UiT
NORGES
ARKTISKE
UNIVERSITET

EKSAMENSOPPGAVE

Eksamen i:	INF-1400 Objektorientert Programmering
Dato:	Tirsdag 21. mai 2019
Klokkeslett:	Kl 09:00 - 13:00
Sted:	Adm.bygget, Aud. max og B154
Tillatte hjelpemidler:	Ingen
Type innføringsark (rute/linje):	Digital eksamen
Antall sider inkl. forside:	5
Kontaktperson under eksamen: Telefon/mobil:	John Markus Bjørndalen og Edvard Pedersen 90148307 (JMB) og 40458598 (EP)
Runde i eksamenslokalet ca. kl.: 11.	

NB! Det er ikke tillatt å levere inn kladdepapir som del av eksamensbesvarelsen.

Hvis det likevel leveres inn, vil kladdepapiret bli holdt tilbake og ikke bli sendt til sensur.

Les gjennom hele oppgavesettet før du begynner å løse oppgavene.

Oppgavene kan besvares på Norsk, Engelsk, Svensk eller Dansk.

I besvarelsen kan du bruke Python, pseudokode eller en kombinasjon.

Noen oppgaver har kun en “a)”. Dette er for å tydeliggjøre hva selve spørsmålet som skal besvares er, det har ikke falt ut noen delspørsmål.

Noen norske termer som er brukt i stedet for de engelske fra boka:

- Klasse - class
- Arv - inheritance
- Atributt - Attribute
- Metode - Method

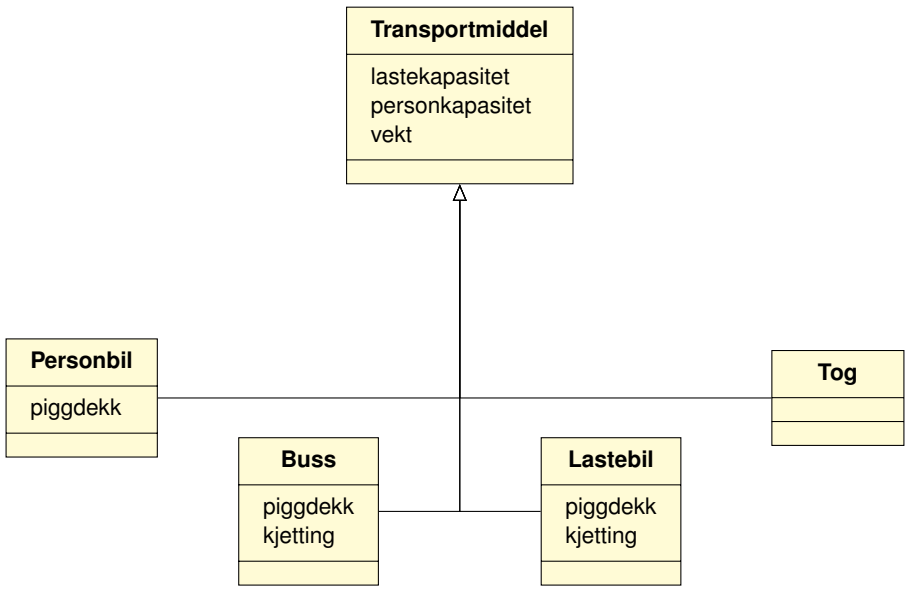
Del A

I forbindelse med debatten rundt tog i Nord-Norge dukker spørsmålet om veislitasje og ulykker opp. For å forstå litt mer om problemstillingen ønsker vi å simulere forskjellige scenarier med bruk av tog, buss, lastebil og personbil for framtidig økt behov for transport av mennesker og last.

For å gjøre oppgaven litt enklere fokuserer vi bare på veislitasje i denne oppgaven. Vi tar derfor med bare noen relevante metoder og attributter og overser også enkelte ting som er relevante for slitasje.

Oppgave 1 (20%)

Følgende klassediagram angir et utgangspunkt for klasser vi kan bruke i simuleringen.



a) Implementer klassehierarkiet som er angitt i UML-diagrammet over. Få tydelig fram arv og attributter. Du trenger ikke skrive noen metoder her.

b) Implementer `__init__`-metoden klassen Lastebil og lag et objekt av typen Lastebil. Hvilke attributter har objektet?

Oppgave 2 - 20 %

Som et første anslag regner vi ut slitasje fra en gitt faktor for hver type transportmiddel. Gitt en liste med transportmiddel (et antall biler, busser, lastebiler og tog) skal vi regne ut slitasjen på en veistrekning.

