Hogeschool Rotterdam / CMI

Design patterns

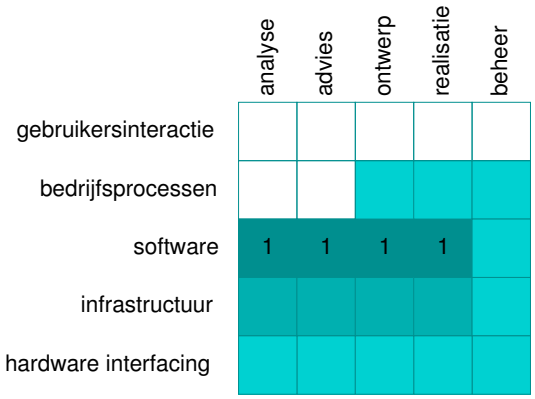
TIRDES01

Aantal studieunten: 3 ects
Modulebeheerder: Wessel Oele

Goedgekeurd door:
(namens toetscommissie)
Datum:

Inhoudsopgave

Modulebeschrijving

Modulenaam:	Design patterns
Modulecode:	TIRDES01
Aantal studiepunten en studiebelastingsuren:	Deze module levert 3 studiepunten op. • 8 × 120 minuten hoorcollege • 8 × 120 minuten practicum • 12 × 120 minuten zelfstudie
Vereiste voorkennis:	Java: basis, o.o.p., toepassen, datastructuren
Werkvorm:	hoorcollege en practicum
Toetsing:	Practicumopdrachten
Leermiddelen:	Design patterns explained, auteur: Alan Shalloway, James R. Trott, uitgever: Addison Wesley, ISBN: 978-0-321-24714-8
Draagt bij aan competentie	
Leerdoelen:	Kunnen werken met U.M.L. Begrijpen en kunnen toepassen van diverse design patterns zoals adapter, facade, bridge, abstract factory, e.v.a.
Inhoud:	• U.M.L. • Diverse design patterns
Opmerkingen:	
Modulebeheerder:	Wessel Oele
Datum:	12 september 2015

1 Algemene omschrijving

Het correct ontwerpen van software is een gecompliceerde zaak. Niet alleen is het ontwerpen zelf niet eenvoudig, moeilijker wordt het wanneer de eisen, waaraan een stuk software moet voldoen ook nog eens veranderen. Ten slotte zijn de meeste computerprogramma' s nooit af en dient er in de jaren na oplevering met enige regelmaat aan onderhoud en uitbreiding gedaan te worden.

Design patterns vormen een hulpmiddel bij het ontwerpen van software, opdat software zo ontworpen kan worden dat het aanpassen en uitbreiden van de software eenvoudiger wordt. Ook vergroot het gebruik van design patterns de herbruikbaarheid van (delen van) de software.

Design patterns vinden hun oorsprong in het einde van de jaren '70 toen Christopher Alexander patronen als hulpmiddel gebruikte bij het ontwerpen van gebouwen en steden. Na enig experimenteel werk van Beck en Cunningham werden design patterns populair in de jaren '90 toen de zogeheten "gang of four" (Erich Gamma, Richard Helm, Ralph Johnson en John Vlissides) het boek *"Design Patterns: Elements of Reusable Object-Oriented Software"* publiceerden.

1.1 Relatie met andere onderwijseenheden

Deze module bouwt voort op de modules tinpro01-1, tinpro01-2, tinpro01-3, en tinpro01-4. Verondersteld wordt het programmeren in een imperatieve en objectgeoriënteerde taal te beheersen.

1.2 Leermiddelen

Verplicht:

- Boek: Design patterns explained, auteur: Alan Shalloway, James R. Trott, uitgever: Addison Wesley, ISBN: 978-0-321-24714-8
- Software: Java Development Kit (JDK) versie 6, te downloaden van `http://www.javasoft.com`
- Presentaties die gebruikt worden in de hoorcolleges (pdf): te vinden op `http://med.hro.nl/oelew`
- Opdrachten, waaraan gewerkt wordt tijdens het practicum (pdf): te vinden op `http://med.hro.nl/oelew`

Facultatief:

- Boek: Design Patterns: Elements of Reusable Object-Oriented Software, auteur: Gamma, Helm, Johnson, Vlissides ,uitgever: Addison-Wesley. ISBN 0-201-63361-2
- Text editors: Emacs, VI, Jedit, Gedit, etc.

