UNIVERSITETET I BERGEN

KANDIDAT

153

PRØVE

INF112 0 Innføring i systemutvikling

Emnekode	INF112
Vurderingsform	Skriftlig eksamen
Starttid	11.06.2025 07:00
Sluttid	11.06.2025 10:00
Sensurfrist	--
PDF opprettet	18.08.2025 08:58

Erfaringer

Oppgave	Tittel	Status	Poeng	Oppgavetype
ℹ	Generell info om eksamen			Informasjon eller ressurser
1.1	Bakgrunn [5 poeng]	Besvart	5/5	Langsvar
1.2	Teamarbeid [5 poeng]	Besvart	4.5/5	Langsvar
1.3	Fremtid [5 poeng]	Besvart	3/5	Langsvar

Verktøy

Oppgave	Tittel	Status	Poeng	Oppgavetype
2.1	Git arbeidsflyt [6 poeng]	Besvart	4.5/6	Langsvar
2.2	IDE [4 poeng]	Besvart	4/4	Langsvar
2.3	CI/CD [3 poeng]	Besvart	3/3	Langsvar
2.4	Statisk analyse [2 poeng]	Besvart	2/2	Langsvar

API-design

Oppgave	Tittel	Status	Poeng	Oppgavetype
ℹ	Introduksjon – geitesimulator			Informasjon eller ressurser
3.1	Valg av abstraksjoner [5 poeng]	Besvart	4/5	Langsvar
3.2	Å mutere eller ikke mutere? [2 poeng]	Besvart	2/2	Langsvar
3.3	Java interface [6 poeng]	Besvart	4/6	Programmering
3.4	Gammel kode [3 poeng]	Besvart	3/3	Langsvar
3.5	Renovasjon [4 poeng]	Besvart	3/4	Programmering

3.6	Refleksjon [3 poeng]	Besvart	3/3	Langsvar

Brukeropplevelse og universell utforming

Oppgave	Tittel	Status	Poeng	Oppgavetype
4.1	Brukerhistorier [3 poeng]	Besvart	2/3	Langsvar
4.2	Universell utforming [4 poeng]	Besvart	4/4	Langsvar

Mappevurdering

Oppgave	Tittel	Status	Poeng	Oppgavetype
5.1	Poeng fra semesterprosjektet [40 poeng]	Ubesvart	39.6/40	Muntlig

1.1 Bakgrunn [5 poeng]

Det kan være krevende å sette seg inn i et nytt prosjekt. Ofte må du lære mange verktøy, biblioteker og teknologier. I tillegg må du kanskje jobbe på nye eller uvante måter.

- **Tenk gjennom erfaringene dine, og nevn to ting…**
  - *a)* …som du føler du eller teamet ditt hadde hatt nytte av å kunne bedre før dere begynte på prosjektet,
  - *b)*. …eller som du trodde ville være viktige på starten av semesteret, men som viste seg ikke å være så nødvendige.
- Du kan velge to ting fra *a* eller *b*, eller gi ett eksempel fra hver. Husk å begrunne svaret.

(Ca. et avsnitt på hver ting)

Skriv ditt svar her

En ting vi hadde hatt nytte av å kunne bedre før vi begynte var Git. Vi hadde en bratt læringskurve på branching og rebasing. Noen på teamet hadde god kontroll på dette fra egne prosjekter, men majoriteten hadde kun erfaringer fra INF101 og INF102 hvor man kun bruker add, commit og push til et repo du styrer alene. Vi løste dette tidlig med å ha en felles gjennomgang av git commands og andre relevante verktøy som lazygit. Vi lagde også rettningslinjer på hvordan commit meldingene skulle være med forskjellige tags og branch navngivning.

Vi trodde på starten at det var viktig at alle hadde komplett og konstant innsikt på alt de andre gjorde, dette viste seg å ikke stemme da de ukentlige møtene (scrum-stil) ga alle kontroll på prosjektet som en helhet og lot oss fokusere på våre egne oppgaver mens Scrum-master passet på at alt gikk som det skulle. Vi endte også opp med et Trello-board som hjalp oss se hvor langt vi hadde komt i den gjeldene sprinten. Dette førte til at vi slapp å ha konstante meldinger med spørsmål til Scrum-master og gjorde oss mer effektive.

