CHAPTER 1

OIS12 leerdoelen

FHICT

Deze leerdoelen komen uit de vakkengids voor het startsemester. Globale doel is dat je alle onderstaande leerdoelen afzonderlijk en gezamenlijk kunt toepassen in een C#-programma dat je schrijft aan de hand van een functionele specificatie. Let op: je moet dit alles uit jezelf kunnen programmeren, alleen code kunnen uitleggen is niet voldoende!

1.1 Leerdoelen VOORSTEL OIS12 cohort201802

Dit zijn de minimale leerdoelen om tot een voldoende indicatie te komen voor het vakgebied.

We beginnen met de centrale technische leerdoelen van OIS12.

constructor

Je kunt in C# vanuit een specificatie constructors programmeren en deze gebruiken.

Bij FUN12 krijg je geleerd hoe je met `classes` moet werken, OIS12 voegt hier aan toe dat je `constructors` gebruikt om `instanties` van een `class` te maken.

private/public

Je weet wat `private` en `public` betekenen en maakt `fields` in een `class` altijd `private`.

Als van buiten een `object` de waarde van een `field` moet wijzigen gaat dit middels hiertoe bestemde `methods` of `properties`. Dit is niet iets wat je 1

keer laat zien maar wat je vanaf dat moment in AL je programma's doet! (dus ook bij FUN12!)

overloading

Je kunt methodes programmeren met dezelfde naam binnen 1 class en je kunt uitleggen wat method overloading betekent.

We gebruiken dit bij OIS12 veel voor constructors (waarbij dit ook kan, net als bij 'gewone' methodes).

property

Je snapt wat een `property` is (zowel *normaal* als *auto property*) en hebt laten zien dat je ze zelf kunt aanmaken.

Het snappen wat een property is is het belangrijkste aangezien veel voorbeelden op internet met `properties` werken: Mensen die bijvoorbeeld al `javakennis` hebben en (ná aantonen dat ze snappen wat een `property` is) voorkeur hebben voor `Get` en `Set methods` mogen dat ook.

List en foreach

Je kunt programmeren met foreach en lijsten van objecten.

Met een lijst wordt hier bedoeld een `List<T>` waarbij T een geldig `type` is, bijvoorbeeld een zelf geprogrammeerde `class`.

Van relation tussen classes naar een List

Je kunt vanuit een gegeven klassendiagram relaties in `C#` programmeren waarbij je gebruik maakt van Lists (lijsten van objecten).

Als een `object` meerdere `objecten` van een `type T` bevat wordt dit gerealiseerd door eerstgenoemd `object` een `Field` te geven van het type `List<T>`.

enum

Je kunt `enums` programmeren.

Dit wil zeggen dat je weet hoe je dit doet, een keer een goed voorbeeld hebt laten zien aan de docoent. Verder hou je je ogen open tijdens het ontwikkelen: *als* je een situatie tegen komt die roept om een `enum` gebruik je die. Enums kunnen een belangrijk wapen zijn om je code leesbaar en onderhoudbaar te houden.

Nu een aantal leerdoelen waarbij je mag laten zien dat je een professional in wording bent.

Professionele houding

Je stelt je professioneel op. Dit blijkt ondermeer uit je aanwezigheid, actieve deelname tijdens de lessen, opbouwend kritische houding, samenwerken met anderen en het houden aan afspraken.

Als je er een keer tijdens lestijden niet bent, pas later aanwezig kunt zijn of eerder weg gaat communiceer je dit naar de docent. Ook als de les langer duurt dan jij zonder roken, gamen of facebook kunt en je dus naar buiten gaat (roken) of naar een open ruimte (gamen/facebook): dat meld je even bij de docent.

Visual Studio

Je kunt je eigen Visual Studio-installatie onderhouden.

Ook dit leerdoel gaat in belangrijke mate over professionaliteit: Zorg dat je je laptop in orde hebt, met visual studio werkend. Natuurlijk kunnen er wel eens problemen zijn, maar je zorgt dat je een backup hebt zodat je geen weken werk kwijt bent. Als je laptop of Visual Studio niet orde zijn weet je binnen afzienbare tijd te regelen dat je toch aan het vak kunt werken.

Analyse

Je kunt een analysedocument schrijven waarbij je voorafgaand aan een project een gepriotiriseerde opsomming geeft van functionele eisen aan een applicatie.

Natuurlijk kan van een startemesterstudent nog geen volledig analysedocument verwacht worden. Wat we echter wel verwachten is dat je requirements verwoordt en hier een MoSCoW-indeling van maakt op een manier die laat zien dat je de klant serieus neemt. Dat wil zeggen dat je aandacht en liefde erin stopt: Verzorgde layout en geen taalfouten. Tot slot verwerk je eventuele feedback op documenten.

Technical design

Je bent in staat om vooraf te ontwerpen uit welke klassen je applicatie bestaat, en wat de eigenschappen, methoden en verantwoordelijkheden van die klassen zijn, de relaties tussen deze classes en dit alles op een grafische manier weer te geven.

