G64OOS (Spring 2014)

Lecture 01

Introduction to Object Oriented Systems

Peer-Olaf Siebers

The University of Nottingham
UNITED KINGDOM · CHINA · MALAYSIA

What is this module about?

Object Oriented Systems

- Object Oriented Systems (OOS) are systems composed of software objects collaborating with each other to achieve some common goal

- Software objects are conceptually similar to real-world objects: they consist of state and related behavior
  - State: Stored in fields or variables
  - Related behaviour: Exposed through methods or functions

Module Mission Statement

- The module covers an introduction to Object Oriented Analysis and Design (OOA/OOD) and how to implement such designs using Object Oriented Programming (OOP) in C++

- Students will learn how to use Object Oriented Principles along the entire trajectory of Software Development.

Motivation for Lecture 1

- Get to know organisational details of the course
- Learn how your knowledge uptake will be evaluated
- Get an idea of how OOS might be used in real life
- Get your hands dirty in your first OOA/OOD case study

Module Modalities

- Details:
  - Convenors: Peer-Olaf Siebers
  - Lectures: Wednesday 4-6pm; Exchge LT2
  - Labs: Fridays 9-11am; CompSci A32
  - Individual course works 40% (including oral examination)
  - Final exam 60%

- Presumed prior knowledge:
  - Knowledge of basic OOP (e.g. G54PRG)

Resources

- Module Website: http://www.cs.nott.ac.uk/~pos/g64oos/

- Software used
  - Software used during labs:
    - Visual Paradigm Standard
    - Visual Studio 12 Pro
  - Free versions are available for use at home
    - Visual Paradigm Community Edition
    - Visual Studio 12 Express

Resources

- Books:
  - Object Oriented Analysis and Design:
    - Fowler (2004) UML Distilled
    - Booch et al (2007) Object Oriented Analysis and Design with Applications
    - Freeman and Freeman (2006) Head First: Design Patterns
    - Gamma et al (1995) Design Patterns: Elements of Reusable Object Oriented Software
  - Object Oriented Programming:
    - Liberty and Cadenhead (2011) Sams Teach Yourself C++ in 24 Hours
    - Parsons (2000) Object Oriented Programming with C++
    - Stroustrup (1997) The C++ Programming Language

Module Overview

- Lectures
  - Lecture 01: Introduction
  - Lecture 02: Object Oriented System Analysis
  - Lecture 03: Object Oriented System Modelling
  - Lecture 04: Object Oriented Programming (Principles)

- Labs
  - Lab 01: Programming Practice (Basics)
  - Lab 02: Programming Practice (Inheritance; Pointers; Arrays)
  - Lab 03: UML Practice
  - Lab 04: Programming Practice (OO Principles)

Module Overview

- Lectures
  - Lecture 05: Object Oriented Programming in C++ (Part 1)
  - Lecture 06: Object Oriented Programming in C++ (Part 2)
  - Lecture 07: Agile Programming
  - Lecture 08: Testing + Test Driven Development

- Labs
  - Lab 05: Programming Practice (OO Principles)
  - Lab 06: Programming Practice (OO Principles) + Coursework Release
  - Lab 07: Coursework Clinic
  - Lab 08: Test driven development case studies

Module Overview

- Lectures
  - Lecture 09: Elements of Reusable OOS
  - Lecture 10: -
  - Lecture 11: Exam Revision

- Labs
  - Lab 09: Coursework Clinic
  - Lab 10: -
  - Lab 11: Coursework – Oral Assessment

Course work

Object Oriented Programming Coursework (current plan)

- Individual work
- Implementation of rich class hierarchy
- Implementation of data structure routines
- Implementation of pre-defined interfaces
- Oral assessment: 11 April @ 9:00-11:00
- Submission deadline: 11 April @ 12:00

- Exam (current plan)
  - 3 compulsory questions
  - Previous exam available from module website

Introduction

- Your first day at work
  - Develop the control software for a novel video tagging tool
  - Project budget: £50.000
  - Project time: 6 month
  - Project team: you and two other programmers
  - Produce software that is easy to maintain and extend

- How would you approach this project?

Introduction

Introduction

Your task requires an OO approach!

- OO Analysis
  - Talking with stakeholders; using use cases
- OO Modelling
  - Using UML diagrams
- OO Programming
  - Using agile methods

Useful OOS Concepts

- Useful concepts:
  - Abstraction
    - Use of abstract classes which cannot be instantiated; a parent class that contains the common functionality of a collection of child classes, but the parent class itself is too abstract to be used on its own.
  - Inheritance
    - One object acquires the properties of another; information is made manageable in a hierarchical order
  - Encapsulation
    - Hiding internal state and requiring all interaction to be performed through an object's methods

Useful OOS Concepts

- Useful concepts:
  - Modularity
    - The source code for an object can be written and maintained independently of the source code for other objects
  - Polymorphism
    - Allows different classes to have different implementations of the same methods

A Brief History of OOS

- Software development theory
  - Procedural Paradigm
  - Modular Paradigm
  - Data Abstraction Paradigm
  - Object Oriented Paradigm

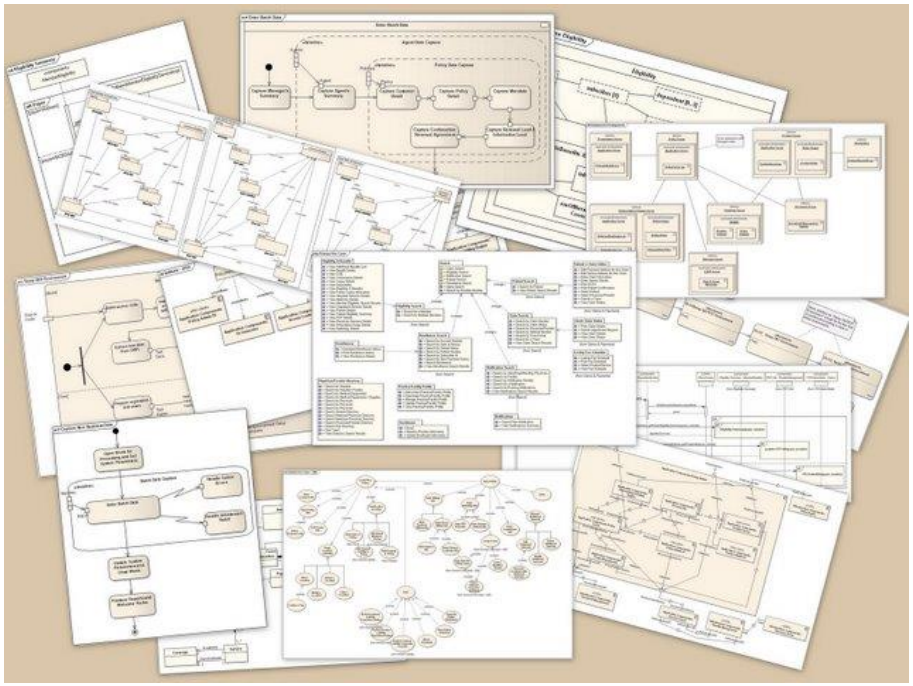
- History of OOP languages
  - Simula
  - C++
  - Eiffel
  - Java