Ord: 189

Maks poeng: 5

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

6 1 2 3 4 0 9

1.2 Teamarbeid [5 poeng]

Tenk gjennom erfaringene dine fra vårens prosjektarbeid.

- **Hva synes du er de to viktigste tingene du har lært om teamarbeid i løpet av arbeidet med prosjektet? Forklar.**

(Ca. et avsnitt på hver ting)

Skriv ditt svar her

Det viktigste jeg lærte var hvordan Scrum virket i praksis. En agil arbeidsmetodikk med et team var noe jeg aldri hadde erfart før, men hadde hørt veldig mye om i jobbintervjuer. Det ble en veldig fin erfaring å se hvordan et team kan bli så samkjørte og alle etterhvert ble selv drevene når vi fulgte sprintene, standup og rollene man vi ble enige om. Dette får jeg bruk for i min gjeldene jobb nå!

Den andre tingen jeg lærte var viktig, var alt alle har forskjellige timeplaner og ikke alle kan ha like mye fokus og belasting hver uke. Å ha et møtereferat/log fra møtene mellom sprintene som gjør at man alltid kan gå tilbake å lese om man har glemt noe. Grunnen til dette ble viktig var at når det kom innleveringer i andre fag så ble det plutselig flere dager mellom møte og når man skulle jobbe med prosjektet. Vi på teamet går forskjellige linjer så vi hadde veldig varierande timeplaner. Da å kunne gå tilbake og lese gjennom møte gjorde at man raskt kunne starte igjen med sine oppgaver. Det eliminerte mye unødvendig kommunikasjon og meldinger som oppsto i starten da møtereferatene ikke var gjennomført like nøye.

Ord: 200

Maks poeng: 5

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

3051645

1.3 Fremtid [5 poeng]

Nevn to ting du synes du bør lære mer om for å bli en god programvareutvikler, og forklar hvorfor de er viktige.
(Ca. et avsnitt på hver ting)

Skriv ditt svar her

Den første viktige tingen er hvordan man fort kan sette seg inn i andres kode base med å lese diagrammer. Som deltids-ansatt som utvikler brukte jeg mye tid i starten av hvert prosjekt bytte, på å bare sette meg inn i hvor forskjellig logikk var, og hvordan prosjektet var strukturert. Å klare å lese klasse diagrammer, flyt modeller og fort absorbere annen dokumentasjon, gjør at man raskere kan starte å produsere kode.

Den andre viktige tingen er hvordan man skriver kode etter retningslinjer og regler satt av noen andre/arbeidsgiver. De fleste har sine måte å gi navn til klasser og metoder, hvordan modulere koden, pakke/mappe struktur. Men når man kommer i jobb og skal inn på et prosjekt hvor det jobber flere titalls andre så er det viktig at man følger generelle retningslinjer og vet hvilke deler/steder av koden som mest sannsynlig har retningslinjer man må følge. Dette gjør det mye lettere for resten av teamet og forstå koden du har skrevet.

Ord: 162

Maks poeng: 5

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

4 2 7 2 4 1 5

2.1 Git arbeidsflyt [6 poeng]

Se for deg at du og noen andre studenter skal begynne på et nytt programmeringsprosjekt (f.eks. som del av INF218/219). Dere er et lite team på 4–6 utviklere. Gruppen skal kode i Java. Du er valgt som teamets Git-ekspert

a. **Hvordan ville du lagt opp Git arbeidsflyten slik at dere kan samarbeide best mulig, både med vedlikehold og videreutvikling?** F.eks. med tanke på branch-struktur, navngiving og bruk av branches, om man skal pushe direkte til `main` eller bruke merge requests osv.
b. Et av de andre teammedlemmene kan ikke så mye om Git, og spør deg om hjelp med en merge konflikt. **Forklar kort hvordan hen bør gå frem for å løse merge konflikten.**

(2–3 avsnitt)

Skriv ditt svar her

a) Jeg som Git-ekspert ville laget noen retningslinjer som alle på teamet måtte fulgt og som hadde gjort det ryddig og oversiklig for alle: vi hadde brukt tags som: feat, realease, hotfix, cleanup, refactor.

