UiT
NORGES
ARKTISKE
UNIVERSITET

EKSAMENSOPPGAVE

Eksamen i:	INF-1400 Objektorientert programmering
Dato:	25 september 2018
Klokkeslett:	09.00-13.00
Sted:	Adm. Bygget K1.04
Tillatte hjelpemidler:	Ingen
Type innføringsark (rute/linje):	
Antall sider inkl. forside:	**5**
Kontaktperson under eksamen: Telefon/mobil:	**Lars Brenna** **90786723**
Vil det bli gått oppklaringsrunde i eksamenslokalet? Svar: NEI	

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.

Eksamen INF-1400
Objektorientert programmering
Vår 2018

Eksamenssettet består av to deler, totalt seks oppgaver.

Les oppgaveteksten grundig og disponer tiden slik at du får tid til å svare på alle oppgavene. I noen oppgaver kan det være nødvendig å tolke oppgaveteksten ved å gjøre noen antagelser - gjør i så fall rede for hvilke antagelser du har gjort, men pass på å ikke gjøre antagelser som trivialiserer oppgaven.

Merk: Fokus i oppgaven er objektorientering og bruk av datastrukturer vi bygger opp. Det gis ikke ekstra poeng for å utvide oppgaven utover det som er spesifisert.

Pseudokode er i de fleste tilfeller godtatt, det kreves ikke kjørbar kode. Python3-syntaks er foretrukket, men ikke strengt nødvendig.

Del 1

Ola og Kari er begge opptatt av kart og turer, og de ønsker å ta vare pa informasjon om hvor og når de har vært på tur. For å gjøre dette enklere så har de kjøpt en GPS-klokke som kan spore turene deres. Det vil si at mens de er på tur så vil klokken med jevne mellomrom lagre deres nåværende *posisjon*. Når de er ferdig så kan de laste over disse posisjonene til en PC. Samlet gir alle posisjonene på en tur *et spor* (en liste av posisjoner). En posisjon inneholder *lengdegrad, breddegrad, meter over havet* og *tidspunkt* (dato og klokkeslett). Ola og Kari bestemmer seg for å lage et program for å håndtere disse dataene. De starter med å identifisere følgende klasser:

- `Posisjon` — En gitt posisjon på en tur med attributtene `lengdegrad`, `breddegrad`, `moh` (meter over havet) og `tidspunkt`.
- `Spor` — Et spor for en tur som inneholder følgende attributter:
  - Liste over posisjoner (objekter av klassen Posisjon)
  - Starttidspunkt for dette sporet (tidspunkt for første posisjon)
  - Stopptidspunkt for dette sporet (tidspunkt for siste posisjon)
- `Tur` — En tur med attributtene `spor` (et Spor objekt), `navn` og `beskrivelse`.

De bestemmer seg også for å spesialisere klassen tur for de turene som er *sightseeing*. Denne klassen `SightseeTur` skiller seg fra klassen `Tur` med at den også har en *liste* over spesielt interessante posisjoner (*POI point of interest*) i løpet av en tur. Hvert enkelt element i denne listen referer til et element i sporet til denne turen (se attributtet `spor` i klassen `Tur`).

Oppgave 1 - 20%

Vi skal foreløpig konsentrere oss om attributtene (og ikke metodene) i de klassene vi jobber med. Vi kommer tilbake til metodene senere.

a) Klassen `SightseeTur` kan både realiseres med arv (inheritance) og komposisjon (composition). Forklar forskjellen på arv og komposisjon og hva vi mener med en *is-a* (er-en) og en *has-a* (har-en) relasjon. Beskriv og begrunn også ditt valg av arv eller komposisjon i klassen `SightseeTur`.

b) Tegn opp klassediagrammene for de fire klassene `Posisjon`, `Spor`, `Tur` og `SightseeTur`.

c) Tegn opp datastrukturene for *en* sightsee tur med et kort spor (noen få posisjoner) og to spesielt interessante posisjoner.

Oppgave 2 - 25%

Implementer klassene vi har spesifisert til nå. Du trenger ikke å ta med andre metoder enn `__init__()` siden vi skal jobbe videre med klassene senere. Argumentene til `__init__()` metoden til de ulike klassene er som følger:

- `Posisjon`: `lengdegrad`, `breddegrad`, `moh`, `tidspunkt`
- `Spor`: en liste med *dictionaries*, hvor hver dictionary har følgende nøkler med verdier:
  - `"lengdegrad"`
  - `"breddegrad"`
  - `"moh"`
  - `"tidspunkt"`

  Eksempel på en slik Python liste med *ett* element (tidspunkt er antall sekunder siden Epoch, 1. januar 1970 klokken 00.00.00 UTC):

  ```
  [{"lengdegrad": 18.96778529, "breddegrad": 69.68299052,
    "moh": 64.4, "tidspunkt": 1369651934}]
  ```

- `Tur`: `spor`, `navn`, `beskrivelse` (`beskrivelse` kan være valgfri)
- `SightseeTur`: Samme som `Tur`

Oppgave 3 - 20%

Vi skal nå lage to metoder. Den første metoden returnerer en referanse eller indeks til den posisjonen i et spor som er nærmest tidspunktet angitt som argument:

```
finnPosisjon(self, tidspunkt):
    ...
```

Den andre metoden legger til en ny interessant posisjon i forbindelse med sightseeing. Også her er tidspunktet angitt som argument. Tidspunktet er det tidspunktet man var ved eller på den interessante posisjonen.

```
registrerPOI(self, tidspunkt):
    ...
```

Angi hvilke klasser du vil plassere disse metodene i og skriv koden for metodene.

(Hint: forsøk å utnytte funksjonalitet du allerede har skrevet kode for når det er mulig.)

Oppgave 4 - 15%

På grunn av tidligere erfaringer med bugs vil Ola og Kari benytte enhetstesting (unit testing) under utviklingsfasen.

Forklar kort hva enhetstesting er, og implementer en (enkel) test for en av komponentene i systemet.

Del 2

Oppgave 5 - 10%

Python comprehensions er høyt optimalisert kode som kan brukes for å iterere over mange elementer så raskt som mulig.

Anta at vi har en liste med strenger

```
input_strings = ['2','5','18','10','12','3']
```

Vis med Python comprehensions hvordan man kan konvertere denne listen til en liste med integere *output_ints*. (Hint: svaret kan uttrykkes med én kodelinje.)

Oppgave 6 - 10%

En av de mest brukte mekanismene vi har i objektorientert programmering er *polymorfisme*.

a) Forklar hvordan *arv* kan brukes for å oppnå polymorfisme.

b) Forklar hvordan polymorfisme kan oppnås i Python uten å bruke arv.