Fakultet for naturvitenskap og teknologi

UiT
NORGES
ARKTISKE
UNIVERSITET

EKSAMENSOPPGÅVE

Eksamen i:	INF-1400 Objektorientert Programmering
Dato:	Tysdag 21. mai 2019
Klokkeslett:	Kl 09:00 - 13:00
Stad:	Adm.bygget, Aud. max og B154
Lovlege hjelpemiddel:	Ingen
Type innføringsark (rute/linje):	Digital eksamen
Antall sider inkl. framside:	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 ikkje lov å levere inn kladd saman med svaret.

Om den likevel vert levert inn, vil kladden verte halden attende og ikkje sendt til sensur.

PO Box 6050 Langnes, NO-9037 Tromsø / +47 77 64 40 00 / postmottak@uit.no / uit.no

Les gjennom heile oppgåvesettet før du byrjar å løyse oppgåvene.

Oppgåvene kan løysast på Norsk, Engelsk, Svensk eller Dansk.

I løysingane (svara) kan du nytta Python, pseudokode eller ein kombinasjon.

Nokre oppgåver har kun ein “a)”. Dette er for å tydeliggjera kva sjølve spørsmålet som skal svarast på er, det har ikke falle ut nokre delspørsmål.

Nokre norske termar som er nytta i steden for dei engelske frå boka:

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

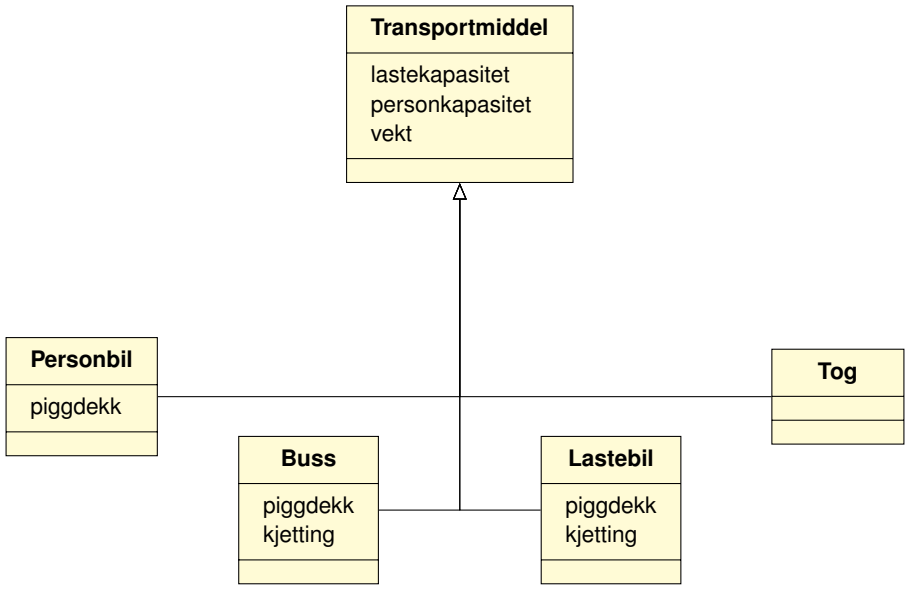
Del A

I forbindelse med debatten rundt tog i Nord-Noreg dukkar spørsmålet om vegslitasje og ulykker opp. For å forstå litt meir om problemstillinga ynskjer me å simulere forskjellige scenario med bruk av tog, buss, lastebil og personbil for framtidig auka behov for transport av mennesker og last.

For å gjera oppgåva litt einklare fokuserer me berre på vegslitasje i denne oppgåva. Me tek difor berre med nokre relevante metodar og attributtar og overser også einkelte ting som er relevante for slitasje.

Oppgåve 1 (20%)

Følgjande klassediagram angir eit utgangspunkt for klassar me kan bruka i simuleringen.



a) Implementer klassehierarkiet som er angitt i UML-diagrammet over. Få tydeleg fram arv og attributtar. Du treng ikkje skriva nokon metodar her.

b) Implementer `__init__`-metoden klassen Lastebil og lag eit objekt av typen Lastebil. Kva for attributtar har objektet?

Oppgåve 2 - 20 %

Som eit første anslag reknar me ut slitasje frå ein gitt faktor for kvar type transportmiddel. Gitt ei liste med transportmiddel (eit antal bilar, bussar, lastebilar og tog) skal me rekne ut slitasjen på ein vegstrekning.

Ein start vil vera å lage ein funksjon som reknar ut slitasjen ved å sjå på kvart av objekta i lista og summere faktorar som er angitt for kvar type transportmiddel.

Tog vert sett til ein faktor 0 for å kunne ta hensyn til transporten som me har flytta frå vegen i andre delar av simuleringen.

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

def slitasje(transportmiddel):
    """Reknar ut vegslitasjen for ei liste med transportmiddel som passerer strekningen. """
    pass
```

a) Lag funksjonen `slitasje` som reknar ut vegslitasjen basert på kva for ein type kvart transportmiddel er. Konstantane over (`FAKTOR_*`) angir kor mykje kvart transportmiddel slit på vegen når det passerer.

Ulempen med dette er at me må endre `slitasje` og leggje til konstantar dersom me ynskjer å leggje til fleire klassar. Me kan gjere dette meir fleksibelt ved å erstatte konstantane over med ein **klasseattributt** (`SLITASJE_FAKTOR`) i kvar av klassane.

b) Vis korleis du gjer dette ved å bruka `Lastebil`-klassen som døme. Vis korleis du implementerer `slitasje` når du kan bruke klasseattributten `SLITASJE_FAKTOR` i alle objekta/klassane.

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

Oppgåve 3 - 25 %

Måten me rekna ut slitasje over er for grov og tek ikkje godt nok hensyn til skilnadane mellom transportmidlane. For å å kome eit stykke vidare introduserer me ein slitasje-metode i klassehierarkiet og endrar på den gamle slitasje-funksjonen for å utnytte desse.

Som eit startpunkt har me følgjande utrekningar for dei forskjellige transportmidlane:

```
# Utregning av slitasje for forskjellige objekt (må leggjast i metodar).

# 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 - settast til 0 = sliter ikkje på vegen
    slitasje_verdi = 0


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

a) Bruk generalisering (introduser ein ny klasse) for å unngå duplisering av kode (me ser bort frå attributtar no). Beskriv korleis du ynskjer å gjera det.

b) Bruk polymorfi for å betre koden over. Beskriv korleis du ynskjer å gjera det og kor du legg metodane.

c) Implementer koden.

Del B

Oppgåve 4 - 20 %

Me har laga eit bibliotek for å behandle dokument som er i bruk av tusenvis av kundar. Eit utdrag frå klassen for dokument finst 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 brukast på følgjande måte av kundane.

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

Det har kome ein forespurnad om at biblioteket bør teste at `id` er eit positivt tal, siden nokre kundar har problem med at det av og til blir generert dokument med negative tal.

a) Korleis kan me sikre at `id` kun blir satt til positive tal? Beskriv kort 2 forskjellige tilnærmingar.

b) Kva bør me gjera når nokon set `id` til noko som ikkje er eit positivt tal? Beskriv kort 2 forskjellige tilnærmingar.

c) Velg ei av tilnærmingane frå kvart av delsvara over og implementer ein kombinasjon av dei (rett syntaks er ikkje nødvendig). Vis kva kundane må endre dersom dei må endre noko i sin kode.

Oppgåve 5 -15%

a) Gi ein kort beskrivelse av 2 av dei følgjande uttrykk/konsept:

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