Jag valde att modellera en simplifierad variant av spelet Football Manager, vilket innehåller en fotbollsliga (League) med fotbollslag (Team) som alla kan ha 0 till många spelare (Player) samt ingen eller en tränare (Coach).

Man skulle kunna argumentera för att Ligan kan existera utan Football_Manager klassen och därav även ha aggregation koppling mellan de två. Dock så har jag valt komposition här på grund av att League objektet skapas i Football_Managers konstruktor och då inte kan existera utan att Football_Manager klassen gör det.

Något av det svåraste under projektet tyckte jag var att definiera vilken koppling klasserna skulle ha av komposition eller aggregation, då jag har svårt att tänka bort kod funktionen.
I Klassdiagrammet jag bifogade har jag försökt att bortse från koden, och fast att man kan se att spelarna endast “existerar” i Team klassens __players list har jag valt att använda mig av aggregation då spelaren trots allt existerar även om det inte tillhör ett lag.
Samma tanke har tillämpats på League <>-- Team.

Fotball_Manager klassen har en hel del metoder, vilket är något som kan ses som redundant, Ex: remove_coach() kallar på remove_coach() i Team.
Anledningen till att jag valde att bryta ut detta i funktioner istället för att kalla på team.remove_coach() direkt, var för att jag hanterar värdet det får tillbaka från funktionerna och tyckte att menyn blev svårare att begripa med en massa if satser inuti.