# Overzicht lessen PROG

Week 1

1. Herhaling : Functions, Classes en Arrays
2. Herhaling : Action event

Week 2

1. Debugging, Breakpoints
2. DRY & SRP

Week 3

1. OOP: Inheritance
2. OOP: Abstraction (abstract class, override)

Week 4

1. OOP: Encapsulation (getter, setters)
2. OOP: Polymorphism (interfaces)

Week 5

1. Class diagrams
2. Design pattern – Singleton

Week 6

1. Design pattern – Factory Method

# Lesrooster aandachtspunten

SD2B 1e les woensdag blok 1:

* evt studenten te laat door trein
* Studenten nog fris
* +-20 studenten

SD2A 1e les woensdag blok 2:

* Studenten hebben in blok 1 aan project gewerkt, even schakelen dus
* +-20 studenten

SD2A 2e les vrijdag blok 1:

* evt studenten te laat door trein
* Studenten nog fris
* +-20 studenten

SD2B 2e les woensdag blok 2:

* Studenten hebben in blok 1 aan wiskunde gewerkt, even schakelen dus
* +- 20 studenten

# Lesplan les 1 : Herhaling : Functions, Classes en Arrays

## Doelen:

De studenten kunnen **functies** schrijven, **argumenten** aan deze functies meegeven en de functies ook weer data terug laten geven(**return**).

De studenten weten dat een **Method** ook een functie is die voor een object kan worden uitgevoerd.

De studenten weten wat een **class** is en dat hiermee objecten geïnstantieerd kunnen worden.

De studenten kunnen zelf een eigen **class** maken en hiervoor ook een eigen **constructor** maken en aanroepen.

De studenten weten hoe ze een **Array** en een **List** kunnen maken en wat het verschil is tussen de Array en de List.

## Situatie:

De studenten hebben deze onderwerpen vorig schooljaar al behandeld in de PROG lessen. Echter zal bij velen deze informatie niet goed zijn blijven hangen..

## Toetsing:

d.m.v verschillende praktijkopdrachten in de les kunnen de studenten laten zien dat ze de onderdelen nog beheersen.

* **functie** opdracht
* **class** opdracht
* **array/List** opdracht

## Vervolgacties:

Studenten die kennis van onderdelen nog missen in kaart brengen. Studenten leven de opdracht in via simulise

Studenten die kennis van onderdelen nog missen uitleg en oefenmateriaal meegeven.

Volgende les checken of oefeningen zijn gedaan en of de stof is begrepen.

# Lesplan les 1 : Herhaling : Functions, Classes en Arrays

|  |  |  |
| --- | --- | --- |
| tijd | activiteit | materiaal |
| 00:05 – 00:10 | Landen/Energizer | Kahoot basic programming: <https://create.kahoot.it/details/96f41b13-37bd-47b8-9953-611bfa8e96fe> |
| 00:10 – 00:20 | Orientatie vragen stellen: functies, classes, arrays | Vragen om uitleg van willekeurige student over een functie. Wat is een functie? Waarom gebruik je ze? Wat is het verschil tussen een function en een Method? Wat is een argument? Wat is een return value?  Vragen om uitleg van willekeurige student over een class. Wat is een class? Waarom gebruik je deze? Wat is een constructor?  Vragen om uitleg van willekeurige student over een Array. Wat is een Array? Waarom gebruik je deze? Wat is het verschil tussen een Array en List? |
| 00:20 – 00:40 | Uitleg geven over functions, methods arguments en return values | Digitaal materiaal via readme delen |
| 00:40 – 00:50 | Opdracht uitwerken: Studenten maken zelf een **method** met **arguments** en **return value** | Opdracht via simulise(>github) delen |
| 00:50 – 00:55 | terugkijken | willekeurige student laat zijn functie zien op de beamer. |
| 00:55 – 01:50 | Doornemen uitleg video‘s over classes en uitwerken bijhorende opdracht keuze uit classes of Arrays | Uitleg video en opdracht over classes en arrays ook posten op simulise (>github) |
| 01:50 -02:00 | Afronding: vragen hoe ver ze zijn met de opdrachten. Willekeurige student laat zijn opdracht (**Classes** / **Array**) op de beamer zien.  Student vragen hoe deze ervoor kan zorgen dat hij deze stof nu kan blijven oefenen? Om er beter in te worden. |  |
| Huiswerk | Afmaken opdrachten **classes** en **arrays**, volgende les vragen stellen |  |

# Lesplan les 2: Action events

## Doelen:

Verifiëren of iedereen **functions**, **classes** en **Arrays** begrijpt en of er nog vragen zijn.

De studenten kunnen een **Action Event** gebruiken om functies tussen verschillende scripts te “triggeren” en daarbij eventueel data mee te geven.

## Situatie:

Sommige studenten hebben vorig jaar al een video gevolgd over **Action Events**.

