Design class diagram

PB007 Software engineering I

Stanislav Chren

1. 11. 2021



Class diagram represents a static view of classes, their attributes, operations and relationships.

Analytical class diagram

- Models the business domain of the system focus on main concpets and relationships
- Attempts to maintain clarity and simplicity without the implementation details

Design class diagram

 Extends the analytical class diagram with implementation classes and details



Design class provides such a level of abstraction so that it can be easily implemented

Design classes can originate from:

- Business domain more detailed specification of analytical classes (decomposition, inclusion of implementation details) .
- Solution domain technology-related classes (classes for working with GUI, DB, ...)

Implementation details include:

- Visibility and types of attributes.
- Visibility, arguments and return types of methods.
- Methods decomposed from analytical operations, constructors (destructors), getter/setter methods, implementation methods.



Design class - Example

analysis BankAccount name number balance

deposit()

withdraw()

calculateInterest()

design

5			
BankAccount			
–name : String –number : String –balance : double = 0	100 miles		
+BankAccount(name:String, number:String) +deposit(m:double) : void +withdraw(m:double) : boolean +calculateInterest() : double +getName() : String +setName(n:String) : void +getAddress() : String +setAddress (a:String) : void +getBalance() : double			

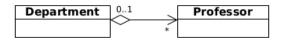


- Specification of aggregation/composition association types.
- Definition of names, navigability and multiplicities.
- Decomposition of bidirectional associations.
- Revision of 1:1, 1:M and M:1 associations.
- Decomposition of M:N associations.
- Decomposition of association classes.



Aggregation is a *whole-part* type of relationship.

- The whole usually may or may not exist without its parts.
- Parts can usually exist independently from the whole.
- The whole is in a sense incomplete if some parts are missing.
- Part can be in theory shared by multiple whole classes.
- Aggregation is transitive and asymmetrical (without cycles).





Composition is a stronger form of aggregation

- At any given time, *parts* can belong to exactly one *whole*.
- The whole is usually responsible for managment of its parts.
- If the *whole* is deleted, it has to either delete its *parts* or the *parts* have to be associated with another *whole*.
- Composition is transitive and asymmetrical (without cycles).





Analysis:

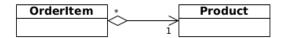






Analysis:

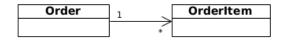


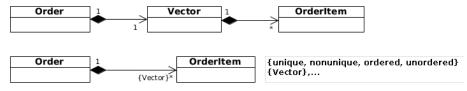




Revision of 1:M associations

Analysis:





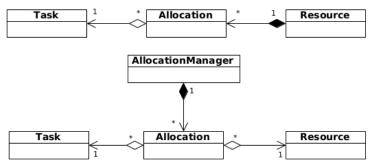


Decomposition of M:N associations

Analysis:



Design:



Note.: This decomposition is suitable only in cases when the allocation class has additional attributes. Otherwise, the M:N association does not have to be decomposed.

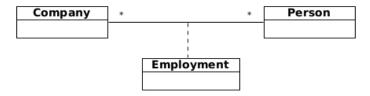
Software engineering I (PB007)

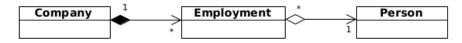
Design class diagram

1. 11. 2021 11 / 17

Decomposition of association classes

Analysis:



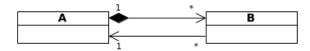




Decomposition of bidirectional associations

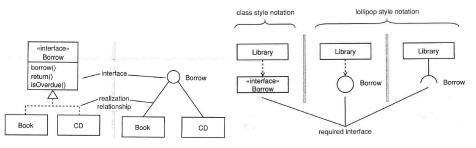
Analysis:







Interface is a special element that defines a set of public services, attributes and relationships but it does not implement them. They are used to define the contract for implementing classes.





Tasks

- Extend a copy of analytical class diagram into design class diagram.
- Specify visibility and type of all attributes.
- Add methods that originated from decomposition of analytical operations, implementation and helper methods (constructors, getters/setters, ...) and determine their visibility, arguments and return types.

The getters/setters should be added only if it is necessary.

- Further specify the associations (names, multipicity navigability, modifiesrs, determine the aggregation/composition and decompose the association classes).
- Add dependency relationships.
- If necessary, add other implmentation classes, enums and interfaces.
- Submit **pdf report** to homework vault(**Seminar 08**). **Deadline:**
 - Saturday (Groups 03, 04)
 - Monday (Group 11)
 - Tuesday (Groups 06, 07)
 - Wednesday ([06:00 AM] Groups 08. 09)

Rules for report submission

- **1** Submit the PDF report, not the VP source file and not an exported image.
- PDF report must be created using the procedure shown on the seminars including the report settings.
- The name of the PDF report file should be *lastname1-lastname2-lastname3* of the team members.
- OPDF report must contain all diagrams modelled until now.
- PDF report must be uploaded to the homework vault by the specified deadline.
- PDF report must be uploaded to the correct homework vault. The name of the homework vault is always specified on the slides.
- Each team uploads only a single PDF report for the whole team.
- Submitted diagrams must be clear and readable.
- Submitted diagrams should not contain serious mistakes. At least, they should not contain mistakes mentioned in the Catalogue of common mistakes.



VP report settings

otions	Details	
Generate table of contents	2 Children	References
Generate table of figures	? Model-based	References documentation
	Diagram-based	✓ Sub-diagrams
Image type : SVG	Members	Include sub-diagram details
Generate diagram type title	ERD Column Details	Comments
	21	Sort by Date/Time: Descending
	Properties	Tagged values
Generate diagram summary Include extra details	I Project management properties	ORM Class Details
	Relationships	Vse Case Details
Suppress element with blank documentation in summary table	Quality information	
	Anti-aliasing	
	Graphics	🔽 Text
Skip heading for empty model element section	Font	
Convert multiline model heading to single line	Font: Unspecified	
Show multiline model name	Font: Unspecified <u></u>	
Treat HTML content as HTML source		
Suppress details if duplicated ▼ Table cell keep together with page		
Vrap : Word wrap		
Shape type style : Icon 💌	2	
RTF content appearance :		
Preserve formatting		
Reset Reset to Default Set as Default	Genera	ite Cancel Apply Help