Object-Oriented paradigm in practice

Advanced Programming

Barbara Russo

OO paradigm

- It is a method to **model system at different stages**
- The more we proceed in the project the more we know and the more we can represent

Why is OO popular?

- The hope that it will increase productivity
- Natural way of structuring the world
  - Objects
  - Messages
  - Responsibility

OO Principles

- Means to achieve high quality, the four principles:
  - Encapsulation
  - Information Hiding
  - Abstraction
  - Modularization
  - Reuse

OO Principles: practical definitions

- Encapsulation
  - Building classes and objects as proper data structures
  - Entities should be divided in logically related groups, keeping interactions between different groups at a minimum
- Information Hiding
  - Hide information not needed for the messaging exchange or object's service provision of an object/class
  - Information hiding is perfectly accomplished by furnishing a compiled version of the source code that is interfaced via a header file
- In Java: Encapsulation and access identifiers; overriding or Reflection API can break encapsulation if understood as "information hiding"

OO Principles: practical definitions

- Abstraction
  - Define entities that need not to exist in the real world but that capture the nature of the derived entities
- Modularization
  - Model the world in entities
- Reuse
  - Use services and attributes of ancestor entities in the inheritance tree
  - Delegation

Exercise

- In group, make an example in each case
  - You can use class diagram or code chunks

Key Concepts recap

- classes
  - attributes
  - methods
  - inheritance
  - relations with other classes
- objects: instances of classes
  - attributes with assigned values
  - instantiated relations
- messages and methods to respond to a message

Classes and objects

- Classes are organized in:
  - Hierarchies
  - A taxonomy is a classification of the real world (Animal tree, Person tree, etc)
- Objects are instances of and belong to one class:
  - Objects know who they are

OO paradigm: Three Views

- Conceptual (OOAnalysis)
  - Shows concepts of the domain
  - Independent of implementation
- Specification (OODesign)
  - General structure of the running system
  - Interfaces of software (types)
- Implementation (OOProgramming)
  - Details of the implementation
  - Most often the only used

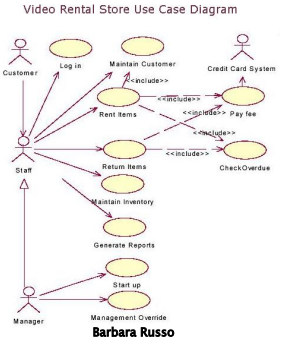
OO Analysis (OOA)

- OOA deals with modelling the system functionalities
- OOA is about "what" is the system
- Non-OO analysis uses data flow diagrams
- OOA uses conceptual diagrams, to model the **use** of the system and its **entities** (classes)

Modeling OOA

- Traditional
- Agile (this course)

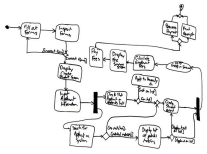
Traditional:UML Use Case Diagram

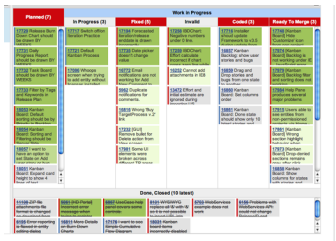


This course

- To render requirements of a system, we use the User Stories (XP approach) and back log (SCRUM).

This course: User Stories Diagrams





OO Design (OOD)

- Objective: finding the right objects and the correct relations among them
- OOA is about "how" is the system and who does what
- Objects are
  - dependent on the domain
  - even a single object performing all system functionalities could work → traditional system analysis
    - But it does not get the best of OOD

Modeling OOD

- Traditional
- Agile (this course)

OOD in this course

- To design the entities in a system, we use the CRC diagrams (agile approach) and the UML class diagrams (traditional approach)

Example: CRC cards

TApplication	Flip
Superclasses: TEventHandler	
Subclasses:	
Responsibilities:	Collaborators:
EventLoop	TWindow
DoMenuCommand	TDocument, TWindo
SetupMenus	TWindow
DoMouseCommand	TWindow

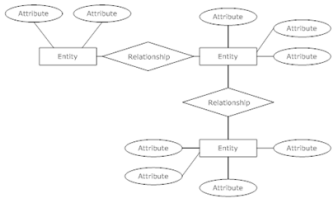
TList	Flip
Superclasses: TObject	
Subclasses:	
Responsibilities:	Collaborators:
Initialize	
Insert	
Delete	
EachItemDo	
Free	

TBox	Flip
Superclasses: TShape	
Subclasses:	
Responsibilities:	Collaborators:
Initialize	
Read	
Write	
Draw	

TObject	Flip
Superclasses:	
Subclasses: TEventHandler, TList, TShape	
Responsibilities:	Collaborators:
Free	

OOD: CRC cards

- http://www.agilemodeling.com/artifacts/crcModel.htm
- remember ER diagrams?

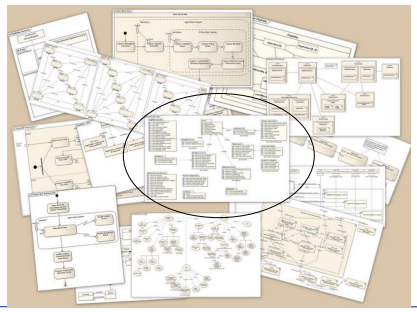


CRC cards

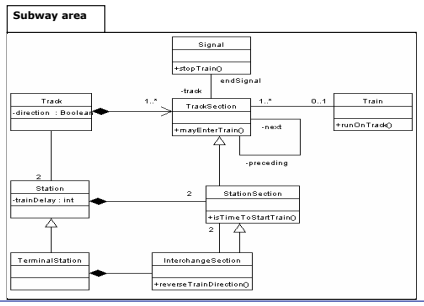
- Can be used for analysis and design. In the analysis are used simple as sets of classes interacting among each other

Traditional: UML

A web of diagrams



This course: UML class diagram



OO programming (OOP)

- The key computational entities are called "Objects"
- Objects know:
  - Who they are – ontology
  - What they do – behaviour
- The objects belong to classes

This course: OOP

- We use:
  - Java language
  - Test Cases and Junit
  - Test Driven Development

Further readings

- A Laboratory For Teaching Object-Oriented Thinking
  - Beck and Cunningham

Next Lecture

- Requirements