# IntroLab coupling table

Dependency Depth er lavet ud fra hvor mange lag en klasse skal gå igennem for at få satisfied alle dens dependencies. GameStateManager dependency depth har jeg defineret til 4 for eksempel. Dette vises på nedenstående graf. Hvor det kan ses at der højest er 4 lag. For eksempel GameStateManager -> Playstate -> Enemy -> Game -> GameInputProcessor. Det skal dog nævnes at der er rigtige mange loops i dette program. Altså hvor en klasse refererer til en klasse som refererer tilbage til den samme klasse. Jeg har derfor valgt kun at tælle lag hvor den refererede klasse ikke allerede er reffereret.

A close up of a map

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| NUMBER | CLASS | DEPENDS ON | DEPENDENCY DEPTH |
| 1 | GameKeys | -- | 0 |
| 2 | GameState | GameStateManager | 3 |
| 3 | GameStateManager | GameState, PlayState | 4 |
| 4 | GameInputProcessor | LibGDX, GameKeys | 1 |
| 5 | Game | LibGDX, GameKeys, GameStateManager, GameInputProcessor | 4 |
| 6 | Main | LibGDX, Game | 5 |
| 7 | SpaceObject | - |  |
| 8 | Player | SpaceObject, Game, LibGDX | 5 |
| 9 | Enemy | SpaceObject, Game, LibGDX, | 5 |
| 10 | Bullet | SpaceObject, Game, LibGDX | 5 |
| 11 | PlayState | GameStateManager, GameKeys, Player, Enemy, Bullet, LibGDX, GameState | 4 |

# NetBeansLab2 coupling table

Jeg vil så vidt muligt forsøge at tage udgangspunkt i de samme klasser som i introlabben, hvis de fortsat er en klasse i dette projekt. Alle klasser vil ikke blive gennemgået.

|  |  |  |  |
| --- | --- | --- | --- |
| NUMBER | CLASS | DEPENDS ON | DEPENDENCY DEPTH |
| 1 | GameKeys | -- | 0 |
| 5 | Game | LibGDX, Java.util, org.openide.util, Entity + andre klasser fra common. | 1 |
| 7 | Entity | Java.util | 1 |
| 8 | Player | Entity | 1 |
| 9 | Enemy | Entity | 1 |
| 10 | Bullet | Entity | 1 |
|  | EnemyControlSystem | Entity + andre klasser fra common, java.util, org.openide | 1 |

Eksempel på Dependency graf på Game:

A picture containing clock

Description automatically generated

# Refleksion

Komponent orienteret design gør dependency grafen meget lavere (mange færre lag). I dette eksempel bliver mængden af dependencies også reduceret voldsomt. Det er nødvendigvis ikke tilfældet for alle systemer som omskrives til et komponent-baseret system, men oftest vil det både give mange færre lag og færre dependencies.

Det betyder naturligvis også at **kobling** og **samhørighed** (**coupling** & **cohesion**) bliver væsentligt bedre. Der er lavere kobling mellem modulerne som det kan ses i dette eksempel og langt større samhørighed. Et modul er netop ment til at skulle kunne genbruges og være isoleret. Det betyder altså at den kan stå på egne ben langt lettere, uden et behov for ekstern kode.