Break

- See you back in 10 minutes

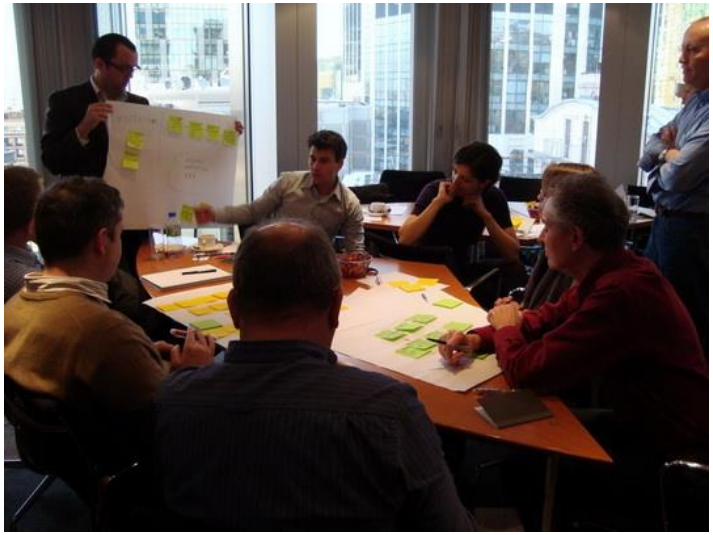
Hands-On OOAD Example

- Analysis
  - The User Story
  - Use Case Diagrams
- Design
  - Class Diagrams
  - State Machine Diagrams
  - Sequence Diagrams
- Programming



Hands-On OOAD Example

- Analysis: The User Story
  - Stating the need
  - Collecting and prioritising high-level features
  - Should be written by project stakeholders and not the developers
  - Keep it simple!



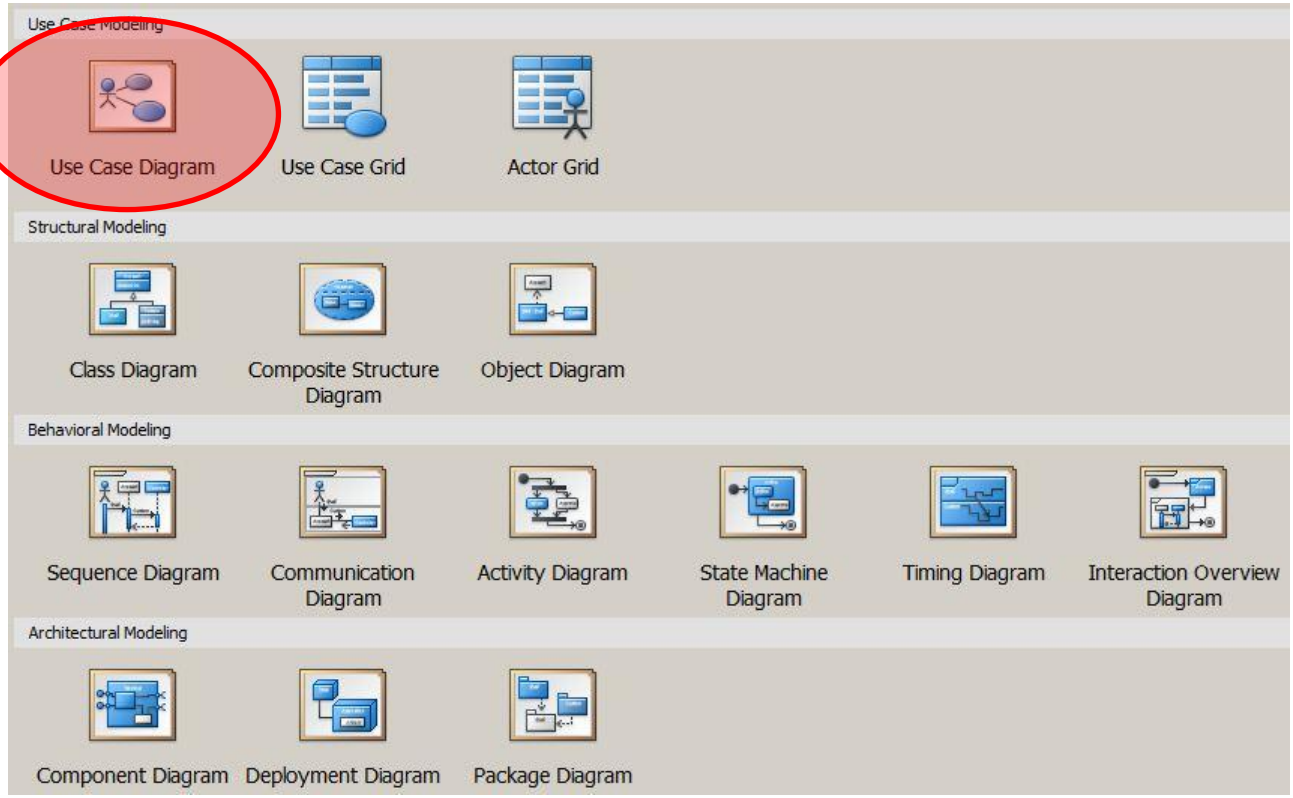
Hands-On OOAD Example

- The User Story
  - Develop a simulation software that allows to simulate customer/staff interactions in a grocery store
    - Story: Customers come into a grocery store, pick up a few items, pay for them, and leaves the grocery store
    - Goals of simulation:
      - Help to improve customer experience
      - Optimise staffing

  - More about Systems Simulation
    - Module G54SIM (http://www.cs.nott.ac.uk/~pos/g54sim/)