**Vorige** **les** hebben we aandacht besteed aan het herhalen van **functions**, **classes** en **arrays**. De kans is aanwezig dat hier nog veel onduidelijkheid over bestaat en dat dit dus nog aandacht nodig heeft.

## Toetsing:

d.m.v verschillende praktijkopdrachten in de les kunnen de studenten laten zien dat ze de onderdelen nog beheersen.

## Vervolgacties:

Studenten die kennis van onderdelen nog missen in kaart brengen.

Studenten die kennis van onderdelen nog missen uitleg en oefen materiaal meegeven.

Volgende les checken of oefeningen zijn gedaan en of de stof is begrepen.

# Lesplan les 2: Action events

|  |  |  |
| --- | --- | --- |
| tijd | activiteit | materiaal |
| 00:05 – 00:10 | Landen/Energizer | Action Event Spel met briefjes en trigger woorden, alle studenten krijgen de op dracht om op te springen en iets over zichzelf te vertellen wat de rest nog niet weet als er een woord wordt genoemd wat op zijn briefjes staat.(action event) |
| 00:10 – 00:20 | Orientatie vragen stellen over: Action Events | Vragen aan de studenten of ze begrijpen waar het spel over ging? Vragen wie er weet wat een action event is. Vragen om een korte uitleg van een student. |
| 00:20 – 00:40 | Keuze maken: Video kijken over Action Events of Artikel lezen voor mensen die al een Action Event kunnen gebruiken(**verdiepen**) | Digitaal materiaal via Simulise>readme delen <https://youtu.be/IKBg97A7Sbw?si=VZhFt4WQyNUpB3aF>  of  <https://gamedevbeginner.com/events-and-delegates-in-unity/> |
| 00:40 – 00:60 | Opdracht uitwerken: passen zelf een Action event toe in een clicker game met een scorebord. Als er 100 punten zijn moet er een animatie of effect worden uitgevoerd dmv Action Event. Inleveren via simulise | Opdracht via simulise(>github) delen |
| 01:05 – 01:10 | Terugkijken op Action Events | willekeurige student laat zijn resultaat zien op de beamer. |
| 01:10 – 01:20 | Terugkijken op **functions** door willekeurige student een recap te laten geven op **functions** |  |
| 01:20 -01:30 | Terugkijken op **classes** door willekeurige student een recap te laten geven op **classes** |  |
| 01:30 -01:40 | Terugkijken op **Arrays** door willekeurige student een recap te laten geven op **Arrays** |  |
|  |  |  |
|  |  |  |

# Lesplan les 3: Debugging & Breakpoints

## Doelen:

Checken of alle opdrachten zijn gemaakt. Is iedereen op schema? Toetsen of iedereen de basis beheerst.(**functies, classes, arrays, action events**)

De studenten begrijpen waarom breakpoints gebruiken in veel gevallen veel sneller kan zijn dan Debug.Log commands invoeren

De studenten kunnen debuggen via **breakpoints** in MS Visual studio, hierbij zijn ze specifiek in staat om na een breakpoint lijn voor lijn door de code heen te stappen en alle data te inspecteren.

## Situatie:

De studenten zijn bekend met het Debug.Log commando en hebben dat in het Mythe project vaak moeten gebruiken om te debuggen..

## Toetsing:

d.m.v verschillende praktijkopdrachten in de les kunnen de studenten laten zien dat ze de onderdelen nog beheersen.

## Vervolgacties:

Studenten die kennis van onderdelen nog missen in kaart brengen.

Studenten die kennis van onderdelen nog missen uitleg en oefen materiaal meegeven.

Volgende les checken of oefeningen zijn gedaan en of de stof is begrepen.

# Lesplan les 3: Debugging & Breakpoints

|  |  |  |
| --- | --- | --- |
| tijd | activiteit | materiaal |
| 00:05 – 00:15 | Energizer: 10 verstopte bugs zoeken on het lokaal | Papieren Bugs verstoppen, elke bug heeft een prijs (kitkat) |
| 00:15 – 00:20 | Oriëntatie vragen stellen over: Debuggen | Vragen aan studenten hoe ze bij Mythe bugs hebben opgespoord. Wie heeft er al eens gewerkt met breakpoints? |
| 00:20 – 00:40 | Uitleg over debuggen | Miro les <https://miro.com/app/board/uXjVNeHQqLI=/> |
| 00:40 – 1:20 | Debug opdracht uitvoeren | Projectje met een bug erin als opdracht bereikbaar en in te leveren via simulize |
| 01:20 – 01:30 | Terugkijken: wie heeft de bug gevonden? Wie heeft de bug opgelost? evt laten zien op de beamer. Evt zelf oplossing tonen | Oplossing van bug |
| 01:30 – 02:00 | TD issues bespreken en ondersteuning in TD code. Studenten zonder vragen mogen gaan. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Lesplan les 4: SRP & DRY

## Doelen:

De studenten weten waar SRP en DRY voor staat.