1. Vi har to Main branch: main som er en stabil og produksjonsklar branch og develop som er en integrasjons branch.
2. Ingen skal jobbe direkte på main.
3. Alt skal først pushes til deveop for integrasjons testing.
4. alle skal jobbe på en egen branch.
5. hver branch skal bare dekke en oppgave
6. branch skal navngis etter: <tag>/<navn>
7. alle skal lage en merge request til develop først som må ble sett over av en annen utvikler.

b)

Merge konflikten skjer når det er gjort endringer (ofte noen andre har merget) på et sted du har endret/lagt inn/slettet kode fra når du gjorde din forrige rebase. For å gå fram for å løse en merge conflict bruker vi først bare git rebase <branch-navn>. Om endringene har en større konflikt å bruker vi git rebase -i HEAD<n (head number/id)> for å få en interaktiv rebase. Da kan vi manuelt se over konflikt og velge hvilke versjon vi vil bruke. Når dette

er gjort skal du kunne pushe og lage en PR. Siste løsning er å manuelt gå gjennom konflikten og endre din kode, for å så bruke git add . og commit.

Ord: 227

Maks poeng: 6

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

8 5 2 5 0 1 3

2.2 IDE [4 poeng]

Java-kode er vanlig tekst og kan skrives i en vanlig teksteditor. Likevel velger de fleste å bruke et integrert utviklingsmiljø *(Integrated development environment – IDE)*, spesielt i større prosjekter. IDE-er har mange nyttige hjelpemidler, for eksempel autofullføring, navigasjon, dokumentasjonsvisning og *mye* annet. De gjør det også enklere å holde orden på prosjektet. Med støtte fra en IDE blir det enklere å skrive korrekt kode, finne frem i ukjent kode og gjøre endringer eller vedlikehold. Eksempler på vanlige IDE-er for Java er Eclipse, IntelliJ IDEA, NetBeans og VSCode.

a. **Hvilken IDE (eller editor) brukte du vanligvis til å redigere koden i semesterprosjektet? Brukte alle på teamet samme IDE?**
b. **Hva tenker du er de nyttigste tingene å se etter når du skal velge IDE, slik at du får best mulig hjelp og støtte i utviklingsarbeidet?**

(1–2 avsnitt)

Skriv ditt svar her

a) Vi på hele teamet brukte IntelliJ som hovededitor. Jeg brukte også neovim når jeg skulle skrive masse "boilerplate" kode på en gang, uten å trenge å kjøre programmet. At hele teamet brukte IntelliJ gjorde formatting veldig lett, da vi alle kunne de samme plugins og instillinger.

b) Det nyttigste for meg er hvor mye IDE kan gjøre av oppsettet for deg, og maven build i dette prosjektet. At den har gode plugins, vim support, formatering, LSP-support for språket. At den er lett å navigere uten å manuelt klikke gjennom mappene. Det er også viktig at den har mange analyse-verktøy for testing og effektivitet. Dette gjør det lett å se hvor forbedringspotensialet ligger.

