Spec av P-uppgift: ELO för League

Skriven av: Kyle Boström Balthazar
Grupp: Basilika

Beskrivning:
Programmet ska kunna simulera (väldigt simpelt) en match av league (tänk 5 manna schackspel). Två lag skapas med 10 spelare totalt och dessa spelare har en elo/rating som då avgör vilket lag de blir tilldelade i för att göra det "jämnt i deras rating".
Användaren kommer kunna lägga till fler och fler spelare med olika ratings och kommer du behöva matcha rätt spelare för att få det så jämnt som möjligt i matchen. Matchinformation kommer att skrivas ut på fil.

Klasser:

Klassen Player som representerar spelarna

Attribut:
- name — namnet som spelaren ska ha
- elo — vilken elo som denna spelare börjar med

Metoder:
- def __init__(self, name, elo):
- def change_elo(self, elo):
  - kan manuellt ändra elo
- def assign_team(self, team):
  - tillsätter lag till spelare manuellt

Datastrukturer:
Varje spelare representeras som ett Player objekt samt att lagen representeras som listor med spelarna.
Spelarna lagras i dictionaries för att kunna spara elo-värdet som väger lagen.
Spelarna läggs till i lagen baserat på värdet.
Resultatet av vinnaren kommer att skrivas ner på fil och leder till en ändring i elo för spelarna i respektive lag.

team1	team2
“Yubuy”: 1100	“TheShy”: 1400
“Stansii”: 1300	“Weiwei”: 1300
“Skagent”: 1400	“Xiaohu”: 1200
“Bui”: 1300	“Light”: 1300
“DyrUpt”: 1300	“Crisp”: 1200

Funktioner:

```
def menu():
    """Skriver ut menyn:
        - Skapa ny spelare
        -

def create_player(name, elo):
    """Indata för spelaren"""

def create_team(name):
    """skapar lag namnet"""

def assign_player(player, team):
    """lägger till en spelare i ett lag, används bara för manuellt tilläggning typ"""

def add_manual:player(players):
    """får ha en input för namn och elo som då kan läggas till i dictionary"""

def sort_players(players):
    """sorterar bara dictionary för att göra det lättare att tilldela lagen"""
```

```
def distribute_players(sorted_players, team1, team2):
    """lägger till vår lista av spelare i lagen
    får fixa så att de två lagen har en balanserad uppställning av spelare"""

def simulate_match(team1, team2):
    """simulerar match mellan två lag
    får skapa någon slags algoritm för att värdera vilket lag som vinner/förlorar
    UTDATAN kommer att innehålla matchresultat, laguppställning osv."""

def main():
    """här finns vår dictionary av spelare och alla anrop av funktionerna"""
```

Algorithm:

1. Skapa spelare och deras ELO:
   - Antingen hårdkodat i källkoden eller lägga till manuellt med "add_manual_player()".

2. Sortera spelarna baserat på ELO-poäng:
   - Använda "sort_players_by_elo()" för att sortera spelarna i fallande ordning efter deras ELO-poäng.

3. Skapa lag med jämn fördelning av ELO-poäng:
   - Dela upp spelarna i två lag ("team1" och "team2") genom "distribute_players_to_teams()"
   - Programmet ska skapa så jämna lag som möjligt,

4. Simulera en match mellan de två lagen:
   - Använd "simulate_match()" för att utföra match simuleringen.
   - Efter matchen kommer spelarnas elo att ändras beroende på resultatet, får se vilka ändringar jag gör för att avgöra förändringar i elo.

5. Visa matchresultatet:
   - Matchresultatet, statistik och lite annat kommer troligtvis att skrivas ut på fil

Tidsplan:

Vecka 49: Grundläggande struktur

- Skapa grundläggande klasser för spelare och lag.
- Implementera metoder och grundläggande funktioner för att skapa, tilldela och hantera spelare och lag.

Vecka 50 - Vecka 3: Match simulering, Utskrift och användarvänlighet

- Utveckla och förbättra match simuleringsprocessen.
- Lägg till variation och realistiska element för att skapa en mer levande och slumpmässig match simulering.
- Testa och finjustera noggrant för att säkerställa korrekt funktionalitet.
- Förbättra utskrift av informationen för att göra den tydlig och lättförståelig.
- Om tiden tillåter det, börja arbeta med att implementera ett grafiskt användargränssnitt (GUI) för ökad användarvänlighet.

Vecka 4 - 5: Finslipning och testning

- Dedikera denna vecka till finslipning av programmet.
- Genomför omfattande tester för att identifiera och korrigera buggar eller otydligheter.
- Fokusera på att optimera användbarheten och stabiliteten hos programmet.

* Väldigt preliminär tidsplan

**Vissa saker kan ta längre tid eller att saker läggs till vilket betyder att perioden vecka 50 - 3 är väldigt flexibel