Het gaat er niet om dat je (bijvoorbeeld) `UML` ('Unified Modeling Language') volgens alle regels kunt toepassen: het gaat er om dat je voordat je gaat coderen op een (enigszins abstracte) manier over een programma kunt praten en hier een grafische weergave kunt maken. Je zult ook hier mogelijk feedback op krijgen waar je iets mee zult moeten doen.

UI separation

Je bent in staat om kleine applicaties te programmeren waarbij er een goede scheiding is tussen code in klassen en userinterface-code.

Met de kennis en vaardigheden die je nu hebt is deze scheiding nog niet altijd mogelijk maar we streven dit zoveel als mogelijk wel na.

De nu volgende leerdoelen zitten in dit vak omdat je ze logischerwijs bij een ander leerdoel tegenkomt (collateral).

FileHandling

Je moet vanuit een specificatie een C#-programma kunnen schrijven waarmee tekstbestanden kunnen worden ingelezen en worden geschreven.

Exception

Je kunt op een correcte manier excepties in je C#-programma afhandelen.

Wat exceptions betreft gaat hier om het gebruik van de `try..catch` op momenten dat dit van toepassing is. Net zo belangrijk is dat je het *alleen dan* gebruikt!

graphics

Je kunt vanuit een specificatie een programma schrijven dat werkt met de diverse functies van de Windows GDI (graphics).

static

Je kunt gebruik maken van bestaande methoden die static zijn en je kunt uitleggen wat het static keyword betekent.

casting

Je weet wat een `cast` is en kunt `casting` toepassen in het geval een `GUI-control` verwijst naar een of meerdere `objecten` van een bepaald type.

Hét voorbeeld hiervan binnen OIS12 is dat je een `object` uit een `ListBox` haalt waarvan je zeker weet dat het van een specifieke `class` als `type` heeft: Als je een element uit de `Items` haalt kun je de waarde `casten` naar het juiste `type`.

Verder weet je dat `cast` inherent *onveilig* is (zeker in sommige programmeertalen) en dus niet onnodig gedaan moet worden.

Value versus Reference Types

Je kunt het verschil tussen value en reference types uitleggen en er mee programmeren.

Dit is niet een leerdoel dat je expliciet moet aantonen.

XAML

Je kunt user interfaces voor applicaties maken met zowel de Windows Forms designer als met de XAML designer.

1.2 Leerdoelen OIS12 cohort201708

visual studio

Je kunt je eigen Visual Studio-installatie onderhouden. In feite gaat dit leerdoel over professionaliteit: Zorg dat je je laptop in orde hebt, met visual studio draaiend. Natuurlijk kunnen er wel eens problemen zijn, maar je zorgt dat je een backup hebt zodat je geen weken werk kwijt bent. Als je laptop of visual studio niet orde zijn weet je binnen korte tijd te regelen dat je toch aan het vak kunt werken.

analyse

Je kunt een analysedocument schrijven waarbij je voorafgaand aan een project een gepriotiriseerde opsomming geeft van functionele eisen aan een applicatie.

filehandling

Je moet vanuit een specificatie een C#-programma kunnen schrijven waarmee tekstbestanden kunnen worden ingelezen en worden geschreven.

exception

Je moet op een correcte manier excepties in je C#-programma kunnen afhandelen.

UML design

Je bent in staat om vooraf te documenteren uit welke klassen je applicatie bestaat, en wat de eigenschappen, methoden en verantwoordelijkheden van die klassen zijn en deze grafisch in UML weer te geven.

associaties

Je kunt associaties (relaties) met multipliciteiten tussen klassen ontwerpen en deze weergeven in een UML-klassendiagram.

constructor

Je kunt in C# vanuit een specificatie constructoren programmeren en deze gebruiken.

overloading

Je kunt in C# binnen een klasse methoden programmeren met dezelfde naam en je kunt uitleggen wat method overloading betekent.

private/public

Je kunt programmeren met private en public.

property

Je kunt property's programmeren vanuit een gegeven specificatie.

enum

Je kunt enum's programmeren.

graphics

Je kunt vanuit een specificatie een programma schrijven dat werkt met de diverse functies van de Windows GDI (graphics).

static

Je kunt gebruik maken van bestaande methoden die static zijn en je kunt uitleggen wat het static keyword betekent.

casting

Je kunt programmeren met casting in de context van het casten van een object naar het gewenste subtype.

List en foreach

Je kunt programmeren met foreach en lijsten van objecten.

professionele houding

Je stelt je professioneel op. Dit blijkt ondermeer uit je aanwezigheid, actieve deelname tijdens de lessen, een kritische houding en samenwerken met anderen.

Van UML relation naar List

Je kunt vanuit een gegeven UML-klassendiagram relaties in C# programmeren waarbij je gebruik maakt van List's (lijsten van objecten).

Value versus Reference Types

Je kunt het verschil tussen value en reference types uitleggen en er mee programmeren.

UI separation

Je bent in staat om kleine applicaties te programmeren waarbij er een goede scheiding is tussen code in klassen en userinterface-code.

XAML

Je kunt user interfaces voor applicaties maken met zowel de Windows Forms designer als met de XAML designer.