# EKSAMENSOPPGAVE I INF-1400

---

| | | |
|---|---|---|
| **Eksamen i** | : | **INF-1400 – Objektorientert Programmering** |
| **Eksamensdato** | : | **2010-09-28** |
| **Tid** | : | 09.00-13.00 |
| **Sted** | : | **Administrasjonsbygget, 1.et., B.154** |
| **Tillatte hjelpemidler** | : | **Ingen** |

**Oppgavesettet er på 5 sider inkl. forside**

**Kontaktperson under eksamen:**
**John Markus Bjørndalen,**
**92671202**

FAKULTET FOR NATURVITENSKAP OG TEKNOLOGI
Universitetet i Tromsø

Les gjennom hele oppgavesettet før du begynner å løse oppgavene.

**Del A, et lite problem**

*Your Destination Has Not Yet Arrived*

"Ladies and gentlemen, this is your captain speaking. I'm happy to report most of the turbulence seems to be behind us, and we're back at a cruising altitude now. The rest of your flight looks like it should be fairly smooth.

"Unfortunately, it looks like we're not going to make it to our destination, today. As you might imagine, this is due to spatial flux caused by the Proxima Centaurian spacecraft that was over London until just a few minutes ago. Their matter reorganization array has shifted our planet's geography, so London is no longer there.

"As we do not have enough fuel to turn back now, we are going to proceed on to London's former location, which seems to now be occupied by Los Angeles, formerly of California. London, meanwhile, is reported to have materialized in Peru, so please talk to your boarding agent about catching a connecting flight. If you were proceeding on to Tokyo with us, it's expected to rematerialize in about 320 minutes.

"Please, now, sit back and enjoy your flight."

- wordwill, 2009-06-05, http://ficly.com/stories/935

Romskip fra Proxima Centauri har begynt å bli et stort problem for reisenæringen. Vi trenger et verktøy for å holde rede på hvor byene til en hver tid er eller hvor de er forventet å dukke opp. Verktøyet trenger heldigvis ikke å være spesielt komplisert siden observasjonsgruppen i Paris (forventet å dukke opp nær Harstad om 6 timer) har gode metoder for å oppdage når noe har blitt flyttet, hvor det er på tur, og når det er forventet å dukke opp.

Vi definerer følgende klasser:

- PasseringsRegister – som inneholder en liste over RomskipPasseringer.
- RomskipPassering – som angir registreringsnummeret til romskipet, hvor det var observert, og når det passerte jorden. Den inneholder også en liste over byer som ble flyttet under denne passeringen.
- FlyttetBy – som angir byens navn, sist kjente posisjon, forventet ankomssted og tid.

I oppgavene står du fritt til å velge hvordan du skal beskrive attributter som posisjon og tid. En by kan for eksempel angis med posisjon 'London' for å angi at den er der London var før romskipene begynte å passere jorden.

Hvis du ikke husker eksakt hvordan noe skrives i Python kan du bruke pseudokode for å angi hva du mener.

## Oppgave 1 (10%)

I det siste har observasjonsgruppen registrert et økende antall fly og turistbåter som har blitt forflyttet. Vi må utvide listen av klasser med en FlyttetBåt- og en FlyttetFly-klasse. FlyttetBy og de to nye klassene har mye til felles, så vi kan generalisere dem ved å introdusere arv og legge til en ny klasse FlyttetTing. Listen i RomskipPassering vil nå inneholde både byer, fly og båter.

Forklar kort hvordan du kan bruke arv når du legger til de nye klassene. Det er kun behov for å fokusere på arv og på attributtene til klassene.

## Oppgave 2 (15%)

Implementer klassene vi har spesifisert til nå. Du trenger ikke å ta med noen andre metoder enn __init__() siden vi skal jobbe videre med klassene senere.

## Oppgave 3 (20%)

a) Tegn opp klassediagrammet for klassene dine. Du trenger ikke å ta med metoder.
b) Tegn opp datastrukturen med et lite antall romskipspasseringer og ting som har blitt flyttet på.

## Oppgave 4 (10%)

Vi trenger å finne ut hvor en by befinner seg til en hver tid. For å gjøre dette skal vi lage en metode finnBy() i PasseringsRegister som leter gjennom hver passering for å finne *siste* registrering av en by med angitt navn. Metoden skal da returnere en liste på følgende form:

[bynavn, sistKjentePosisjon, forventetAnkomstSted, forventetAnkomstTid]

for eksempel:

['London', 'London', 'Peru', '2010-09-28—09:00']

Hvis vi ikke finner noe om en gitt by skal vi returnere None, noe som angir at byen ikke har flyttet på seg. Vis også metoder du eventuelt velger å introdusere i de andre klassene. Du kan anta at listen over passeringer i passeringsregisteret er sortert på tid.

## Oppgave 5 (15%)

En kreativ forretningsmann ønsker å undersøke om han kan spare fraktkostnader ved å la varer haike med byer som blir forflyttet av romskip. Han betaler gode penger, så vår oppgave er å lage en metode som skriver ut alle byene som har vært på hver posisjon vi har i registeret. Dette skal han senere bruke til analyse, men vi trenger bare å lage listen for han.

Lag en metode (skrivutAnkomster) i PasseringsRegister som søker gjennom passeringsregisteret. For hvert ankomststed vi har registrert for en by skal vi skrive ut alle byene som har vært der og når de ankom. Listen trenger ikke å være sortert.

Etter hvert ønsker han også å gjøre dette for fly og båter, men i denne omgangen vil han bare se på byer som flytter seg, så vi må unngå å ta med fly og båter i utskriften.

## Del B

### Oppgave 6 (10%)

I Python har vi en konvensjon med å bruke variabelen *self* i metoder. Hva er hensikten med *self* og hva brukes den til?

Hvis du kjenner til språk som C++ og Java har de nøkkelordet *this* med tilsvarende funksjon.

### Oppgave 7 (10%)

Det siste romskipet som passerte flyttet ikke bare på by og båter, det rotet også til noe av koden vår. Vi må nå finne ut hva følgende program egentlig gjør.

```
class Sneetch:
    def __init__(self):
        self._a = 3
        self._b = 4

    def x(self):
        print self._a

    def y(self):
        print self._b

class StarBellySneetch(Sneetch):
    def __init__(self):
        Sneetch.__init__(self)
        self._b = 7
        self._c = 8

    def y(self):
        print self._b, self._c

    def z(self):
        print self._a, self._c
```

For hvert av uttrykkene under, angi hva programmet vil skrive ut. Uttrykkene kan også utløse en feil (error) i programmet. Hvis dette skjer, angi hvorfor feilen oppsto.

a) alice.x()
b) alice.y()
c) alice.z()
d) bob.x()
e) bob.y()
f) bob.z()

**Oppgave 8 (10%)**

Velg to av uttrykkene/konseptene under og gi en kort beskrivelse av dem.

- object vs. class
- parameter
- operator
- instance
- method