2020

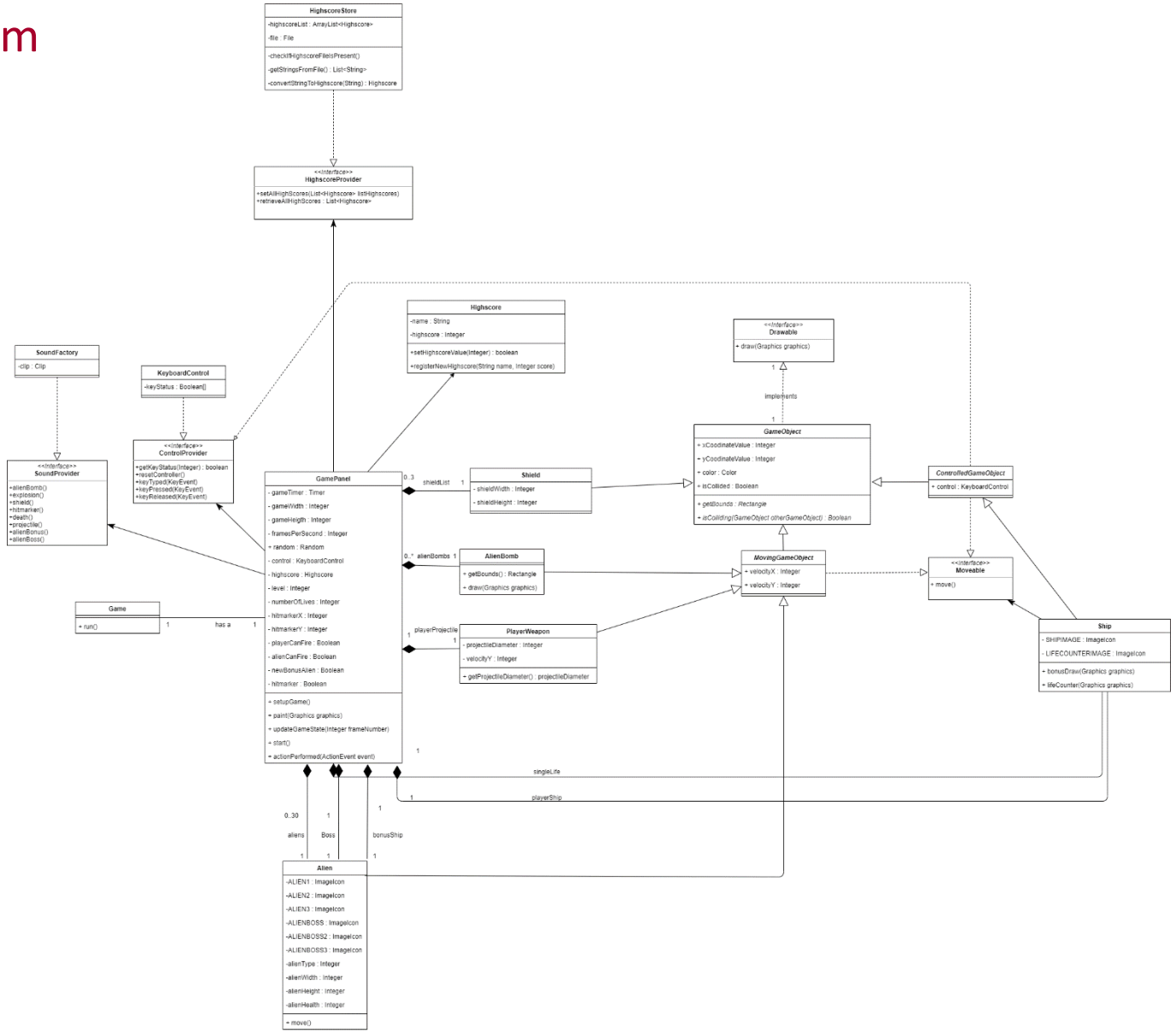
Space Invaders

SPACE INVADERS

BASTIAAN CLEMENT

Inhoudsopgave

Klassendiagram



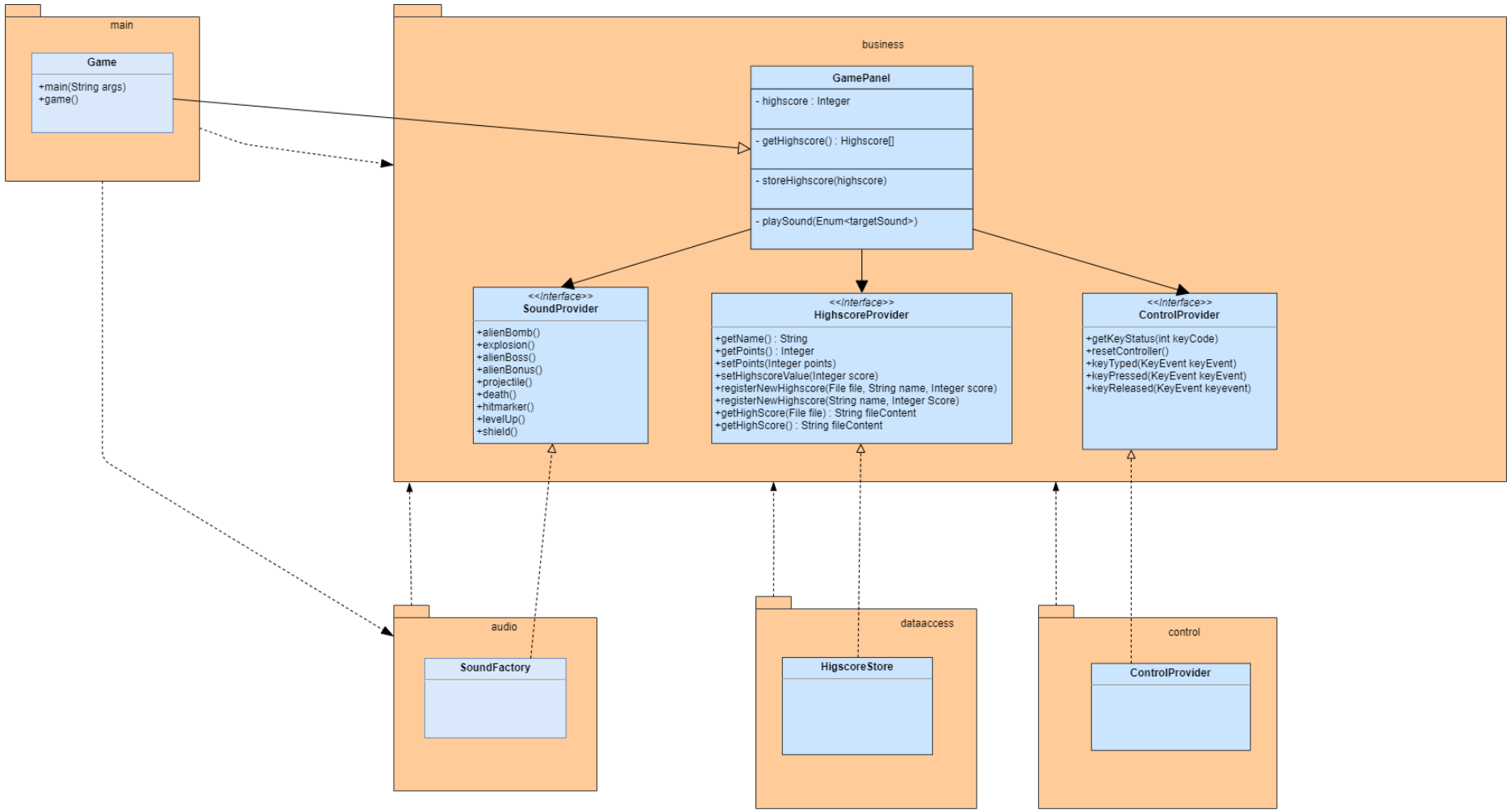
Toelichting klassendiagram

In het boventstaande diagram wordt een overzicht weergegeven van de geplande object structuur van de applicatie Space Invaders. Zo is ervoor gekomen om bij het aanroepen van een methode uit de SoundFactory om geen terugkoppeling te geven en alleen het geluid af te spelen. Een optie zou zijn geweest om een methode te implementeren die als parameter een object als geluid accepteert en deze afspeelt. Dit zou mooier zijn geweest maar daar is destijds helaas niet voor gekozen. Ook de late oplevering en de grootte van het programma is er weinig tijd meer overgebleven om een soortgelijk iets te implementeren.

Gezien de tijdsdruk is de focus te komen liggen op de highscore. Eerst werd deze door de klasse GamePanel rechtstreeks aangemaakt en een string werd opgeslagen met de naam en de score.
Na feedback is de highscore omgezet naar een object en zijn alle bewerkingen met betrekking tot de highscore ook in de klas Highscore te vinden, single responsibility.
Ook de klas HighscoreStore leest alleen het bestand in en uit en zet het om van een highscore met naam om naar een object of een string wanneer deze weer opgeslagen moet worden.

Er zijn nog veel zaken die geoptimaliseerd kunnen worden en de aanpak had beter gekund. Helaas is er geen tijd meer om de hele code te herzien.
Helaas heeft de klasse GamePanel teveel logica in zich.

