NTNU
Kunnskap for en bedre verden

Sensorveiledning IDATx1003

Høst 2024

Om emnet IDATx103 Programmering 1

Emnebeskrivelse

(https://www.ntnu.no/studier/emner/IDATG1003)

Emnet er felles på tvers av campusene Gjøvik, Ålesund og Trondheim. Emnet har egen emnekode pr campus: IDAT**G**/IDAT**T**/IDAT**A** 1001, vi refererer til emnet som **IDATx1003**.

IDATx1003 Programmering 1 dekker følgende læringsutbytter i OOP:

- Klasser og objekter, samhandling av objekter
- Datatyper, variabler og konstanter
- Metoder med og uten retur og parametre
- Betingelser (if- og switch uttrykk)
- Løkker (for-each, for og while), primært i forbindelse med **samlinger** (ArrayList, HashMap HashSet)
- Samlinger av objekter og klasser for å håndtere disse (ArrayList, HashMap og HashSet osv)
- Iterator-klassen som hjelp ifm å iterere over en samling
- Designprinsippene **coupling**, **cohesion** og **responsibility driven design**.
- Dokumentasjon, kodestil og robust kodedesign (studentene har krav på seg å kjøre **CheckStyle** for å sikre god kodestil.
- Testing (enhetstesting/JUnit), debugging og robusthet
- Enkel unntakshåndtering (nytt høsten 2024)
- Enkelt tekstbasert/menybasert brukergrensesnitt.
- Grunnleggende diagrammer i UML (for eks, aktivitetsdiagram, klassediagram, sekvensdiagram, pakkediagrammer)
- Grunnleggende versjonskontroll med Gitlab/Github

Emnet dekker **ikke**:

- Maven (men prosjektet skal være satt opp i maven)
- Arv, polymorfi, interfaces
- Filhåndtering
- Grafisk Brukergrensesnitt
- For versjonskontroll dekkes ikke greiner (branches) og merge-konflikter

Lærebok og Verktøy/IDE

Lærebok benyttet i Ålesund og Gjøvik er «Objects First with Java» av David J. Burnes og Michael Kölling.

I Trondheim benyttes « Core Java Volume I - Fundamentals 11. edition» av Cay S. Horstmann.

Studentene bruker profesjonelle IDE'er som IntelliJ, VSCode, Eclipse, Netbeans etc. (studentene velger selv) men er pålagt å bruke Maven-oppsett som byggesystem i mappen.

NTNU
Kunnskap for en bedre verden

Om Mappen

Mappen som er levert for sensur består av:

- En prosjektoppgave i programmering som kandidaten har jobbet **individuelt** med i ca 10 uker. Kandidaten har fått mulighet for **tilbakemelding muntlig** på sitt arbeid 3 ganger i løpet av de første 8 ukene, i tillegg til bistand i lab-timene ved behov. Prosjektoppgaven er lastet opp i Inspera sammen rapport.
  - *Prosjektet (kode) teller 70% av endelig karakter*
- En rapport, der kandidaten skal beskrive sin løsning, hvilke valg kandidaten har gjort under utviklingen av løsningen, og hvilke endringer (refaktorering) som er gjort. Kandidaten skal også begrunne sitt design basert på teoriene rundt **coupling**, **cohesion** og **responsibility driven design**, med konkret henvisning til eksempler fra koden.
  - *Rapporten teller 30% av endelig karakter*

Kravspesifikasjonen til prosjektet er levert ut til kandidaten i 3 deler:

- Del 1 i uke 40 - Beskriver prosjektet i sin helhet, med hva man skal utvikle i løpet av prosjektet. For Del 1 skal kandidaten fokusere på å implementere **entitetsklassen/entitetsklasser** med tilhørende **enhetstester**.
- Del 2 i uke 43 - Fokuserer på **register/registre** som holder på- og administrerer objekter av **entitetsklassen/entitetsklassene** og nødvendig funksjonalitet i registeret
- Del 3 i uke 45 - Med fokus på **brukerinteraksjon**. I denne delen åpnes også oppgaven opp betydelig og gir rom for at kandidaten skal foreslå egne utvidelser, egne endringer til opprinnelig kravspek osv. for å sette sitt eget personlige preg på løsningen.

