UiT

NORGES
ARKTISKE
UNIVERSITET

SENSORVEILEDNING

Eksamen i: INF-1400 Objektorientert programminering
Dato: Mandag 26. mai 2014

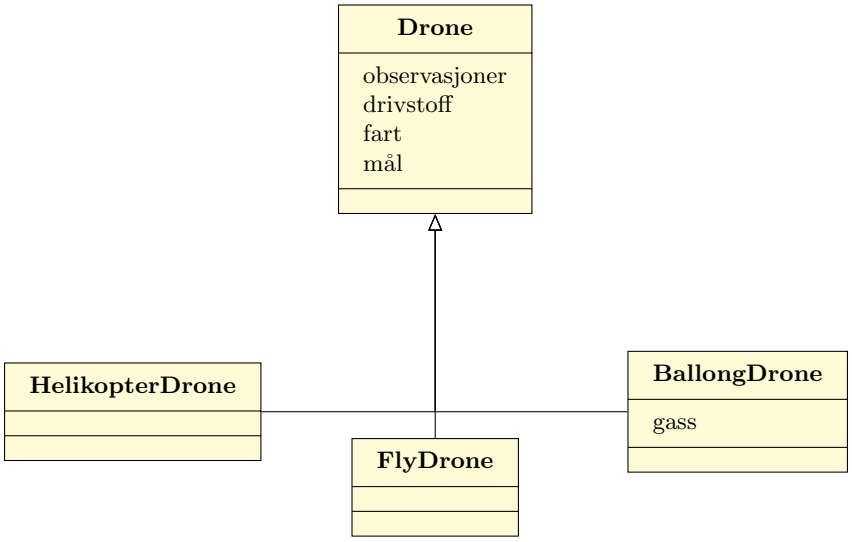
Sensorveiledningen er på 4 sider inklusiv forside.

Fagperson/intern sensor: John Markus Bjørndalen.
Telefon: 90148307

Del A

Oppgave 1 - 20%

Vi krever ikke eksakte UML-diagram (vi har brukt enkle diagram i kurset). En mulig løsning er som følger.



De viktige poengene som må med her for å få full telling:

- felles attributter flyttet til `Drone`
- `gass` ligger i `BallongDrone`
- arv indikert (og korrekt for klassene)

Det er ikke behov for å forklare hva generalisering er hvis man får resten rett.

Oppgave 2 - 20%

```
class Observasjon(object):
    def __init__(self):
        self.posisjon = None
        self.tidspunkt = None
        self.bilde = None  # bilde tatt av dronen

class Menneske(Observasjon):
    """Mennesker. mulig_infisert er True kun hvis det er en sannsynlig infeksjon. """
    def __init__(self):
        Observasjon.__init__(self)
        self.mulig_infisert = False

# python nekter ææå i klassenavn
class Baat(Observasjon):
    def __init__(self):
        Observasjon.__init__(self)
        self.baatreg = None
        self.baatstoerrelse = None
        self.baatfarge = None
```

