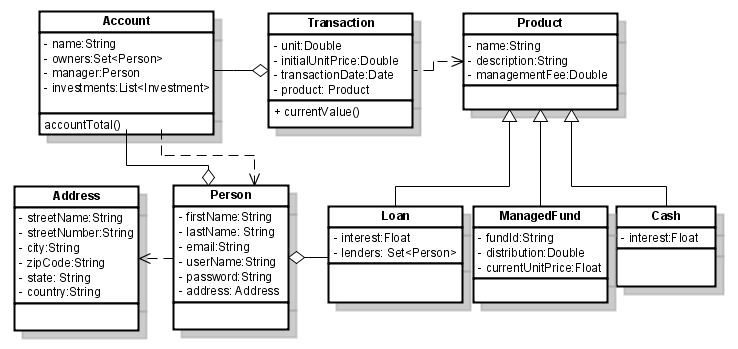
## Implementasjon

[Her beskriver du de tekniske detaljene til løsningen du har endt opp med. Hvilke eksterne biblioteker og rammeverk, APIer osv. har du/dere benyttet?

Vis med **klassediagrammer** hvordan klassene i løsningen din henger sammen. Husk at du kan vise flere klassediagrammer for å vise ulike sider ved løsningen (kun klasser som benyttes til brukerinteraksjon f.eks., eller kun klasser som utgjør forretningslogikken).

Bruk **aktivitetsdiagramm(er)** for å beskrive logikken/flyten i løsningen din. Fint også om du kan vise hvordan objekter av klassene dine **samhandler** for å løse **de mest sentrale** oppgavene i løsningen, da i form av **sekvensdiagramm(er)**.

Klassene dine bør beskrives i forhold til hvilken **rolle** og **ansvar** de har, men det er ikke nødvendig å beskrive hver enkelt metode eller felt til klassen. De står uansett dokumentert i Javadoc’en din.]



Figur 2 Klassediagram som viser...

## Testing

[Beskriv hvordan løsningen din har blitt testet, både i form av enhetstesting, men også **brukertesting**. Hvor mange brukere har testet løsningen? Hvordan ble brukertesting gjennomført? Vis også eventuelt resultatene fra brukertestingen her. (svar på spørreskjema el.l.)

Enhetstesting: si noe om hvilke klasser du/dere har valgt å skrive tester for, og hva **strategien** som dere/du har lagt opp til for å sikre best mulig testet kode.]

## Utrulling til sluttbruker (deployment)

[Her beskriver du hvordan programvaren din gjøres tilgjengelig for sluttbruker. Rulles den ut å en web-server? I så fall hvordan? Lages det en desktop-applikasjon som bruker kan dobbeltklikke på for å starte? Eller kjøres applikasjonen fra Maven (mvn javafx:run) Osv.]

# Drøfting

## Drøfting av løsning/design

[Her oppsummerer du/dere oppgaven. Hvor langt kom du/dere (resultat)? Hva fikk du/dere ikke gjort i forhold til oppgaveteksten? Hva var de store utfordringene/problemene du/dere møtte, etc..

Spesielt viktig er det å **drøfte din egen løsning i forhold til det du har lært om gode prinsipper for design av programvare** (robust kode, kodestil, designprinsipper osv) som beskrevet i teori-kapittelet. I en godt skrevet rapport, er det ingen teorier som beskrives under teori-kapittelet som ikke drøftes under drøfting-kapittelet.

Husk å være **konkret**: Det holder ikke å skrive «Jeg/vi har designet en løsning som er i trå med prinsippene om coupling og cohesion». Du/dere må «**bevise**» **hvorfor** dere kan påstå dette. Altså hente eksempler fra egen kode som underbygger teoriene om god design: «I klassen ..... har vi valgt å returnere .... fra metoden.... Dette bidrar til lav kobling....»

Her bør man også gjøre seg tanker rundt kvaliteten av det arbeidet som er nedlagt.

Er de kildene du/dere bruker pålitelige, er det sprik mellom forskjellige kilder (og i så fall hvorfor), er det andre forhold som kan være med å gjøre noen av de vurderinger og valg du/dere har gjort usikre?]

## Drøfting av prosess

[Fulgte dere prosessen som dere planla (og beskrev under kapittelet «Metode»)? Var det lurt, eller ikke? Hva fungerte bra hva fungerte mindre bra? Hva ville du/dere ha gjort annerledes neste gang?]

## Drøfting av bruken av KI-verktøy

[Dersom du/dere benyttet KI-verktøy i denne oppgaven, drøft kort erfaringene dine/deres. Hva var KI-verktøyene nyttige for å løse? Hvilke svakheter oppdaget du/dere?

Har du/dere ikke benyttet KI-verktøy dropper du dette kapittelet.]

# Konklusjon - erfaring

[Overbevisninger /erfaring som en er kommet fram til på grunnlag av det presenterte materialet.

* Fikk du realisert hele problemstillingen fra kapittel «‎1.2 Kravspesifikasjon»?
* Hva ville du ha gjort annerledes dersom du kunne begynn på nytt?
* Hva slags begrensninger kan en forvente når en bruker løsningen?
* Hva skal tas opp i fremtidige arbeid dersom du eller noen andre ville ha tatt utvikling videre?]

Referanser

[Forfatter, årstall, tittel på bok eller artikkel, navn på tidsskrift eller forlag/utgiver, nr. eller dato for tidsskrift, sted som det vises til eller refereres fra i oppgaven.

Konkret for programmeringsemner: Regner med at du/dere kommer til å måtte slå opp litt i læreboka, så den er en innlysende referanse. Dersom du/dere i tillegg benytter internett, så list URL’er til sidene du/dere har benyttet.

Her er en god guide til hvordan oppgi referanser og hvordan referere til de (benyttes av IEEE): <https://www.bath.ac.uk/publications/library-guides-to-citing-referencing/attachments/ieee-style-guide.pdf> ]

1. ”Objects First With Java”, Sixth edition, av Barnes og Kölling. ISBN ….
2. http://.....
3. Osv.

VEDLEGG

[Materiell som er utarbeidet eller innsamlet i tilknytning til rapporten, men som det ikke er naturlig eller hensiktsmessig å ta inn i hoveddelen, som feks brukerveiledning, skal tas inn som vedlegg.

Vedleggene skal være nummererte og ha en overskrift.

Har du/dere ingen vedlegg, så droppes dette kapittelet.]