***Prosjektoppgaven høsten 2024*:**

Kandidaten skal utvikle en løsning for å redusere kasting av mat fra husholdningen. Oppgave er tilbydt på 2 **nivåer**:

Nivå 1: Et system som holder oversikt over matvarer i kjøleskap/matlager i husstanden.

Nivå 2: Utvidet funksjonalitet fra nivå 1 der systemet også skal kunne håndtere ulike **matoppskrifter** og enkelt kunne identifisere om en oppskrift kan lages basert på innholdet i kjøleskap/matlager.

Nivå 1 og 2 er tenkt skal gi muligheter for kandidater på alle nivå. Det vil forventes at en A-kanidat har løst begge nivåer. En kandidat som har løst Nivå 1 og ikke Nivå 2, vil som utgangspunkt typisk vurderes i utgangspunktet til en C (forutsatt at øvrige vurderingskriterier er oppfylt).

Brukergrensesnittet skal være **tekstbasert (konsoll)**, og kan enten være **meny-basert** med ulike valg, eller implementert som **kommandoer/kommandolinje**.

Sensor mottar full prosjektbeskrivelse i eget dokument.

Om *Verktøy*/IDE

Vi har ikke stilt krav om ett bestemt IDE, men studentene skal sett opp prosjektet som et ***Maven*** prosjekt. Studentene får selv velge hvilket IDE de ønsker å bruke. Følgende IDE'er benyttes typisk av studentene:

- IntelliJ
- VSCode

Karakterskala ved NTNU

Følgende karakterskala benyttes ved NTNU:

Symbol	Betegnelse	Generell, ikke fagspesifikk beskrivelse av vurderingskriterier
A	Fremragende	Fremragende prestasjon som klart utmerker seg. Kandidaten viser svært god vurderingsevne og stor grad av selvstendighet.
B	Meget god	Meget god prestasjon. Kandidaten viser meget god vurderingsevne og selvstendighet.
C	God	Jevnt god prestasjon som er tilfredsstillende på de fleste områder. Kandidaten viser god vurderingsevne og selvstendighet på de viktigste områdene.
D	Nokså god	En akseptabel prestasjon med noen vesentlige mangler. Kandidaten viser en viss grad av vurderingsevne og selvstendighet.
E	Tilstrekkelig	Prestasjonen tilfredsstiller minimumskravene, men heller ikke mer. Kandidaten viser liten vurderingsevne og selvstendighet.
F	Ikke bestått	Prestasjon som ikke tilfredsstiller de faglige minimumskravene. Kandidaten viser både manglende vurderingsevne og selvstendighet.

Sensurskjema

Her følger sensurskjema som er lagt til grunn for sensureringen av mappene ved alle 3 campus (Trondheim, Gjøvik, Ålesund).

Ved sensurering benyttes poenger og vekting, men kun som **veiledende** (obs - ingen «eksakt matematisk vitenskap»). Det vil alltid gjøres en total helhetsvurdering av kandidatens besvarelse før endelig karakter settes.

Sensurskjema er delt inn i 3 deler:

- Del 1 – Funksjonelle krav, og brukervennlighet
- Del 2 – Implementasjon/kode/løsning
- Del 3 – Rapport

Del 1 - Funksjonelle krav

Funksjonelle krav
Hvor godt oppfyller løsningen de **funksjonelle kravene** (Nivå1 og Nivå2) i kravspesifikasjonen (del 1 - 3) ?
Brukervennlighet/UI-design
Prosjektet skal leveres med et tekstbasert brukergrensesnitt. Følgende skal vurderes i forhold til brukervennlighet og design av brukergrensesnitt: • Fremstår brukerinteraksjonen som "brukervennlig"/intuitivt? M.a.o. det er enkelt å forstå for bruker hva hen skal gjøre for å utføre de ulike mulighetene i applikasjonen? • Er det greit å forstå informasjonen som presenteres? Presenteres informasjon med et godt detaljnivå (enhet f.eks.) • Håndteres feil inntasting av bruker på en god måte? • Er det en logisk og fornuftig flyt i brukergrensesnittet?