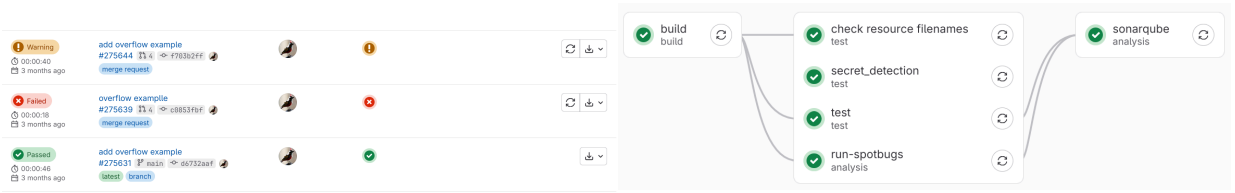
Ord: 113

Maks poeng: 4

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

9 8 2 3 5 2 7

2.3 CI/CD [3 poeng]



Under vårens prosjektarbeid var GitLab satt opp med kontinuerlig integrasjonstesting (*CI/CD – Continuous Integration / Deployment* – vi brukte bare CI), slik at hver gang du pushet endringer til `git.app.uib.no` ble det startet en såkalt *pipeline* på serveren, som sjekket ut den nyeste koden, kompilerte den og kjørte JUnit-tester og evt. andre tester. Resultatet av testkjøringen ble deretter gjort tilgjengelig på prosjektsiden.

a. Når du utvikler kjører du vanligvis testene selv på din egen maskin før du committer og pusher. **Hvorfor kan det likevel være nyttig å også ha kontinuerlig integrasjonstesting når du pusher til serveren? Forklar.**
b. **Opplevde du i løpet av semesteret at CI-serveren fant feil / problemer du ikke var klar over før du pushet?**

(1–2 avsnitt)

Skriv ditt svar her

a) Det er ikke alltid din pc/mac har samme konfigurasjon og env som serveren. At alle må kjøre koden på serveren gjør at man er sikker på at koden du pushet kan kjøre uten feil på serveren. Pipleline feil gjør også at andre på teamet kan se hva som ble feil, det gjør det også lett å "roll-backe" koden om en feil skulle oppstå. Det gjør også at man ser feil med fil-stier til ressurser om man har skrevet det feil, da serveren ikke har samme mappe struktur som deg.

b) Nei, den fant aldri uventede feil på mine push. Vi hadde en på gruppen som hadde komplett filsti til en lydfil i koden sin og ikke sti fra src folder. Så der fanget pipeline en feil som ikke ble sett at utvikleren.