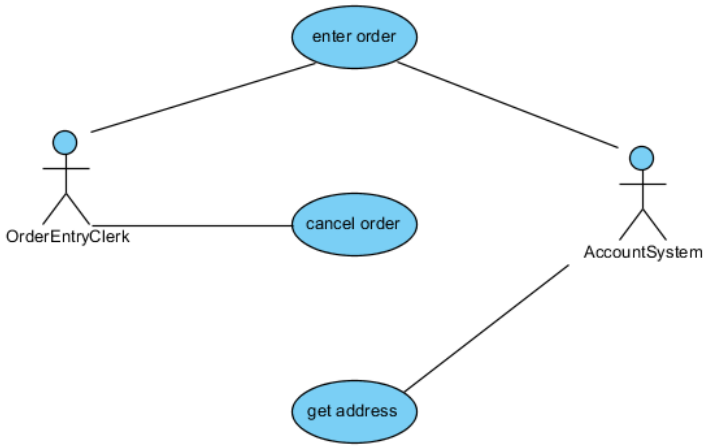
Hands-On OOAD Example

- Analysis: Use Case Diagrams



Hands-On OOAD Example

- Use Case Diagrams
  - UML diagram(s) + Specification + Prototype Screen(s)
  - Telling a story in a highly structured way
    - Define actors: Entities that interface with the system (roles or systems)
    - Define use cases (procedures by which an actor might use a system)
    - Sometimes it is useful to sub-divide UseCases into lower level activities
    - Define environment (not always required)

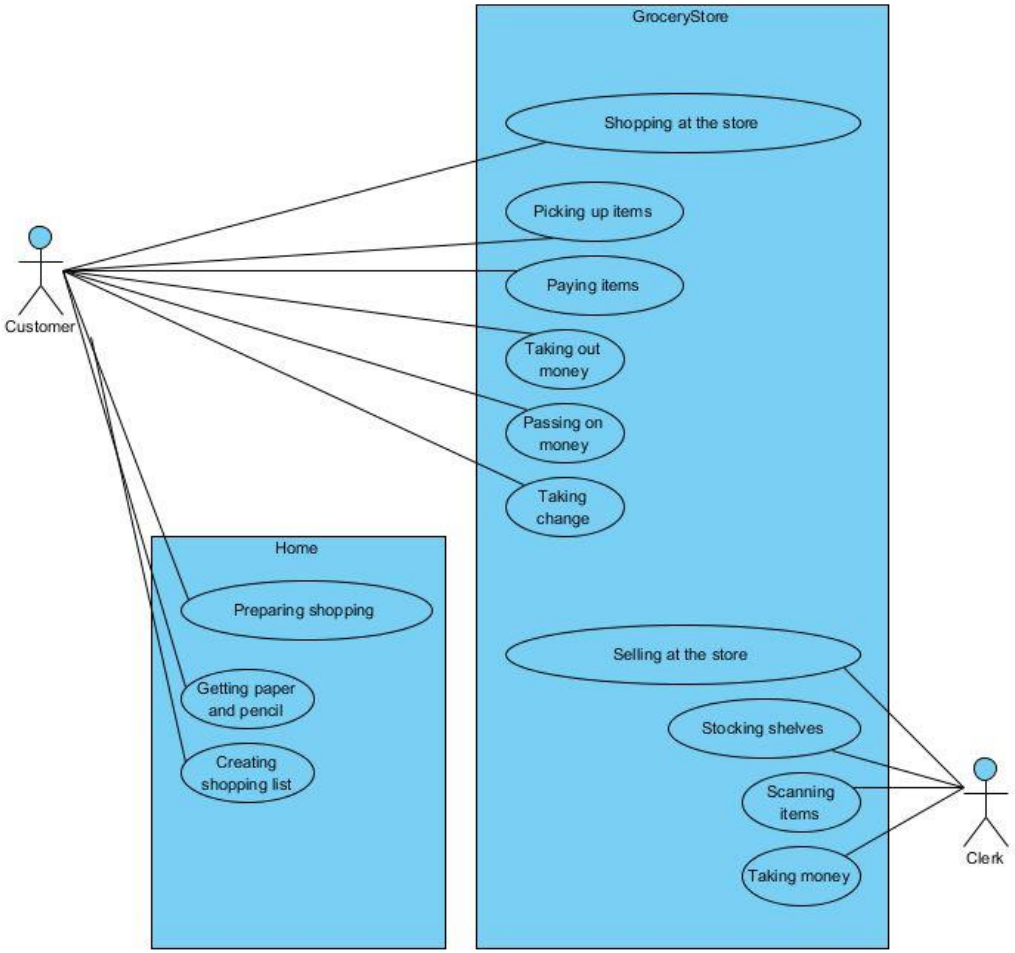


Hands-On OOAD Example

- Use Case Diagrams
  - Develop a simulation software that allows to simulate customer/staff interactions in a grocery store
    - User Story: Customers come into a grocery store, pick up a few items, pay for them, and leaves the grocery store.
    - Goals of simulation:
      - Help to improve customer experience
      - Optimise staffing