Del 2 - Implementasjon/Design

KRITERIUM	Ikke bestått (tilsvarer F)	Svak (tilsvarer D/E)	Middels (tilsvarer C)	Utmerket (tilsvarer B/A)
Grunnleggende OOP				
Kan kandidaten deklarere en klasse?	Kandidaten viser betydelig manglende forståelse for hvordan deklarere en klasse	Kandidaten viser noen mangler, f.eks. ved: • Varierende Navnekonvensjon • Varierende grad av Innkapsling	Kandidaten deklarerer klasser med: • Riktig navnekonvensjon for klasser, variabler og pakker • Nødvendige konstruktører • God innkapsling	Kandidaten har i tillegg vist: • Robust(e) konstruktør(er)
Datatyper og variabler		Kandidaten viser noen manglende forståelse for datatyper: • Velger å bruke **String** der det typisk burde vært brukt **int** eller **double** eller en **klasse-type** (spesielt i retur fra metoder) • Lite gode beskrivende navn på variabler. Følger ikke navnekonvensjon for variabler.	Kandidaten benytter fornuftige datatyper til felt- og variabler: • Riktige datatyper i forhold til hva variablene representerer. • Gode beskrivende navn på variabler	Kandidaten har: • Hensiktsmessig bruk av **final**/**static** • Har opprettet egne datatyper i form av klasser der dette er fornuftig.

KRITERIUM	Ikke bestått (tilsvarer F)	Svak (tilsvarer D/E)	Middels (tilsvarer C)	Utmerket (tilsvarer B/A)
Get- og set-metoder		Kandidaten viser noe manglende forståelse for fornuftig bruk av get- og set metoder: • Har implementert get- og public set metoder for samtlige felt helt ukritisk. Ingen vurdering av hvilke set-metoder som **det er naturlig** gjøres tilgjengelig (public) • Har ikke, eller i svært liten grad validert parametre i set-metodene • Benytter ikke seg av set-metoder i konstruktør.	Kandidaten har en fornuftig bruk av set- og get metoder: • En get-metode pr felt det er naturlig å gjøre tilgjengelig • Kun public set-metoder for felt det er naturlig at skal endres etter at objektet er opprettet. • Noen grad av validering av parametre.	Kandidaten har i tillegg: • Set-metoder for **samtlige** felt i klassen, **MEN** kun et fåtall er satt til **public** • Set-metoder der det **alltid** utføres validering av parameter • Aktiv bruk av set-metoder i konstruktør(ene) for å redusere/forhindre duplisering av kode.
Metoder i klasser		Kandidaten viser manglende forståelse for deklarasjon av metoder: • Følger ikke navnekonvensjon • Metoder returnerer feil/ufornuftig datatype (typisk String der det burde vært returnert int eller double osv. • Misbruk av **toString()** - benyttes for å lage en streng som skal presenteres sluttbruker.	Kandidaten viser god forståelse for deklarasjon av metoder i en klasse: • Samtlige metoder følger navnekonvensjon. • Metoder returnerer **void** der dette er naturlig. • Metoder returnerer fornuftige datatyper der det er fornuftig	Kandidaten har i tillegg: • I metoder som tar parameter valideres parameter(ne) før øvrig kode i metoden ("guard condition") • Svært gode og beskrivende navn på metoder.
Samlinger (ArrayList, HashMap osv)				