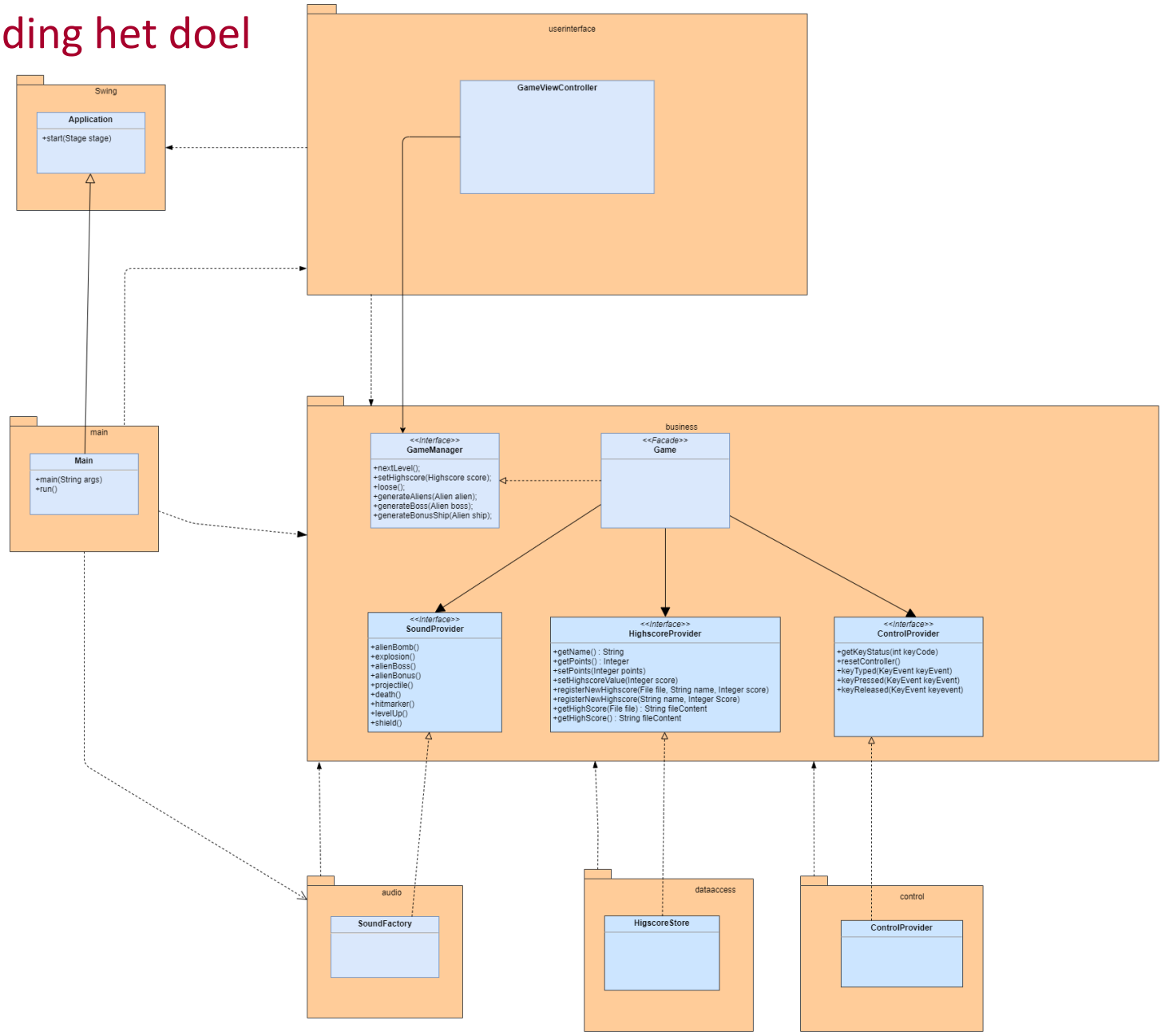
LagenArchitectuur



Lagenarchitectuur huidig programma

Dit is een overzicht van hoe de lagenscheiding momenteel is in de huidige staat van het programma.
Helaas is er te weinig tijd om dit verder uit te werken zodat er een betere tekening ontstaat.
In het volgende hoofdstuk heb ik een fictieve tekening gemaakt hoe ik het graag had willen hebben met de daarbij behorende uitleg.

Lagenscheiding het doel



Het doel was om de lagen te splitsen in 3+1 lagen.

Business laag

Deze laag bevat de *kern* van de applicatie. De verschillende objecten, hun relatie met elkaar, de methodes & verificaties, etc. Om er voor te zorgen dat deze business-laag onafhankelijk staat van de overige lagen, is ervoor gekozen om in deze laag interfaces te definiëren.

De HighscoreProvider is een interface die door de data-laag geïmplementeerd kan worden om een object met data uit een bestand te halen. Omdat deze via een interface loopt kan er eenvoudig gewisseld worden. Of bijvoorbeeld een Mockdatabase bouwen zodat de business laag onafhankelijk getest kan worden. Doordat het Game object géén informatie nodig heeft van de datal-aag, over hoe deze een object opgebouwd heeft of waar dit object uit bestaat(anders dan wat in de interface gedefinieerd is) heeft de logic laag geen dependency op de dataaccess-laag.

De tweede interface die deze laag definieert is de GameManager interface. Om te zorgen dat de userinterface weet welke methodes aan te roepen zijn, fungeert de klas(in het geval van deze applicatie “Game”) als een soort van façade klasse die zorgt dat alle aangeroepen methodes bij de juiste afgeleide klassen terecht komen.

Dataaccess

De dataaccess-laag bevat de bron met informatie(in dit geval de highscore(s)) en de objecten die de mogelijkheid hebben om deze informatie naar een voor de applicatie leesbare vorm om te zetten.
Om dit mogelijk te maken implementeert deze de HighscoreStore klas een interface uit de business laag.

Om de highscore(s) op te slaan is ervoor gekozen om dit in een tekst bestand op te slaan.