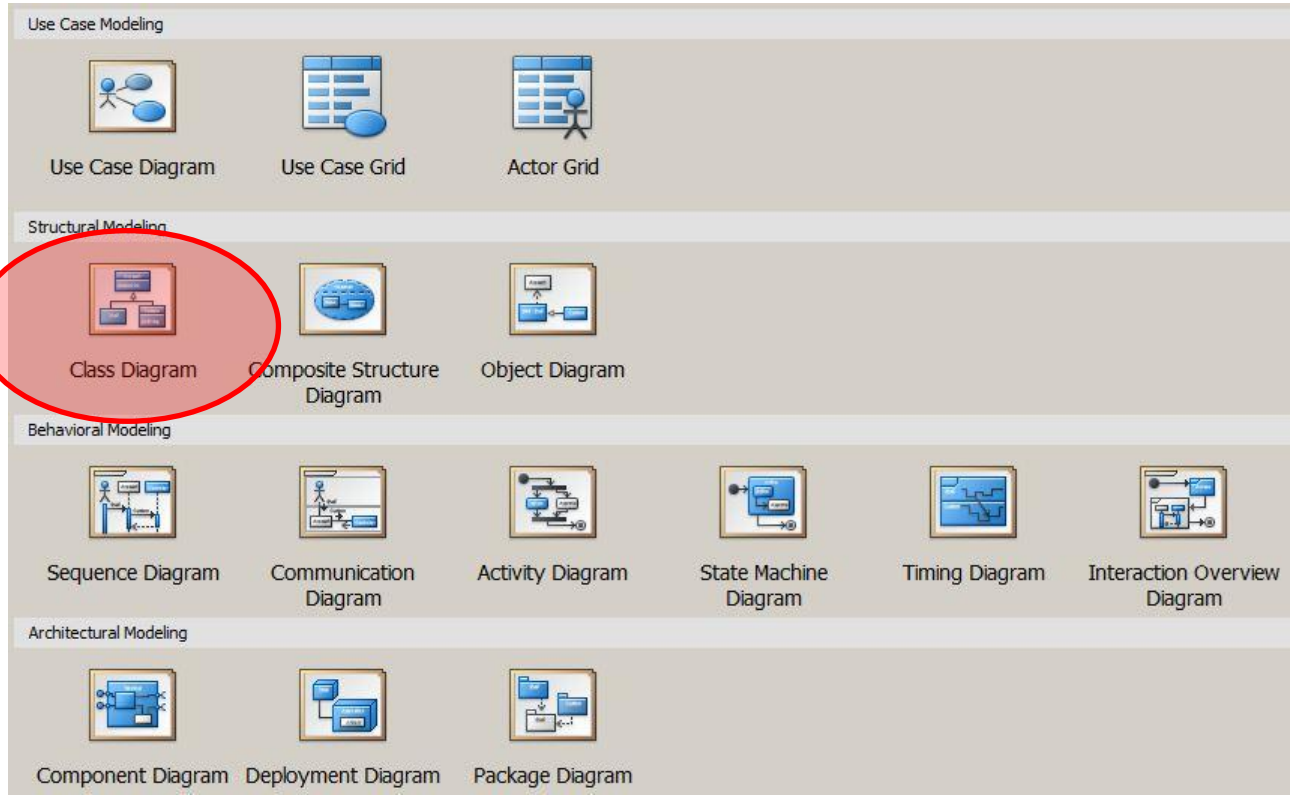
- Your task: Create a Use Case Diagram

Hands-On OOAD Example



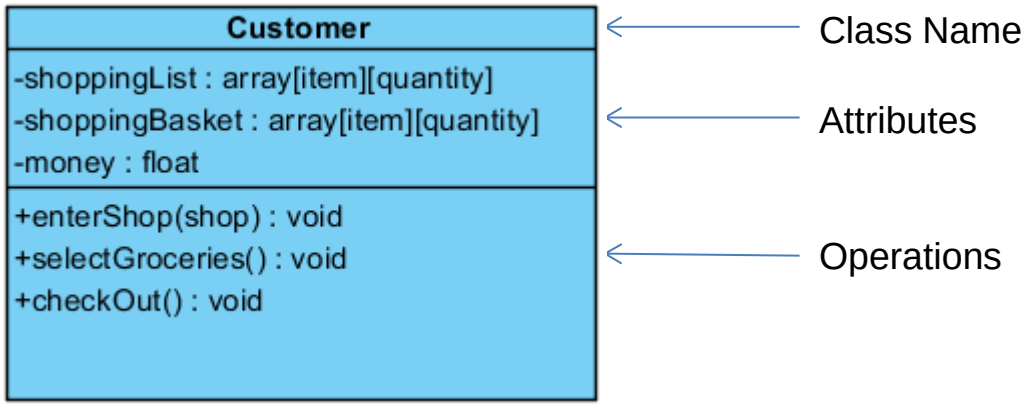
Hands-On OOAD Example

Design: Class Diagrams



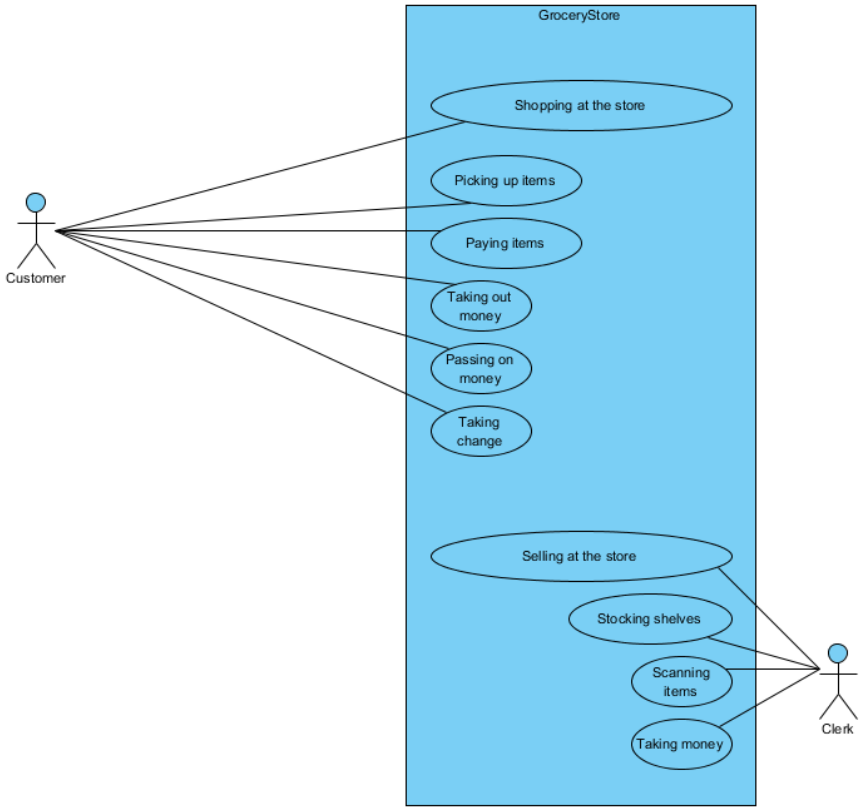
Hands-On OOAD Example

- Class Diagrams
  - Show a set of classes, interfaces and collaborations, and their relationships
  - Addresses static design view of a system
  - Classes
    - Blueprints (templates) for objects
    - Contain data/information and perform operations