KRITERIUM	Ikke bestått (tilsvarer F)	Svak (tilsvarer D/E)	Middels (tilsvarer C)	Utmerket (tilsvarer B/A)
Grunnleggende forståelse av **samlinger**		Kandidaten har: • Ikke egen klasse som representerer vareregister, men bruker isteden ArrayList (el.l.) direkte i Main-klassen el.l. • Har brukt **primitiv array** istedenfor en av klassene i **collection-biblioteket.**	Kandidaten viser grunnleggende forståelse for **samlinger:** • Bruker riktig samling for å håndtere register. • Benytter innebygde metoder i samling-klassen der det er mulig.	Kandidaten har i tillegg: • Benyttet avanserte samlinger for håndtering av register • Eventuelt: kandidaten beskriver at hen har vurdert bedre løsninger og konkludert med annen løsning.
Løkker				
For-each og while-løkker		Bruker ikke løkker iht anbefalingene: • Avbryter for/for-each løkke med **break** eller **return** i løkken • I while-løkker: samler ikke alle betingelser for å iterere i selve **while-statmenetet** (f.eks. bruker return og/eller break inne i while løkka. • for/while-løkke med index istedenfor den bedre løsningen med Iterator	Kandidaten viser grunnleggende forståelse av løkker: • for-each for samlinger - når **hele** samlingen skal itereres over • while-løkke, når det **søkes i en samling**	Kandidaten har i tillegg: • Konsekvent bruk av kun for-each/for løkker når man skal iterere over samtlige objekter, og ikke ellers. • Bruker **while med iterator** • Kandidaten bruker **streams** og **filters** istedenfor løkker. Må da gå ut ifra at kandidaten også behersker for- do while-løkke selv om kandidaten ikke bruker dette i sin løsning.
Kodekvalitet/Design				

KRITERIUM	Ikke bestått (tilsvarer F)	Svak (tilsvarer D/E)	Middels (tilsvarer C)	Utmerket (tilsvarer B/A)
Er koden godt dokumentert iht JavaDoc-standard?	Ingen dokumentasjon	Kandidaten har: • Sporadisk dokumentasjon av klassene • Sporadisk dokumentasjon av metoder • Dokumentasjon av typen "This methode...", eller dok som beskriver hvordan og ikke hva • Mangelfull/fraværende bruk av "@param", "@return" osv	Kandidaten viser grunnleggende forståelse for dokumentasjon ved at: • Samtlige klasser er dokumentert med god beskrivelse av rolle/ansvar. • De fleste metoder som er public er godt dokumentert. • Grei "ordlyd" i dokumentasjonen (Ikke "This method...", "Gets the.." osv) • Gjennomgående bruk av "@param" og "@return" osv	Kandidaten viser i tillegg: • Gjennomført høy kvalitet i dokumentasjon • God formulering som klart dokumenterer klasser og metoder • "@param", "@return" konsekvent brukt • Innslag av HTML-formatering for bedre lesbarhet • CheckStyle med Google-styles rapporterer 0 eller minimalt med feil for JavaDoc
Er koden robust?	Ingen verifisering	Kandidaten har: • Sporadisk eller ingen sjekk av parameterverdier	Kandidaten viser grunnleggende forståelse av robust kode ved: • Gjennomgående grei verifisering av parameterverdier. • Håndterer ugyldige verdier på en god måte (unntak).	Kandidaten har i tillegg: • Verifiserer *samtlige* parameter. • Der parameter har ugyldig verdi kastes fornuftige unntak
Har variabler, metoder og klasser beskrivende navn?		Kandidaten har: • Svært mye bruk av variabelnavn som ikke gjenspeiler data de representerer/holder. • Navn på metoder gjenspeiler i ingen eller liten grad **hva** metoden gjør.	Kandidaten viser grunnleggende forståelse: • Gjennomgående gode variabelnavn som gjenspeiler data de representerer/holder. • Gjennomgående gode navn på metoder som gjenspeiler **hva** metoden gjør. • Klassenavn beskriver godt rollen/ansvaret til klassen	Kandidaten har i tillegg: • Svært gode variabelnavn som gjenspeiler data de representerer/holder. • Svært gode navn på metoder som gjenspeiler **hva** metoden gjør. • Alle klassenavn beskriver godt rollen/ansvaret til klassen