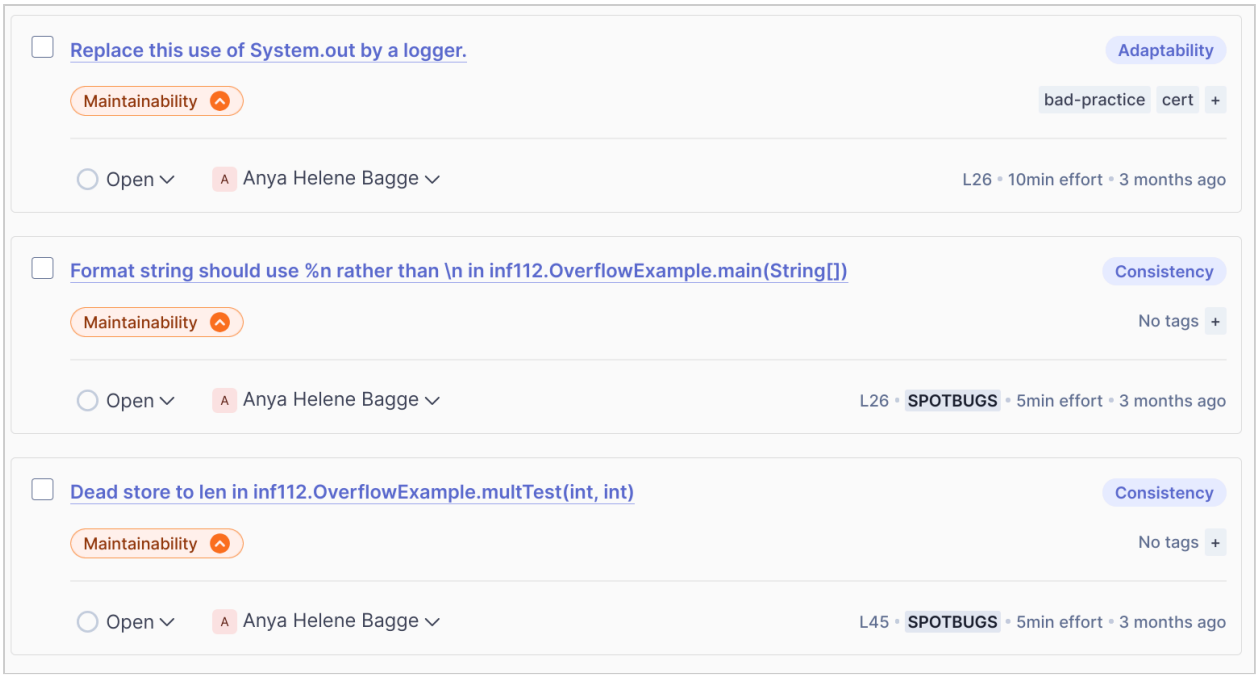
Ord: 133

Maks poeng: 3

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

7 9 0 6 3 0 9

2.4 Statisk analyse [2 poeng]



I løpet av semesteret satte vi også opp noen *statiske analyseverktøy* – *SpotBugs* og *SonarQube*.

Slike verktøy analyserer koden – dvs. teksten og de kompilerte klassefilene – uten å kjøre den. Målet er å finne feil og potensielle problemer – samt ting som ikke egentlig er feil men kan være dårlig praksis eller føre til vedlikeholdsproblemer. SonarQube presenterer funnene på en nettside, sammen med litt statistikk (blant annet test coverage).

Eksempler på ting SpotBugs og SonarQube sjekker etter er: metoder som er for kompliserte, literaler som burde defineres som konstanter, feltnavn som kan misforstås, definisjoner som burde vært `private`, `protected` eller `final` osv. Java-kompilatoren gjør også litt statisk analyse – blant sjekker den for typefeil, variabler som ikke blir brukt osv.

a. **Tittet du på SonarQube-siden i løpet av semesteret? I så fall, var noen av tilbakemeldingene nyttige?**
   (SonarQube-oppsettet var litt eksperimentelt, så det er helt rimelig om du ikke sjekket det.)

Du la antakelig mye arbeid i å skrive JUnit-tester i løpet av semesteret. Men trenger vi egentlig å skrive enhetstester når vi har fancy analyseverktøy som SonarQube – eller trenger vi SonarQube når vi har tester?

b. **Gi et eksempel på feil/problemer som statisk analyse kan oppdage, men som er vanskelig å oppdage med enhetstester.**
c. **Gi et eksempel på feil/problemer som er greit å oppdage med tester, men som vil være vanskelig for et statisk analyseverktøy å oppdage.**

(Noen setninger om hvert punkt.)

Skriv ditt svar her

a) Jeg var inne å så på sonarqube for å rydde opp i koden før innlevering, den viste meg alt av små "errors" som ubrukte imports, typos og andre små

forbedringer som ga oss høyere kodekvalitet.

b)

- Ubrukte imports
- Typos
- kodeduplikat
- innkapsulering, at man marker det som ikke blir endret med final, osv.

c) statisk analyse kan ikke oppdage:

- kjøre feil
- logikk feil, at funksjonen gjør det den faktisk skal
- sideeffekter, at funksjonen ikke endrer på ting den ikke skal endre.

Ord: 88

Maks poeng: 2

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

2 3 4 8 2 4 8

3.1 Valg av abstraksjoner [5 poeng]

En viktig designavgjørelse er hvilke ting som skal være egne abstraksjoner (egne `interface`s), og hva som kan representeres direkte med primitive typer og klasser fra Java-biblioteket eller andre vanlige biblioteker. Førstnevnte kan gi et API som er bedre tilpasset bruken, mens sistnevnte har fordelen at folk gjerne kjenner det fra før (og det blir mindre jobb for deg). For eksempel:

- Vi kan lage en egen type `Map` for kartet, eller vi kan bruke Javas `MapSquare[][]` eller en generell grid-datastruktur `Grid<MapSquare>`.
- Posisjoner kan enkelt representeres som tall, *x* og *y* – men kanskje det er lurt å ha metoder for å regne ut retninger og avstander? Burde vi i så fall lage en egen `Position`, eller heller bruke f.eks. LibGDX sin `Vector2`?
- Retninger kan være tall (i grader) eller vektorer – men det er også mulig det er nyttig med en egen `Direction` abstraksjon
- **Hvilke designvalg ville du gjort for *kart*, *posisjon* og *retning*?** Husk å begrunne svaret.