En start vil være å lage en funksjon som regner ut slitasjen ved å se på hvert av objektene i listen og summere faktorer som er angitt for hver type transportmiddel.

Tog settes til en faktor 0 for å kunne ta hensyn til transporten som vi har flyttet fra veien i andre deler av simuleringen.

```
FAKTOR_BIL = 10
FAKTOR_BUSS = 100
FAKTOR_LASTEBIL =  300
FAKTOR_TOG = 0

def slitasje(transportmiddel):
    """Regner ut veislitasjen for en liste med transportmiddel som passerer strekningen. """
    pass
```

a) Lag funksjonen `slitasje` som regner ut veislitasjen basert på hvilken type hvert transportmiddel er. Konstantene over (`FAKTOR_*`) angir hvor mye hvert transportmiddel sliter på veien når det passerer.

Ulempen med dette er at vi må endre `slitasje` og legge til konstanter hvis vi ønsker å legge til flere klasser. Vi kan gjøre dette mer fleksibelt ved å erstatte konstantene over med en **klasseattributt** (`SLITASJE_FAKTOR`) i hver av klassene.

b) Vis hvordan du gjør dette ved å bruke `Lastebil`-klassen som eksempel. Vis hvordan du implementerer `slitasje` når du kan bruke klasseattributten `SLITASJE_FAKTOR` i alle objektene/klassene.

Et lite hint: funksjonen i b) burde bli merkbart enklere enn den i a).

Oppgave 3 - 25 %

Måten vi regnet ut slitasje over er for grov og tar ikke godt nok hensyn til forskjellene mellom transportmidlene. For å å komme et stykke videre introduserer vi en slitasje-metode i klassehierarkiet og endrer på den gamle slitasje-funksjonen for å utnytte disse.

Som et startpunkt har vi følgende utregninger for de forskjellige transportmidlene:

```
# Utregning av slitasje for forskjellige objekter (må legges i metoder).

# Personbil
slitasje_verdi = self.vekt * self.SLITASJE_FAKTOR
if self.piggdekk:
   slitasje_verdi *= 3

# Lastebil eller buss
slitasje_verdi = self.vekt * math.log(self.vekt) * self.SLITASJE_FAKTOR
```

```
    if self.piggdekk:
        slitasje_verdi *= 3
    if self.kjetting:
        slitasje_verdi *= 10

    # Tog - settes til 0 = sliter ikke på veien
    slitasje_verdi = 0


    # Hovedfunksjon som regner ut total slitasje for mange objekter
    def slitasje(transportmiddel):
        total = 0
        for tm in transportmiddel:
            total += tm.slitasje()
        return total
```

a) Bruk generalisering (introduser en ny klasse) for å unngå duplisering av kode (vi ser bort fra attributter nå). Beskriv hvordan du ønsker å gjøre det.

b) Bruk polymorfi for å forbedre koden over. Beskriv hvordan du ønsker å gjøre det og hvor du legger metodene.

c) Implementer koden.

Del B

Oppgave 4 - 20 %

Vi har laget et bibliotek for å behandle dokumenter som er i bruk av tusenvis av kunder. Et utdrag fra klassen for dokumenter finnes under.

```
class Document:
    def __init__(self):
        self.id = 0
        self.name = ""
        self.contents = []

    def get_info(self):
        return (self.id, self.name, self.contents)
```

Denne klassen brukes på følgende måte av kundene.

```
d = Document()
d.id = 10
d.name = "Important document"
d.contents = "This document contains (...)"
```

Det har kommet en forespørsel om at biblioteket bør teste at `id` er et positivt tall, siden noen kunder har problemer med at det av og til blir generert dokumenter med negative tall.

p

a) Hvordan kan vi sikre at `id` kun blir satt til positive tall? Beskriv kort 2 forskjellige tilnærminger.

b) Hva bør vi gjøre når noen setter `id` til noe som ikke er ett positivt tall? Beskriv kort 2 forskjellige tilnærminger.

c) Velg en av tilnærmingene fra hvert av delsvarene over og implementer en kombinasjon av dem (riktig syntaks er ikke nødvendig). Vis hva kundene må endre hvis de må endre noe i sin kode.

Oppgave 5 -15%

a) Gi en kort beskrivelse av 2 av de følgende uttrykk/konsepter:

- Klasse vs. objekt
- Polymorfi (det holder å angi 1 variant av polymorfi)
- Multiple inheritance
- Encapsulation (innkapsling)
- Pattern (mønster)