De studenten begrijpen waarom je SRP en DRY goed toe moet passen in je code.

De studenten herkennen wanneer code niet DRY of SRP geschreven is.

De studenten kunnen een manier bedenken hoe de code DRY en SRP kan worden gemaakt.

## Situatie:

De studenten hebben in p4 van leerjaar 1 allemaal spaghetti code geschreven en zijn veelvuldig tegen structuurproblemen aangelopen. Dit is dus een mooie kapstok om het belang van structuur aan te kaarten.

De meeste studenten kennen de termen SRP en DRY nog niet.

## Toetsing:

d.m.v verschillende praktijkopdrachten in de les kunnen de studenten laten zien dat ze de onderdelen nog beheersen.

## Vervolgacties:

Studenten die kennis van onderdelen nog missen in kaart brengen.

Studenten die kennis van onderdelen nog missen uitleg en oefen materiaal meegeven.

Volgende les checken of oefeningen zijn gedaan en of de stof is begrepen.

# Lesplan les 4: SRP & DRY

|  |  |  |
| --- | --- | --- |
| tijd | activiteit | materiaal |
| 00:05 – 00:15 | Energizer: Ga van links naar rechts staan als je de meeste spaghetti code hebt geschreven tijdens mythe. Zorg voor bewijs hiervan. | Laptops van studenten |
| 00:15 – 00:20 | Oriëntatie vragen stellen over: SRP & DRY  Wat maakte je code spaghetti code? Wat was het gevolg voor je proces?  Waar staat DRY/SRP voor? Waar zou WET voor staan (Write Everything Twice / Waste Everyones Time) |  |
| 00:20 – 00:40 | Uitleg over SRP & DRY | Miro les <https://miro.com/app/board/uXjVMpW9alY=/> |
| 00:40 – 1:40 | SRP & DRY opdracht uitvoeren refactoren code naar SRP & DRY  *Lange opdracht dus even waarschuwen bij 15 min en 7 min over!* | Projectje met non SRP non DRY class als opdracht bereikbaar en in te leveren via simulize |
| 01:40 – 02:00 | Terugkijken: 3 Willekeurige studenten laten hun oplossingen op de beamer zien en lichten dit toe. Ik laat mijn oplossing ook zien. | Gerefactorde versie van de code |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Lesplan les 5: OOP Inheritance

## Doelen:

De studenten weten waar OOP voor staat.

De studenten weten dat OOP bestaat uit 4 basis principes: Inheritance, Abstraction, Encapsulation en Polymorphism

De student is zich bewust dat hij in Unity al vaak overerving heeft gebruikt op de **Monobehaviour** class.

De student weet wat **Inheritance** inhoud en kan zelf een eigen class overerven.

De studenten zijn zelf in staat om technisch onderzoek te doen

## Situatie:

De studenten zijn onbewust bezig geweest met inheritance namelijk op de monobehaviour class. Een mooie kans om ze hier dus bewust van te maken.

De studenten zijn nog niet bekend met OOP wellicht dan een enkeling er een vaag beeld van heeft.

## Toetsing:

d.m.v verschillende praktijkopdrachten in de les kunnen de studenten laten zien dat ze de onderdelen nog beheersen.

## Vervolgacties:

Studenten die kennis van onderdelen nog missen in kaart brengen.

Studenten die kennis van onderdelen nog missen uitleg en oefen materiaal meegeven.

Volgende les checken of oefeningen zijn gedaan en of de stof is begrepen.

# Lesplan les 5: OOP Inheritance

|  |  |  |
| --- | --- | --- |
| tijd | activiteit | materiaal |
| 00:05 – 00:15 | Energizer: plak stickers op 2 objecten in het lokaal.  bedenk per object 3 eigenschappen en functies  langs 10 stickers en vragen wat de eigenschappen en functies van de objecten zijn.. | Stickers die weer makkelijk los te halen zijn |
| 00:15 – 00:20 | Oriëntatie vragen stellen over: OOP  Waar staat OOP voor? Wie heeft er wel eens van gehoord? Wat zijn Objecten eigenlijk? Wat betekend Georiënteerd? |  |
| 00:20 – 01:30 | In groepen (5\*4) uitzoeken wat OOP inhoud en eind van de les presenteren in een PowerPoint.  Waar staat het voor?  Waar bestaat het uit? Onderdelen  Wat is het doel van OOP?  Wat is inheritance?  Hebben jullie al eens inheritance gebruikt?  Geef een code voorbeeld van inheritance.  Werken waar je wil | Tafels in groepsopstelling, studenten met laptops en internet. Powerpoint.(inleveren op simulise)  Groepsopdracht bereikbaar via simulise.  Google, chat gpt, youtube, vragen stellen aan docenten. |
| 01:30 – 02:00 | Presentaties van 5 min per groep (5 groepen van 4) | Beamer / scherm |
|  |  |  |
|  |  |  |