(1–3 avsnitt)

Skriv ditt svar her

Jeg tenker siden det er biologer som skal bruke API, som mest sansynelig ikke har mye koderfering, er det best å lage en egen type Map for kartet, jeg vil også ha en MapCell record som holder (Position, T thing) så posisjon og en ting som er på kartet. Vi brude også bruke en egen Position, med felt variabler som gir mening for biologer som kordinater til geiten. Vector2 klassen kan bli brukt innad i Position til å regne ut kordinater hvis vi vet vilke kordinater som Map spanner. Det viktigste er at når biolognen skal ta over så har de mulighet til å forstå hva ting betyr og gjør. Retningen bør også ha abstraksjon i form av et Direction Interface. Dette kan ha metoder som getDirection (som kan retunere circa retning i form av nord, sør, øst, vest) og en metode som getAngleDircetion som gir nøyaktig retning i grader. Totalt sett blir dette lettere for en biolog å forstå da de slipper å lese seg opp på hele Vector2, MapSquare, osv og kun blir presentert med relevante og selvforklarende metoder.

Ord: 181

Maks poeng: 5

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

6 5 1 5 4 1 7

3.2 Å mutere eller ikke mutere? [2 poeng]

Et annet valg er om vi skal kunne oppdatere objektene (de er muterbare, *mutable*), eller om hver utregning gir et nytt objekt (de er ikke-muterbare, *immutable*, og kan ikke endres etter at de er opprettet). For eksempel er LibGDX sin `Vector2` *mutable*, mens Java sin `String` er *immutable*.

- **Hvorfor kan det være nyttig å bruke ikke-muterbare objekter?** Forklar kort.
- **Hva ville du valgt for dine abstraksjoner?** Forklar kort.

(Ca. 1 avsnitt)

Skriv ditt svar her

Om et objekt er ikke-muterbart så kan man unngå mange feil og bugs. At man eksplisitt på reassign et objekt for å endre på det gjør koden mer leselig og forståelig. Dette er spesielt relevant om man skal gi en verdi til en funksjon som da endrer på objektet når dette ikke var tanken. Med mine abstraksjoner ville jeg holdt objektene ikke-muterbare så biologene blir tvunget til å eksplisitt spesifisere at de vil endre med å reassign. Det jeg da må passe på et at metode-navnene må vise dette. Så istedetfor position.move(delta), så ville jeg brukt position.movedBy(delta) for å gjøre det mer forståelig.

Ord: 103

Maks poeng: 2

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

4 7 6 9 7 4 5

3.3 Java interface [6 poeng]

Skriv Java-kode (kun `interface`) for eventuelle abstraksjoner (*kart*, *posisjon* og *retning*) du valgte å ha med i designet i de første deloppgavene.

- Ta med metoder du mener er nødvendige, og metoder du tenker vil være nyttige for de som implementerer geite-oppførselen.
- Skriv *kort* JavaDoc-dokumentasjon – en linje eller to om hva metodene gjør. Du trenger ikke ha med full `@param` beskrivelse av parameterne, men ta med eventuell informasjon brukeren må være oppmerksom på – spesielle krav til parametre, at en metode kan returnere `null`, om metoder må kalles i en spesiell rekkefølge osv.