2 Programma

Week	Literatuur	Lesinhoud
1	D.P. Explained t/m blz. 45,	introductie, herhaling o.o.p., u.m.l.
2	D.P. Explained t/m blz. 73	o.o.p. problemen
3	D.P. Explained t/m blz. 115	facade en adapter pattern
4	D.P. Explained t/m blz. 136	denkwijze en perspectief
5	D.P. Explained t/m blz. 157	strategy pattern
6	D.P. Explained t/m blz. 191	bridge pattern
7	D.P. Explained t/m blz. 212	abstract factory pattern (inleveren groepsopdrachten bij practicum)
8	D.P. Explained t/m blz. 266	
9		vragenuur/inhaalles
10		vragenuur/inhaalles

3 Toetsing en beoordeling

Twee procedures zijn, afhankelijk van de opdracht en docent, beschikbaar:

3.1 Procedure 1

Deze module wordt getoetst middels groepsopdrachten. Voorwaarden:

- Opdrachten worden op papier en tijdens de practicumlessen ingeleverd.
- Een groep bestaat uit maximaal vier studenten. Deze leveren gezamenlijk één opdracht in.
- Bij het inleveren zijn alle leden van de groep aanwezig, opdat *elk* lid de uitwerking mondeling kan verdedigen.
- De practicumdocent kan naar eigen inzicht (bijvoorbeeld om didactische redenen) afwijken van de practicumopgaven en alternatieve opdrachten aanbieden. Hierbij staat duidelijkheid en integriteit richting de studenten uiteraard voorop.

3.2 Procedure 2

The module is examinated through a series of heavily connected practicum assignments. The assignment feature the building of a simple game in a modern mainstream OO language such as Java or C#. The assignments will begin with an unstructured, mostly procedural game which, through the application of design patterns, will become always more structured. The parts of

the game will become more and more independent from each other, and at the same time more reusable.

Handing in is done through GitHub, with a project with name `MODULE-CODE_STUDENT-NUMBER1_STUDENT-NUMBER2_STUDENT-NUMBER3_STUDENT-NUMBER4`.

Therefore, a group made up of students 123456, 654321, 123654, and 654123 would hand-in a public GitHub project with name:

`TIRDES01_123456_654321_123654_654123`.

At the worst weekly check-ins will be required *from every member of the team*. Even though the course allows group work, delegation is not permitted. Each and every member of the team **must know** how all aspects of the handed-in code work. This will be ascertained through an oral check (**not an oral examination**) where students will be asked to explain random parts of the code, independently from authorship.

The oral check will be done during the last practicum lecture, which is also the final deadline. During the last lecture grades will be given. Retake will happen following the same examination structure: a short oral check based on the handed-in GitHub project. Retake will be done during the exam weeks of the second period.

Assignment 0 - the basics The first assignment requires building a very small game. For the assignment to be sufficient, the game must feature at the very least:

- at least three different entities with different roles within the game (for example spaceships, asteroids, and projectiles)
- input in order to control some entities (for example the keys to move the spaceship and the space key to shoot projectiles)
- interaction between entities (for example contact between projectiles and asteroids causes both to disappear and an explosion to take place)

It is highly recommended to use the combination of C# and the open-source software MonoGame in order to build the game. Similar environments do exist in Java and may be used. For an extra challenge (and therefore extra points), other languages which may be used are F# and Haskell.

Score: 25%

Assignment 1 - adapter/faade for rendering Separate the game logic and the rendering logic through a mixture of faade (the rendering system) and adapters (for the objects to render). The main representation of data is done in the game-logic object, whereas the adapters store additional, rendering-specific data.

Score: 25%

Assignment 2 - factory and abstract factory Factor out the logic for the creation of objects (and their rendering adapters) within a list of abstract factories that determine themselves when and how to create objects for the game logic and rendering faade.

Score: 25%

Assignment 3 - strategy/decorator Build a series of classes that implement the interface:

```
interface Script {
  bool Tick(float dt);
  Script Reset();
}
```

Concrete implementations of the class will be, at the very least:

- the `Wait` class, which `Tick` method returns `true` only after a certain amount of time
- the `WaitKeyPress` class, which `Tick` method returns `true` only after a certain key is pressed
- the `Sequentialize` class, which `Tick` method returns `true` only after all its internal `Script` instances have returned `true`, in the proper storage order
- the `Repeat` class, which `Tick` method always returns `false` but, whenever the internal `Script` is done (its `Tick` method returns `true`), invokes `Reset` on the internal `Script` to start it again

Score: 25%

4 Bijlage 1: Toetsmatrijs

	Leerdoelen	Dublin descriptoren	Verwijzing naar opdracht / vraag / criteria
1	o.o.p., u.m.l.	1,2,3,4	practicumopdracht 1
2	adapter, facade	1,2,3,4	practicumopdracht 2
3	abstract factory	1,2,3,4	practicumopdracht 3
4	strategy/decorator	1,2,3,4	practicumopdracht 4

Dublin-descriptoren:

1. Kennis en inzicht
2. Toepassen kennis en inzicht
3. Oordeelsvorming
4. Communicatie

5 Bijlage 3: Studielast (normering in ecs)

	aantal weken	aantal lesuren van 50 minuten	klokuren
lesuren	10	4	33
zelfstudie			
leestijd	aantal pagina' s		
		3 per uur	
		6 per uur	
	120	10 per uur	12
presentaties			
overlegtijd			
uitzoektijd/research			11
niet ingeroosterde lestijd			
toetsen	voorbereiden		3.5
	toets		1.5
	nabespreking		1
werkstuk,verslag,rapport,scriptie	uitzoeken		
	overleggen		
	schrijven		
Stage, Praktijkopdracht	voorbereiding		
	aanwezigheid		
	overleg		
Subtotaal in klokuren			56
Ruis 5%			
Totaal in klokuren			56
Totaal in studiepunten (ects)			2