KRITERIUM	Ikke bestått (tilsvarer F)	Svak (tilsvarer D/E)	Middels (tilsvarer C)	Utmerket (tilsvarer B/A)
Har koden god struktur, med løse koblinger og høy kohesjon?		Kandidaten viser manglende forståelse: • Det er tett kobling mellom klasser som ikke burde hatt kobling (high coupling) • Mange metoder som utfører flere oppgaver (low cohesion) • Mange klasser med flere roller/ansvar	Kandidaten viser grunnleggende forståelse: • Generelt god løs kobling mellom klasser som ikke burde hatt kobling (low coupling) • Jevnt over de fleste metoder utfører en enkelt oppgave (high cohesion). Der en metode må utføre flere del-oppgaver, benyttes **deligering** til sub-metoder. • De fleste klasser har **en** klart definert rolle/ansvar	Kandidaten har i tillegg: • Gjennomgående svært god løs kobling mellom klasser som ikke burde hatt kobling (low coupling) • Samtlige metoder utfører en enkelt oppgave (high cohesion). Der en metode må utføre flere del-oppgaver, benyttes **deligering** til sub-metoder. • Samtlige klasser har **en** klart definert rolle/ansvar
Versjonskontroll og enhetstesting				
Enhetstesting				
	Har ingen enhetstester	• Har noen enhetstester/tester noen få av klassene • Mangler negative tester/har ikke forstått "negativ test" • Skriver tester som medfører at tester må kjøres i en bestemt rekkefølge • Ikke gode beskrivende navn på testene	• Har gode beskrivende navn på testene • Følger prinsippet om Arrange-Act-Assert • Har enhetstester for de grunnleggende klassene • Har helt greie negative tester (viser at kandidaten har forstått hovedpoenget med positive/negative tester)	• Har også gode negative tester for alle test-klassene • Tester Exceptions • Benytter seg av "@BeforeAll", "@BeforeEach" osv - der det er hensiktsmessig • Har svært høy "test coverage" for aktuelle klasser • Har svært gode negative tester
Versjonskontroll				
Lokalt og sentralt repo, commits	Har ikke benyttet versjonskontroll	• Prosjektet er lagt til versjonskontroll lokalt • Sporadiske innsjekkinger (commits) - mye som er endret mellom hver commit • Mangelfulle commit-meldinger (enten tomme, eller ikke beskrivende for endringene som er utført) • Ikke sentralt repo	• Prosjektet har sentralt repo (GitHub/GitLab) • Fornuftig jevnlig innsjekking (commit) av endringer • Gode commit-meldinger som beskriver kort hvilke endringer som er gjort/hvilke problem som er løst	• Svært gode commits og svært gode commit-meldinger. • Har benyttet tags for å merke versjoner

KRITERIUM	Ikke bestått (tilsvarer F)	Svak (tilsvarer D/E)	Middels (tilsvarer C)	Utmerket (tilsvarer B/A)
Totalintrykk				
Her vurderes totalinntrykket til kandidaten. Dette kriteriet kan benyttes for vurdering som ikke fanges opp av de øvrige kriteriene.		Kandidaten presterer under gjennomsnitt. Løsningen er unødvendig tungvint implementert, eller er mangelfull.	En prestasjon som er midt på treet totalt sett. Kandidaten viser at han/hun har forstått de fleste læringsmål og viser god forståelse.	En fremragende løsning på alle måter. Kandidaten benytter elementer i sin løsning som er ut over det som er forventet innenfor gjeldende pensum. NB! Å benytte arv, og grafisk brukergrensesnitt inngå **ikke** i "..over forventet.."