Skriv ditt svar her

```
1  public record mapCell(Position pos, T thing)
2  public enum Dir {
3      north,
4      south,
5      east,
6      west
7  }
8  public recourd Step(int x)
9
10 public interface Map {
11     //T is an enum representing possible "tings" that can be placed on the map
12
13     //return a list of what is on that mapSqure,
14     mapCell get(Position pos)
15
16     //changes that mapCell to hold t
17     set(T t, Position pos)
18
19     //returns all mapCells
20     mapCell[] getAll()
21 }
22
23 public interface Position {
24     //returns a new position moved to pos (nice if Position holds other values)
25     Position posMovedTo(Position pos)
26
27     //returns a new postion shifted by x animalsteps in direction of animal
28     Position posShiftedBy(Step x)
29
30
31 }
32
33 public interface Direction {
34     //returns direction in terms of north, west, south, east
35     Dir getDirection()
36
37     //returns degree of direction
38     Degree getDegree()
39
40 }
```

Maks poeng: 6

Knytte håndtegninger til denne oppgaven? **6 1 5 6 8 6 2**

Bruk følgende kode:

3.4 Gammel kode [3 poeng]

En annen gruppe studenter laget en prototype av geitesimuleringen i fjor. De hadde ikke tatt INF112, og kunne betydelig mindre om programutvikling enn deg. I kodeeksempelet (se vedlagt PDF) kan du se deres implementasjon av `Goat::findFood()`, som skal lete etter mat i nærheten og snu geiten i riktig retning om den finner mat. Studer koden – du trenger ikke bruke tid på alle detaljene i hvordan den virker, det holder å forstå hva den gjør.

- Tenk gjennom designprinsippene (SOLID, osv.) og det du har lært om kodekvalitet og vedlikeholdbarhet. **Hva tenker du om implementasjonen av `findFood()`? Er dette et bra grunnlag for vedlikehold og fremtidige utvidelser?**

(Ca. 1 avsnitt)

Skriv ditt svar her

Koden burde vært mye mer delt opp og brukt hjelpemetoder som er mer abstrakte. Det bryter dirkete med single-responsibility prinsippet da findFood metoden gjør. The open-closed prinsippet er kanskje ikke direkte brukt med koden er så inviklet med mange nested for loops at den blir lukket for utvidelser. Det er ingen brudd op the Lisknov subsitution prinsippet som jeg finner. Det er heller ikke brudd på the interface segreagtion prinsippet da det ikke er brukt noe interface, dette er også kritikk da interface hadde gjort koden mye bedre. Det er stort brudd på The dependecy inversion prinsippet som sier at "depend on abstractions, not concretions". Her er det nesten ingen abstraksjon og alt baseres på primitive verdier.

Ord: 117

Maks poeng: 3

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

5 5 1 8 9 6 8

3.5 Renovasjon [4 poeng]

Anta at noen allerede har implementert og testet API-et du laget, og at `Goat`-klassen er oppdatert til å bruke det nye API-et i stedet for `int[][] foodMap` o.l.

- **Lag en ny implementasjon av `findFood()` som bruker API-et ditt.** Du står fritt til å endre argumenter og gjøre passende antakelser om feltvariabler og andre metoder i `Goat`.

Skriv ditt svar her

```
1  public interface animal {
2      //since we are adding more animals later i collect all shared methods in an
3
4      //returns the steps to a perticular mapCell
5      Step stepsFrom(mapCell);
6
7      //returns the direction toward a perticular mapCell
8      Direction directionTo(mapCell);
9
10 }
11
12 public class Goat implements animal {
13     Position pos;
14     Direction dir;
15     Postion dest;
16
17
18 public boolean findFood(Map map, Step maxSteps){
19
20     MapCell cell = map.get(cell);
21     if(cell.T.instaceof(Food)){
22         return true
23     }
24     for(MappCell cell : map.getAll()){
25         if(cell.T.instanceOf(food)) && stepsFrom(cell) < maxSteps){
26             dest = cell;
27             direction = directionTo(cell)
28             return true;
29         }
30     }
31     return false;
32 }
33 @Overide
34 public Step stepsFrom(MapCell){
35     /*...*/
36 }
37
38 @Overide
39 private Direction directionTo(cell){
40     /*...*/
41 }
42
```

Maks poeng: 4

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

7 5 0 1 3 1 4

3.6 Refleksjon [3 poeng]

Hva tenker du om din implementasjon av `findFood()`? Er den en forbedring i forhold til den gamle?
(Noen få setninger)

Skriv ditt svar her

Min er kortere og bruker mer abstrakte løsinger som gjør den leselig og lett å forstå. Bruken av metoder som er selvforklarende gjør også den lett å justere på. Logikken er også mye letter å følge: Først sjekk om det er mat der geiten står, hvis ikke se gjennom alle cellene til kartet, om cellen har mat og er innenfor rekkevidde. Gå ditt!