Userinterface

De userinterface-laag of presentative-laag bevat de informatie om gegevens om te zetten naar grafische objecten en deze op het scherm tonen waar mee interactie kan plaatsvinden. Om dit te doen heeft het Controller object een object van het type GameManager nodig. In deze userinterface is er voor gekomen om Java Swing voor de weergave te gebruiken. Hierdoor heeft deze module wel een afhankelijkheid met Java Swing.

Main

De Main module wordt gebruikt om de informatie van de ene naar de andere laag door te zetten. Op deze manier kan er vanuit de main module een HighscoreStore aangemaakt worden uit de dataaccess-laag. Deze in de business-laag te gebruiken door middel van de GameBuilder factory en het game object.

De Main klas is een afgeleide klas / extends de JFrame klas. Hierdoor heeft deze laag een afhankelijkheid op alle 3 de modules + java Swing. Om te zorgen dat bij wijzigingen van één van de overige lagen zo min mogelijk informatie veranderd, is er geprobeerd deze laag zo beperkt mogelijk te houden.

Swing(Java)

Voor de interface(gui) maakt de applicatie gebruik van de Swing module. Dit betekend dat de Main class van de JFrame class extends. Hierdoor hebben zowel de userinterface module als de Main module een afhankelijkheid op Swing.

Highscore.dat

Voor de highscore(s) is een bestand of wordt er een bestand aangemaakt. Wanneer het bestand niet aanwezig is wordt deze aangemaakt. Het is zo opgezet dat er met een kleine aanpassing meerdere highscores uitgelezen en opgeslagen kunnen worden. In de toekomst is deze bijvoorbeeld uit te breiden

met datum en tijd, aantal pogingen, etc. Het wordt weg geschreven als een string bestaande uit *naam* gevolgd door een : en de *score*. Een voorbeeld kan zijn *Bertje:89000*. Om te voorkomen dat een gebruiker valsspeelt en een naam opgeeft dat als een : bevat worden alle niet alfanumerieke tekens gefilterd.

Complexiteit

Een lastig scenario in het spel is het bijhouden wanneer welke alien bijvoorbeeld geschoten heeft, waar de *bom* is en of hij iets heeft geraakt. Je wil tenslotte dat een alien maar 1 keer schiet en niet dat alle aliens een regen van bommen laten vallen.

Daarnaast kan het gebeuren dat een speler bijvoorbeeld in het figuur hierboven alien **E** raakt en deze moet dan natuurlijk verdwijnen en mag vervolgens niet meer schieten. Maar als die alien net voordat hij geraakt wordt een bom laat vallen dan moet deze nog wel blijven bestaan en niet verdwijnen samen met de alien.

Zoals in het voorbeeld hierboven wil je niet wanneer een alien uit de bovenste rij, bijvoorbeeld alien **B** een bom laat vallen dat hij de alien onder zich(alien **E**) raakt maar wel de bunker onderin of de speler.

Zo moet er ook bijgehouden worden welke alien al een bom heeft laten vallen en welke niet. Voor het speel effect is het fijne dat een alien willekeurig een bom laat vallen en niet precies elke 3 seconden.

Om alles bij te houden worden aliens in een array gestopt en een aparte array voor de bommen.

Een bunker kan geraakt worden door aliens maar ook door de speler. Een bunker bestaat uit 3 lagen en elke laag kan 3 keer geraakt worden voordat deze verdwijnt. Bij elk schot wat de bunker raakt verandert die betreffende laag van kleur waardoor het visueel zichtbaar wordt wat de status is van de bunker.

Om het spel interessant te houden komt er elke derde level een eindbaas. Deze begint met 30 levens en elk volgende keer dat deze verschijnt wordt het aantal levels vermenigvuldigt met het level. De eind baas is wel groter maar vuurt ook 3 bommen tegelijk af. De valsnelheid van de bommen is gekoppeld aan het level. De bom arraylist wordt hiervoor ook gebruikt om de 3 bommen bij te houden en wanneer alle 3 van het scherm verdwenen zijn mag de eind baas een nieuwe poging ondernemen om de bommen te laten vallen.