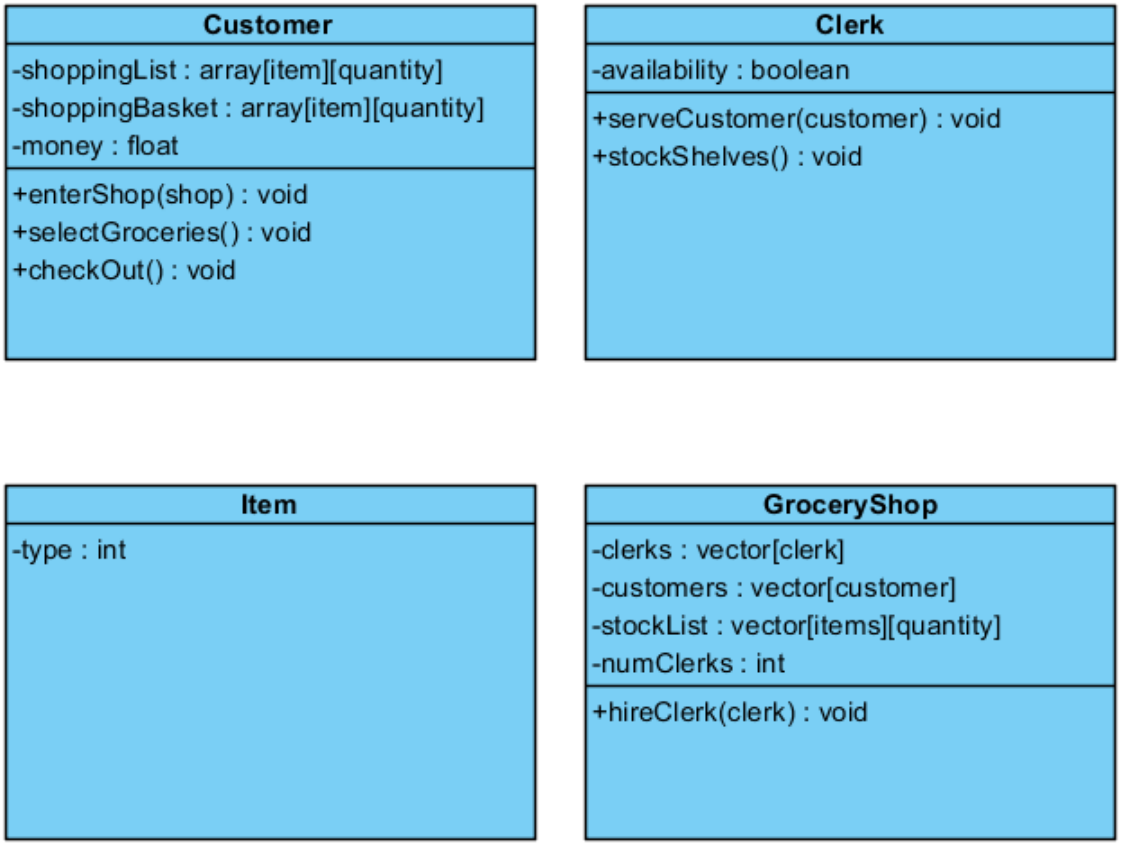
Hands-On OOAD Example

- Your Task: Create the remaining Classes



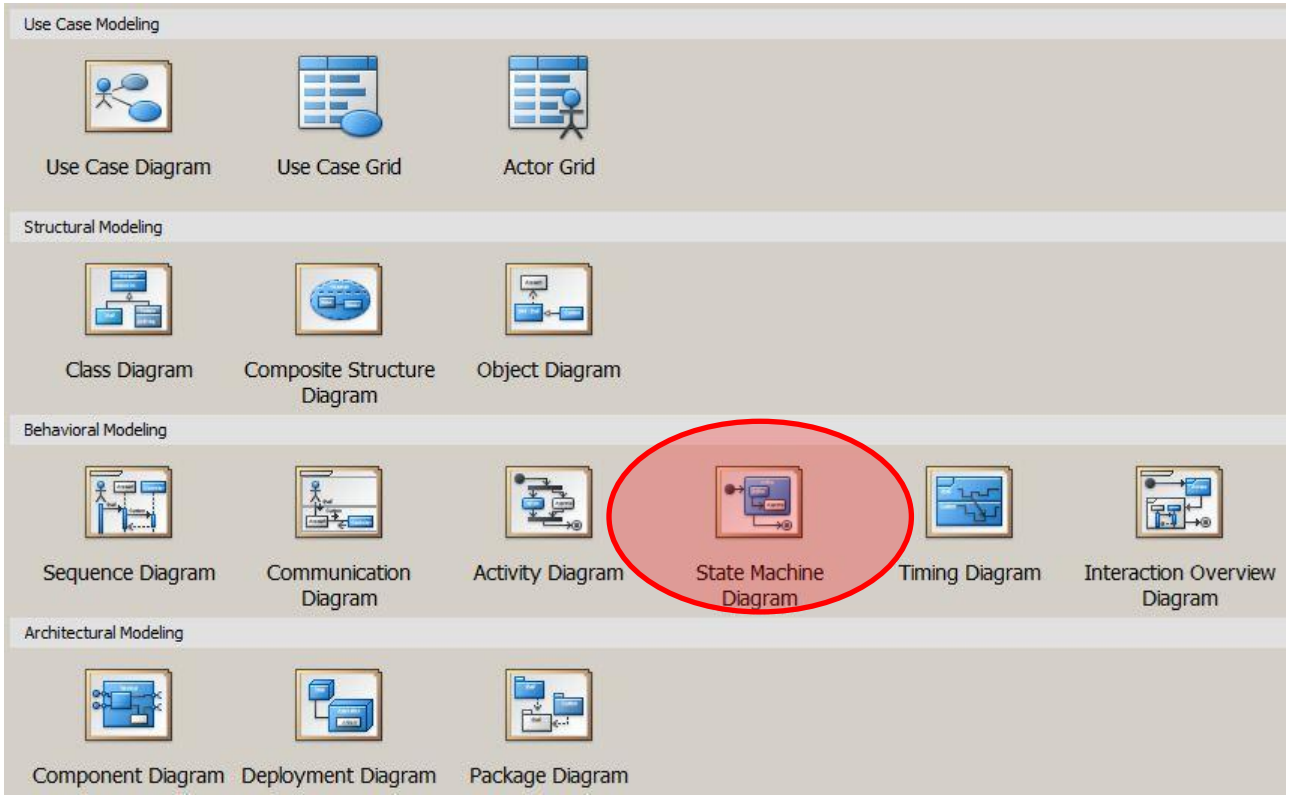
Hands-On OOAD Example

- Class Diagrams



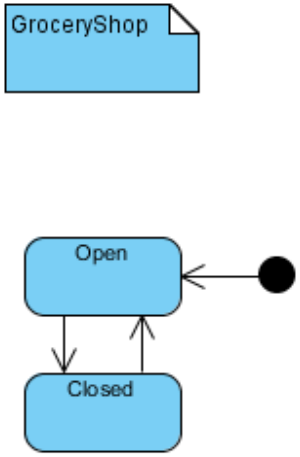
Hands-On OOAD Example

- Design: State Machine Diagrams



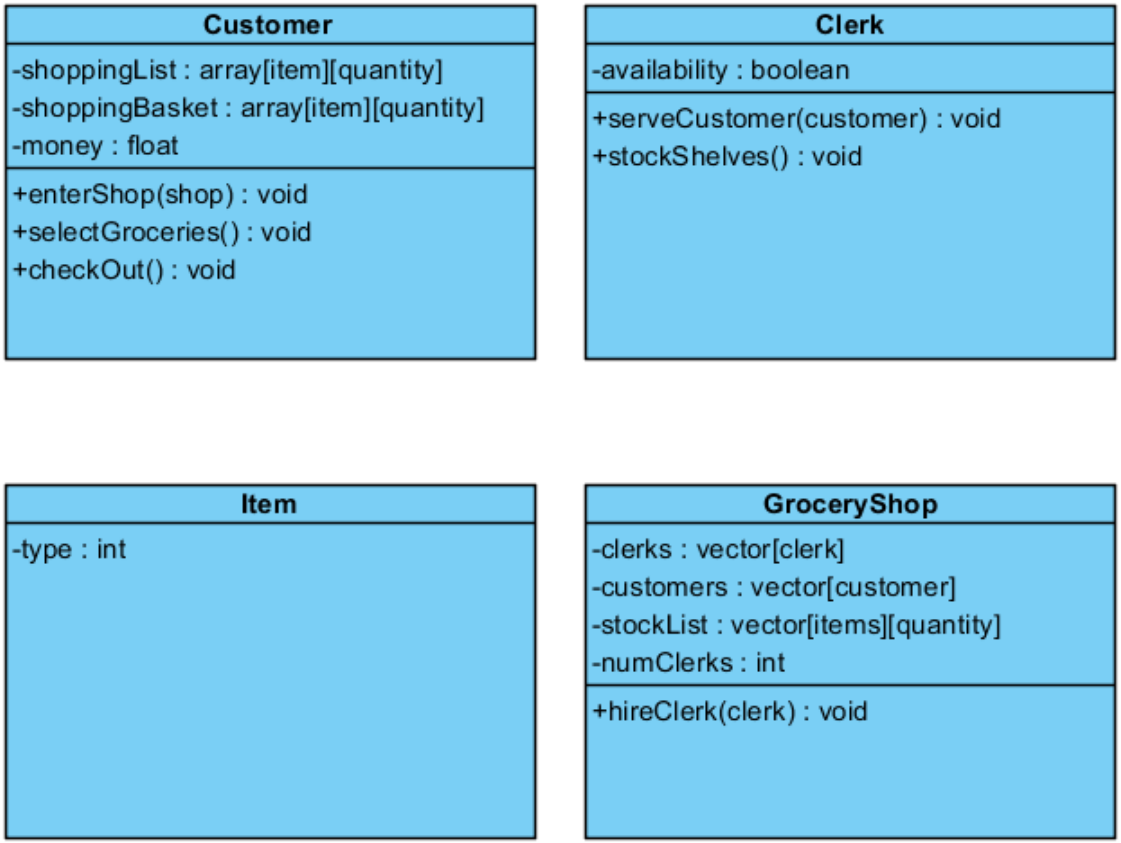
Hands-On OOAD Example

- State Machine Diagrams
  - Addresses the dynamic view of a system and is important in modelling the behaviour of an interface, class or collaboration
  - Helps to understand how the system behaves in reaction to key events
  - Consist of states, transitions, events, and activities