Ord: 63

Maks poeng: 3

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

2 8 9 5 0 4 9

4.1 Brukerhistorier [3 poeng]

Som UiB-student har du erfaring som bruker (i student-rolllen) i den digitale eksamensløsningen Inspera. Tenk gjennom erfaringene dine, brukeropplevelsen og hva som er viktigst for deg som student når du bruker Inspera til å besvare eksamen.

- Med utgangspunkt i student-rollen og deg selv som *persona*, **skriv to brukerhistorier for en digital eksamensløsning.** For eksempel noe du synes fungerer spesielt godt eller dårlig, eller noe du savner i dagens løsning.

(To punkter)

Skriv ditt svar her

Som en elev vil jeg ha kodeautofylling og LSP-support slik at man har mer tid å bruke på kvalitet og ikke skrive mye repetitivt.

Som en elev vil jeg ha mulighet til å koble til eget trådlåst ergonomisk tastatur for å slippe skrive pauser grunnet smerter i håndledd.

Ord: 48

Maks poeng: 3

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

0 1 0 8 9 2 1

4.2 Universell utforming [4 poeng]

Universell utforming av programvare er viktig for at så mange som mulig skal kunne delta i samfunnet på en likeverdig måte, uavhengig av eventuell funksjonsnedsettelse, kulturell bakgrunn, kjønnsidentitet, alder osv. Både EU, USA, Norge og andre land har regler om dette – i Norge er det regulert gjennom *Likestillingsloven* og *Forskrift om universell utforming av IKT-løsninger*, og *UU-tilsynet* fører tilsyn med og gir råd om universell utforming.

a. **Hva slags konsekvenser kan det eventuelt få for en virksomhet (privat eller offentlig) hvis de tilbyr en nett-tjeneste som ikke oppfyller minimumskravene til universell utforming?**
(Svar kort.)
b. Universell utforming av eksamen er viktig for å sikre like muligheter i samfunnet – det er noe både utviklere og forelesere må være oppmerksom på. Tenk gjennom brukeropplevelsen på eksamen i dag og eventuelt andre eksamener du har tatt**.**
Kommer du på et tilfelle der du føler at brukeropplevelsen kunne være til hinder for enkelte studenter, selv om den kanskje er grei for de fleste? Forklar kort.
*(Det trenger ikke være snakk om et faktisk brudd på regler eller retningslinjer, bare at **du** tenker det kan være til hinder for noen. Hvis du ikke kommer på noe eksamensrelatert, kan du eventuelt tenke på andre tjenester/applikasjoner du bruker.)*

(1–2 avsnitt)

Skriv ditt svar her

a) Konsekvenser kan være alt fra tapte kunder som påvirker virksomheter, til straff fra myndighetene i form av bøter. Siden det også er veldig mye fokus i samfunnet på likestilling så kan virksomheter også oppleve press/boikotter fra samfunnet om de ikke følger universell utforming.

b)

Jeg tenker at det kan være hinder for folk fra en annen kulturell bakgrunn at eksamen digitalt krever at man klarer å skrive relativt fort på pc, da ofte ord-kravene er høyere enn når man skriver for hånd. Elever fra spesielt familier fra andre kulturer som ikke nødvendigvis er oppvokst forran tastaturet og kan skrive ute å tenke seg om bruker mye tid og krefter bare på å beherske skrivingen.

Ord: 115

Maks poeng: 4

Knytte håndtegninger til denne oppgaven? **3 4 6 8 0 6 3**

Bruk følgende kode:

5 Poeng fra semesterprosjektet [40 poeng]

God sommer!

Maks poeng: 40

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

5 7 6 5 0 1 7