IT2 - Oppgave 4 fra tentamen H23

I denne oppgaven skal du lage et program som holder oversikt over fotballspillere, lag og serier.

4.1 - Modell og oppsett

1. Les gjennom oppgavetesktene i oppgave 4.2, 4.3 og 4.4
2. Tegn et UML-diagram med klassene og metodene til programmet.
    - Bruk diagrams.net eller tegn for hånd.

4.2 - Klassen Spiller

Skriv klassen `Spiller` i filen `spiller.py` med følgende grensesnitt.

Klassen skal ha en konstruktør med parameter for `navn` (i tillegg til `self`). Konstruktøren skal opprette instansvariablene `navn`, som får verdi fra parameteren `navn`, og `antall_kamper` som skal settes til heltallet 0. I tillegg til konstruktøren skal klassen ha følgende metoder:

- `hent_navn` som returnerer `navn`
- `hent_antall_kamper` som returnerer `antall_kamper`
- `spill_kamp` som øker `antall_kamper` med 1
- `sjekk_navn` som tar inn et navn som parameter, og returnerer `True` hvis spillerens navn inneholder ett av navnene.

  For eksempel for spillernavnet "Caroline Graham Hansen" skal `sjekk_navn("Caroline")`, `sjekk_navn("Hansen")` og `sjekk_navn("Caroline Jenssen")` alle returnere `True`, mens `sjekk_navn("Karoline")` skal returnere `False`.

4.3 - Klassen Lag

Skriv klassen `Lag` i filen `lag.py` med følgende grensesnitt.

Klassen skal ha en konstruktør med parameter for `navn` (i tillegg til `self`). Konstruktøren skal opprette instansvariablene `navn`, som får verdi fra parameteren `navn`, `spillere` som er en tom liste, `seier` som skal settes til heltallet 0, `uavgjort` som skal settes til heltallet 0 og `tap` som skal settes til heltallet 0. I tillegg til konstruktøren skal klassen ha følgende metoder:

- `hent_navn` som returnerer `navn`
- `hent_poeng` som returnerer antall poeng laget har, hver seier gir 3 poeng, uavgjort gir 1 poeng og tap gir 0 poeng.

For eksempel skal `hent_poeng()` returnere 4 for et lag med 1 seier, 1 uavgjort og 2 tap.

- `legg_til_spiller` som tar inn en spiller som parameter og legger den til i listen `spillere`
- `spill_kamp` som øker antall kamper for alle spillerne på laget med 1
- `seier` som øker `seier` med 1 og kjører metoden `spill_kamp` (slik: `self.spill_kamp()`)
- `uavgjort` som øker `uavgjort` med 1 og kjører metoden `spill_kamp`
- `tap` som øker `tap` med 1 og kjører metoden `spill_kamp`
- `finn_spiller` som tar inn et navn som parameter, hvis en spiller på laget har navnet i navnet sitt, returnerer den spilleren, ellers returnerer den `None` (Samme regler gjelder for sjekk av navn som i metoden `sjekk_navn` i klassen `Spiller`).

4.4 - Klassen Serie

Skriv klassen `Serie` i filen `serie.py` med følgende grensesnitt.

Klassen skal ha en konstruktør med parameter for `navn` (i tillegg til `self`). Konstruktøren skal opprette instansvariablene `navn`, som får verdi fra parameteren `navn` og `lag` som er en tom liste. I tillegg til konstruktøren skal klassen ha følgende metoder:

- `hent_navn` som returnerer `navn`
- `hent_lagliste` som returnerer `lag`.
- `legg_til_lag` som tar inn et lag som parameter og legger den til i listen `lag`
- `spill_kamp` som tar inn parameterne `hjemmelag` som er et lag-objekt, `bortelag` som er et lag-objekt, `bortemål`som er et heltall og `hjemmemål` som er et heltall, hvis et lag har flere poeng enn det andre, skal laget med flest mål vinne og det andre laget tape, hvis lagene har like mange mål, skal begge lagene spille uavgjort.
- `finn_spiller` som tar inn et navn som parameter, hvis en spiller i serien har navnet i navnet sitt, returnerer den spilleren, ellers returnerer den `None` (Samme regler gjelder for sjekk av navn som i metoden `sjekk_navn` i klassen `Spiller`).