Del 3 - Rapport

Kandidatene fikk utlevert en mal for Word. I malen var det gitt en mengde *hjelpetekst* for å hjelpe kandidaten til å forstå hva som er forventet i de ulike kapitlene i rapporten. Malen baserer seg på strukturen ofte benyttet i vitenskapelige artikler og i Bachelor rapporter. Hovedvekten av tekst bør være i kapitlene om resultat og drøfting. Kapitlene «Innledning, problemstilling, teori og metode» forventes å holdes korte (oppsummerende) og inneholde relevant informasjon som blir referert tilbake til i drøftingen.

KRITERIUM	**Ikke bestått (tilsvarer F)**	**Svak (tilsvarer D/E)**	**Middels (tilsvarer C)**	**Utmerket (tilsvarer B/A)**
Struktur	Kaos	Elementer mangler eller er plassert på ulogiske steder	Grei struktur, fin rød tråd og ting er beskrevet der de hører hjemme	Ekstra elementer som f.eks. • Kryssreferanser • bibliografi
Kravspesifikasjon (Kapittel "Innledning - Problemstilling")	Mangler	Vanskelig å forstå hva som skal utvikles basert på beskrivelsen.	Har beskrevet de viktigste funksjonelle kravene.	Utfyllende med f.eks • Avgrensninger. • Passende UML diagrammer • Andre typer krav (systemkrav, ikke-funksjonelle krav)
Teoretisk grunnlag Hvilke **teorier** er benyttet ved utviklingen av løsningen?	Mangler	Mangelfullt beskrevet. Beskriver bare deler av teorigrunnlaget.	God beskrivelse av teoriene om god design. Men beskriver teorier som kandidaten senere ikke refererer til	Har i tillegg inkludert: • Teori om en eller flere elementer • Teori og praksis rundt **kodestil** • Teori rundt god brukerinteraksjon Har kun omtalt teorier som senere diskuteres i rapporten.
Metode Hvilke **verktøy** er benyttet for å løse oppgaven? Hvilken **prosess/prosjektmodell** ble fulgt? I denne mappen har vi f.eks. gitt prosjektet i 3 deler med tilbud om 3 tilbakemeldinger.	Mangler	Mangelfull beskrivelse av IDE og prosess/metode	• Beskriver hvilke verktøy (IDE etc.) som er brukt. • Versjonskontroll	• Beskriver i tillegg greit arbeidsmetoden (hvilken arbeidsmetode kandidaten har fulgt) • Versjonskontroll og refaktorering

KRITERIUM	Ikke bestått (tilsvarer F)	Svak (tilsvarer D/E)	Middels (tilsvarer C)	Utmerket (tilsvarer B/A)
Design og Implementasjon (i kapittel "Resultat")	Mangler	Utydelig, manglende diagram. Skriver «dagbok», og ikke en beskrivelse av rolle- og ansvar til de ulike klassene.	Godt beskrevet • med bruk av klassediagram • med god bruk av f.eks. kodeeksempler og skjermdump • med god beskrivelse av hvilke **refaktorering** er gjort og hvorfor?	Ekstra utfyllende med f.eks. • Diagrammer og illustrasjoner • Resultat av gjennomførte **brukertester** • Peker på konkrete valg som er tatt, med begrunnelse. • Ryddig og oversiktlig.
Drøfting og refleksjon, Konklusjon	Mangler	Diskusjon: Knapt beskrevet. Ingen reell *drøfting* av egen løsning i forhold til teoriene beskrevet i Teori-avsnittet.. Konklusjon: Svært kort, der kandidaten ikke peker på de viktigste oppnådde målene i prosjektet	Drøfter greit endelig resultat i forhold til **teoriene**. God konklusjon der kandidaten oppsummerer prosjektet i forhold til: • Om kandidaten mener prosjektet er iht de prinsipper og standarder adressert i emnet • Om kandidaten har fått vist sine ferdigheter igjennom prosjektet	Drøfter **svært godt** endelig resultat i forhold til **teoriene** fra læreboken. Viser ved eksempler fra egen kode at løsningen følger designprinsippene og beste praksis gjennomgått i emnet. Konklusjon: Har også fått med f.eks. hva som burde vært gjort annerledes om det skulle gjøres igjen