Inf-1400 - Objektorientert Programmering – Exam prep 2

John Markus Bjørndalen

2010-05-01

Del 1 av oppgavesett fra Oslo

Dette er en sannsynlig form for en del av eksamen i Tromsø. Oppgaven er delvis omskrevet fra del A av del A av eksamen i INF-1010 objektorientert programmering fra Oslo (3. juni 2009).

En egen Tromsø-versjon av denne oppgaven ble også lagt ut etter forelesningen.

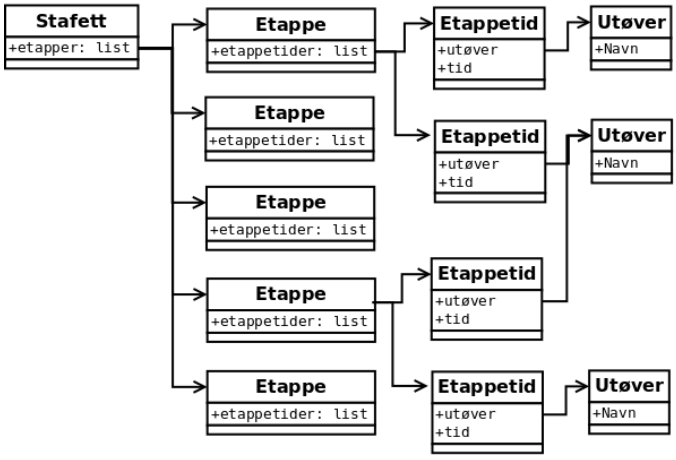
Kildekoden til løsningsforslaget vil også bli lagt ut.

Oppgave 1 - Tegning av datastrukturen

Vekt 5%.

Tegn datastrukturen til et stafettobjekt med 5 etapper, og for hver etappe et antall etappetidobjekter. Tegn også noen utøverobjekter, og gjør det klart hvordan disse er forbundet med de andre objektene i figuren. Legg inn en utøver som går igjen i minst 2 etapper.

Noe tilsvarende følgende figur vil være greit. For sikkerhets skyld kan dere f.eks. ta med noen prikker for å angi at det er flere etappetidsobjekter.



Oppgave 2

```
class Utover:
    def __init__(self, navn):
        self.navn = navn

class Etappetid:
    def __init__(self, utover, etappetid):
        self.utover = utover
        self.etappetid = etappetid

class Etappe:
    def __init__(self, etappenummer, etappetider):
        self.etappenummer = etappenummer
        self.etappetider  = etappetider

class Stafett:
    def __init__(self, etapper):
        # Denne skal være sortert etter etappenummer...
        self.etapper = etapper
```

Oppgave 3

Her kan du også beskrive metodene uten å skrive selve koden (tilsvarende docstrings). Koden er som følger:

```
class StafettLag:
    def __init__(self):
        self.etapper = []
        self.totalTid = 0

    def leggTilEtappe(self, etappetid):
        """Legger til en utøver med en kjent etappetid.
        etappetid er et objekt av typen Etappetid.
        """
        self.etapper.append(etappetid)
        self.totalTid += etappetid.etappetid

    def utoverErPaaLaget(self, utover):
        "Returnerer True hvis utøveren er med på laget allerede"
        return utover in [e.utover for e in self.etapper]

    def copy(self):
        "nyttig for å lage en kopi av dette objektet"
        return copy.deepcopy(self)
```

Oppgave 4

```
def stafettid(self, lag, etapperIgjen):
    """
    lag er laget vi har nå.
    etapperIgjen er en liste over etappene vi ikke har fylt ut enda.
    """
```

```
    if len(etapperIgjen) == 0:
        # Vi har et mulig lag, prøv å legge det til
        self.nyttMuligStafettlag(lag)
        return

    denne_etappen = etapperIgjen[0]
    neste_etapper = etapperIgjen[1:]

    for etid in denne_etappen.etappetider:
        if lag.utoverErPaaLaget(etid.utover):
            continue # ikke bruk samme utøver igjen

        # Lag et nytt lag-objekt som vi legger til denne etappetiden i.
        # Deretter kaller vi stafettid igjen for å sjekke neste etappe.
        nytt_lag = lag.copy()
        nytt_lag.leggTilEtappe(etid)
        self.stafettid(nytt_lag, neste_etapper)
```

Oppgave 5

```
def nyttMuligStafettlag(self, lag):
    """Sjekker om det nye laget er bedre enn det beste vi har sett til nå.
    Lagrer i så fall det nye laget som beste lag.
    """
    if self._besteLag == None:
        self._besteLag = lag
        return

    if lag.totalTid  < self._besteLag.totalTid:
        self._besteLag = lag
```

Oppgave 6

```
def finnBesteStafettLag(self):
    self._besteLag = None
    self.stafettid(StafettLag(), copy.copy(self.etapper))
    print "Beste stafettlag er:"
    for e in self._besteLag.etapper:
        print "  ", e.utover.navn, e.etappetid
    print "Total tid er:", self._besteLag.totalTid
```