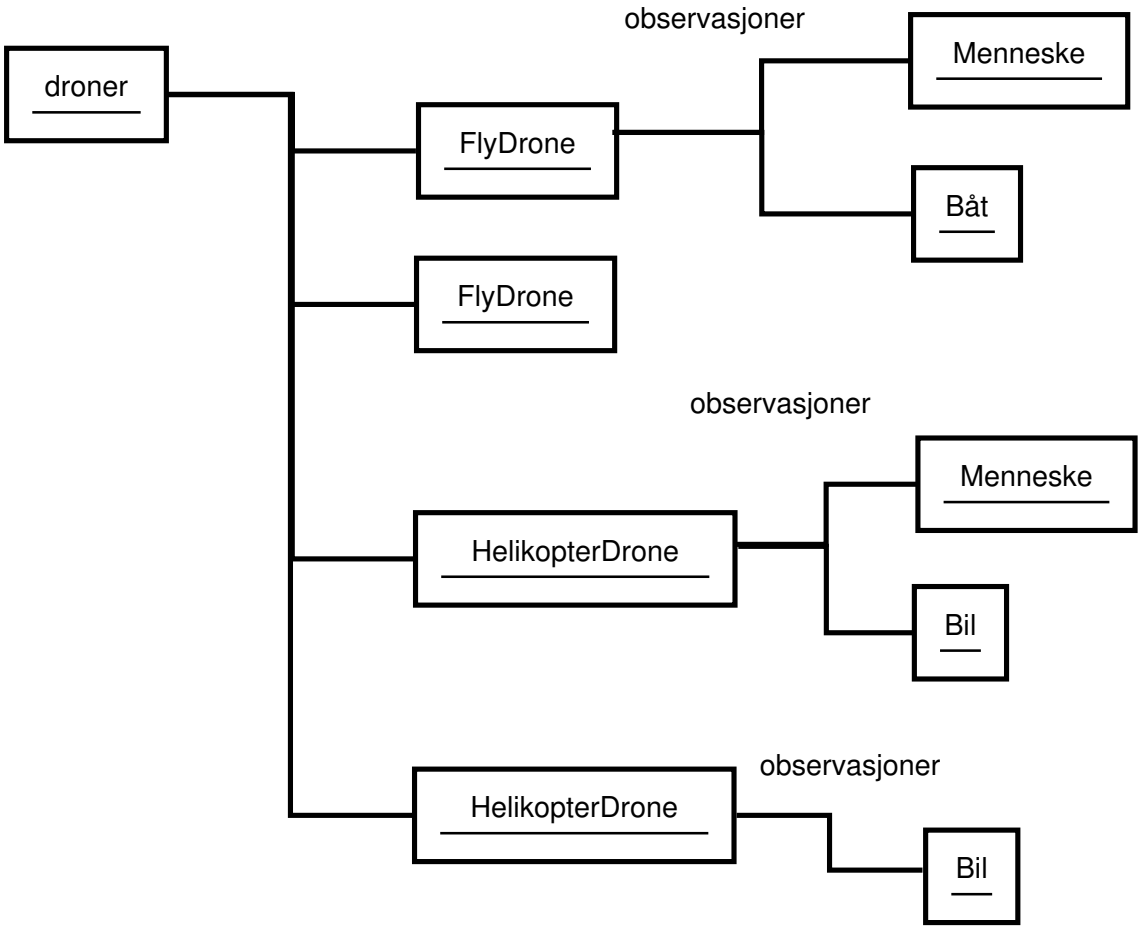
```
        self.baattype  = None

class Bil(Observasjon):
    def __init__(self):
        Observasjon.__init__(self)
        self.regnummer = None
        self.bilfarge  = None
        self.biltype   = None
```

Kriterier å se etter her:

- De skiller attributter ut i egne (rette) klasser
- Arv blir gjort korrekt (de andre arver fra `Observasjon`)
- Arv må gjøres slik at de faktisk arver attributtene (`Observasjon.__init__(self)` hvis man bruker Python).

Hva attributtene initialiseres til er ikke viktig.

Oppgave 3 - 15%



Dette er ikke en fasit siden man kan tegne det på forskjellige måter, men man bør få fram følgende poeng:

- droner er en liste av forskjellige dronetyper.
- hver drone har en liste av observasjoner
- observasjonene er av forskjellige typer

Oppgaven er delvis ment å teste at studentene klarer å skille mellom klassehierarki og objekt-relasjoner og delvis ment som en hjelp for å løse neste oppgave.

Oppgave 4- 15%

En mulig løsning:

```
def plott_mulige_infeksjoner(map, droner):
    for drone in droner:
        for obs in drone.observasjoner:
            if isinstance(obs, Menneske):
                if obs.mulig_infisert:
                    map.set_marker(obs.posisjon)
```

Følgende poeng bør med for full uttelling:

- De viser hvordan man får inspisert alle observasjoner ved å starte med drone-listen.
- De tester for type/klasse av observasjon før de sjekker `mulig_infisert`. De trenger ikke å huske at den heter isinstance så lenge de klarer å indikere at det er sjekk av type.
- At de klarer å plotte korrekt: de henter ut observasjonens posisjon.

Del B

Oppgave 5 - 15%

De trenger ikke å få med at det blir en exception, men de bør få med at siste deloppgave kommer til å trigge en feil på grunn av en manglende metode.

Ellers tester oppgaven forståelse av hvordan arv og polymorfi faktisk virker.

a) `False`

b) `True`

c) `True`

d) `cannot hoot - no owls present`

e) `Almost owl...ish`

f)
```
Traceback (most recent call last):
  File "platypus_test.py", line 12, in <module>
    p.hoot("detected")
AttributeError: 'Platypus' object has no attribute 'hoot'}
```

Oppgave 6 -15%

Det holder med en kort beskrivelse av hvert av de to punktene de velger som tar med hovedtrekkene/konseptet.

For polymorfi er det sannsynlig at de beskriver valg av metodeimplementasjon basert på objektets klasse, men de kan ha vært borti en av de andre variantene. Hvilken de velger er ikke viktig.