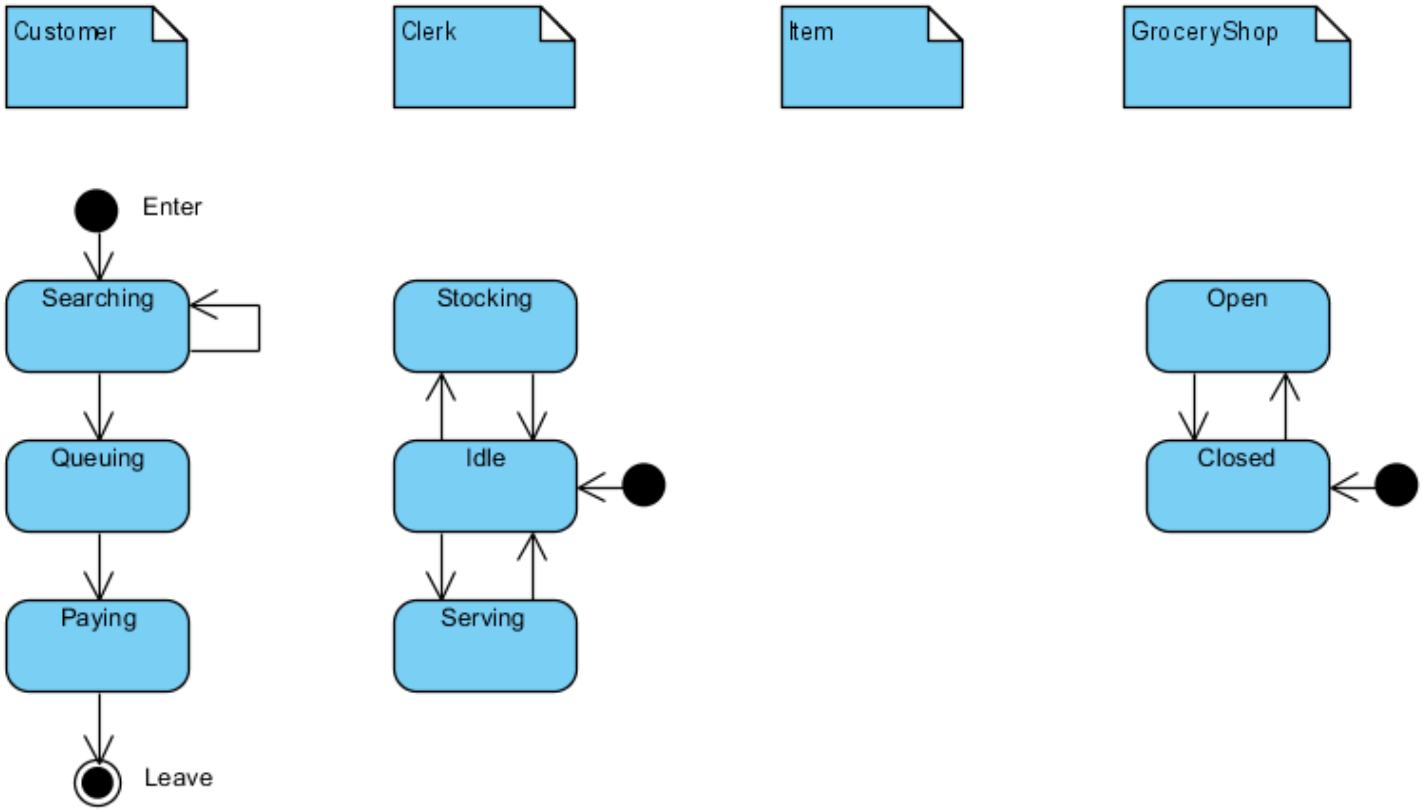
Hands-On OOAD Example

- Your Task: Create the remaining State Machines



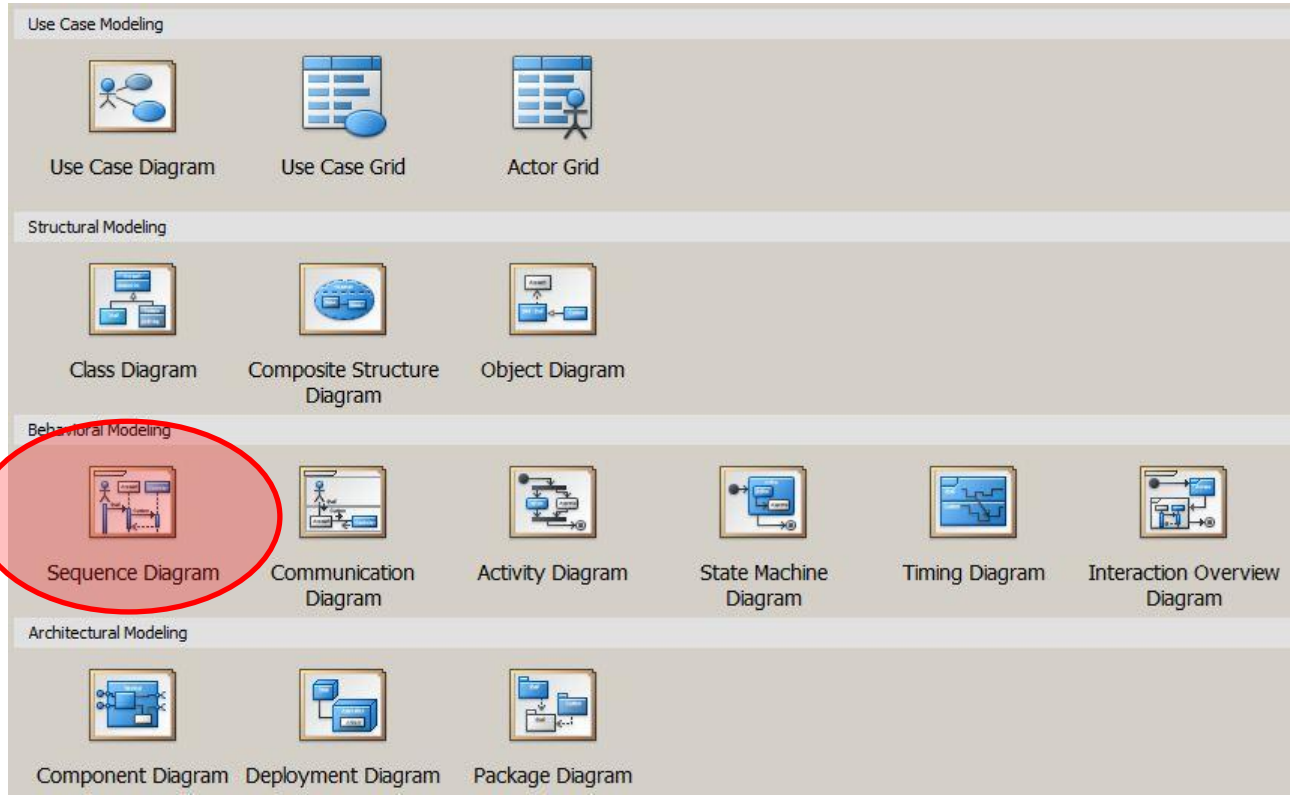
Hands-On OOAD Example

- State Machine Diagrams



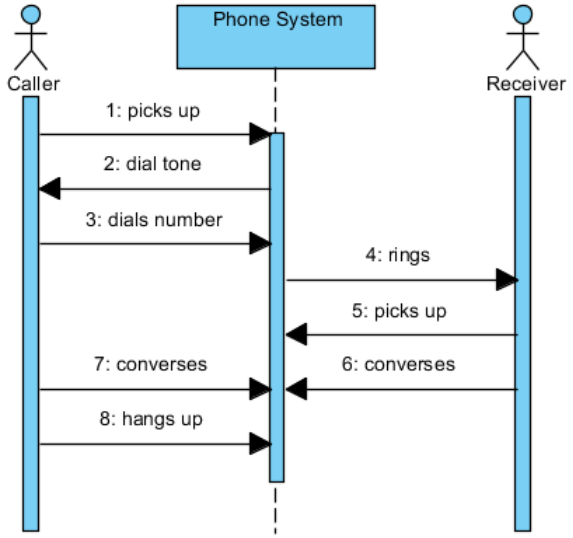
Hands-On OOAD Example

- Design: Sequence Diagrams



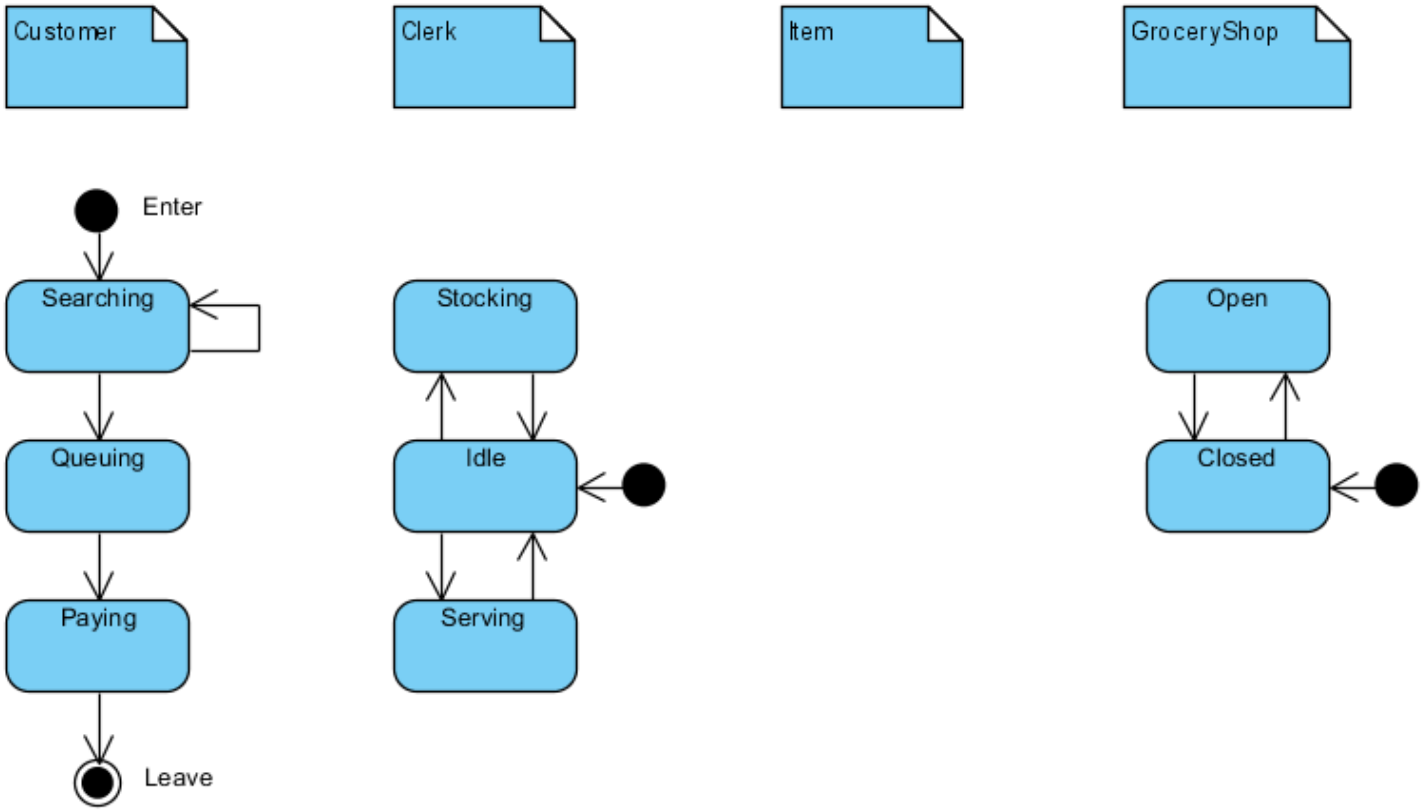
Hands-On OOAD Example

- Sequence Diagrams
  - Shows potential interactions consisting of a set of objects and the messages sent and received by those objects
  - Address the dynamic behaviour of a system with special emphasis on the chronological ordering of messages
  - Consists of objects, messages, object lifelines, activation



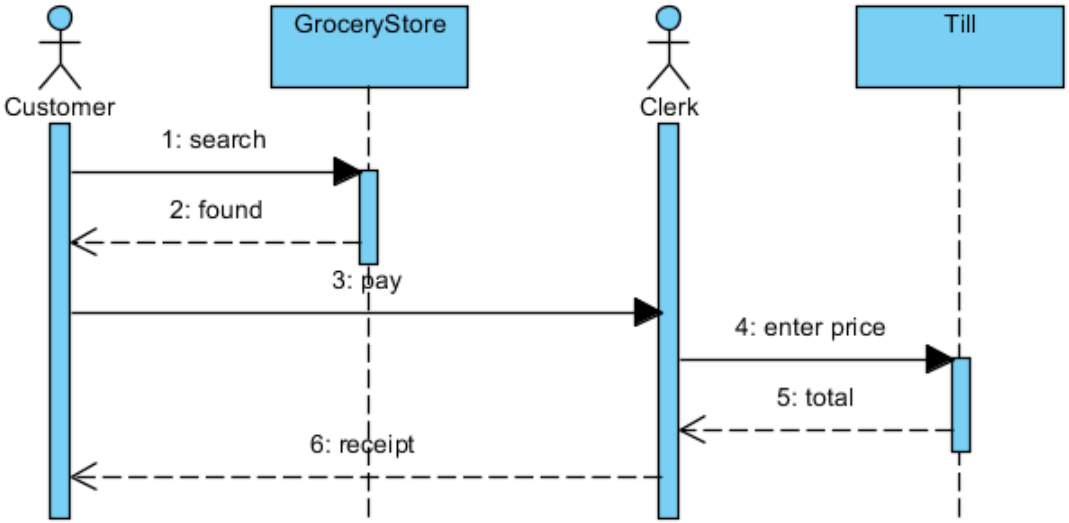
Hands-On OOAD Example

- Your Task: Create the Sequence Diagram



Hands-On OOAD Example

- Sequence Diagram



From OOAD to OOP

- Programming

```
class Customer
{
    // -- Members variables
    float money;
    std::list<item> shoppingList;
    std::list<item> shoppingBasket;

    // -- Member functions
    void shop();
    void collectGroceries();
    void checkOut();
    void pay(float m);
};
```

```
class Clerk
{
    // -- Member variables
    bool availability;

    // -- Member functions
    bool isAvailable();
    void setAvailability(bool b);
    void serve();
    void stock();
}
```

From OOAD to OOP

- The result :)

Summary

- The OOS paradigm aims to design software by modelling problems in a way that comes natural to humans, that is, by categorising the world around us into objects that relate to and interact with each other.
- C++ was designed to implement such OOMs efficiently, while still giving the programmer great power over the available resources (most notably memory and processor cycles).
- UML is used to specify, visualise, modify, construct and document the artefacts of an OO software-intensive system under development.

Questions / Comments