E-course

What Is Object-Orientation?

In This Lecture You Will Learn

E-course

- The fundamental concepts of object-orientation
- The justifications for an object-oriented approach

Structured Approach

E-course

- Broadly speaking, there are two general approaches to software development: the structured approach and the object-oriented approach.
- The structured approach has been very fashionable since the 1970s; it was adequately supported by conventional procedural languages.
- The structured approach is centered on the system's functional views and uses different models at various stages of the development process.
- When development progresses from one stage to the next, the models in the current stage are transformed into the models of the next stage.

Structured Approach (cont'd)

E-course

- There are three major weaknesses with the structured approach:
  - when functions of the system change => the analysis, the design models and the implementation of the system will have to be changed substantially.
  - transformation needs to be carried out whenever the models of the early stages have changed as a result of changes in the requirements or the correction of previous mistakes.
  - the dynamic view is almost non-existent in the structured approach.
- The above weaknesses of the structured approach have made it less cost-effective when compared with the object-oriented approach.

Object-oriented Approach

E-course

- This approach models a software system as a collection of collaborating objects.
- An object and its data interact with other objects through messages being sent and received by the object and which manipulate the object's data in the process.
- The object-oriented approach allows the software engineer to develop consistent models of a software system much more easily, because the same set of models are used throughout the whole development process.

Object-oriented Approach (cont'd)

E-course

- No effort or time is wasted by transforming and updating models in different stages.
- Changes to an object-oriented system are localized in the objects themselves.
- Therefore, the structure of a system developed by the object-oriented approach is more stable than that by the structured approach.

Basic Concepts of OO

E-course

- The main concepts introduced here are:
  - Objects, Classes and Instances
  - Object State
  - Generalization and Specialization
  - Message-passing and Encapsulation
  - Polymorphism

Objects

E-course

An object is:

"an abstraction of something in a problem domain, reflecting the capabilities of the system to

- keep information about it,
- interact with it,
- or both."

Coad and Yourdon (1990)

Objects

"Objects have state, behaviour and identity."

Booch (1994)

- *State*: the condition of an object at any moment, affecting how it can behave
- *Behaviour*: what an object can do, how it can respond to events and stimuli
- *Identity*: each object is unique

Examples of Objects

E-course

Object	Identity	Behaviour	State
A person.	‘Hussain Pervez.’	Speak, walk, read.	Studying, resting, qualified.
A shirt.	My favourite button white denim shirt.	Shrink, stain, rip.	Pressed, dirty, worn.
A sale.	Sale no #0015, 18/05/05.	Earn loyalty points.	Invoiced, cancelled.
A bottle of ketchup.	*This* bottle of ketchup.	Spill in transit.	Unsold, opened, empty.

Class and Instance

E-course

- All objects are *instances* of some *class*
- Class:

  a description of a set of objects with similar
  - features (attributes, operations, links);
  - semantics;
  - constraints (e.g. when and whether an object can be instantiated).

OMG (2004)

Class and Instance

E-course

- An object is an instance of some class
- So, instance = object
  - but also carries connotations of the class to which the object belongs
- Instances of a class are similar in their:
  - *Structure*: what it *knows*, what information it holds, what links it has to other objects
  - *Behaviour*: what an object *can do*

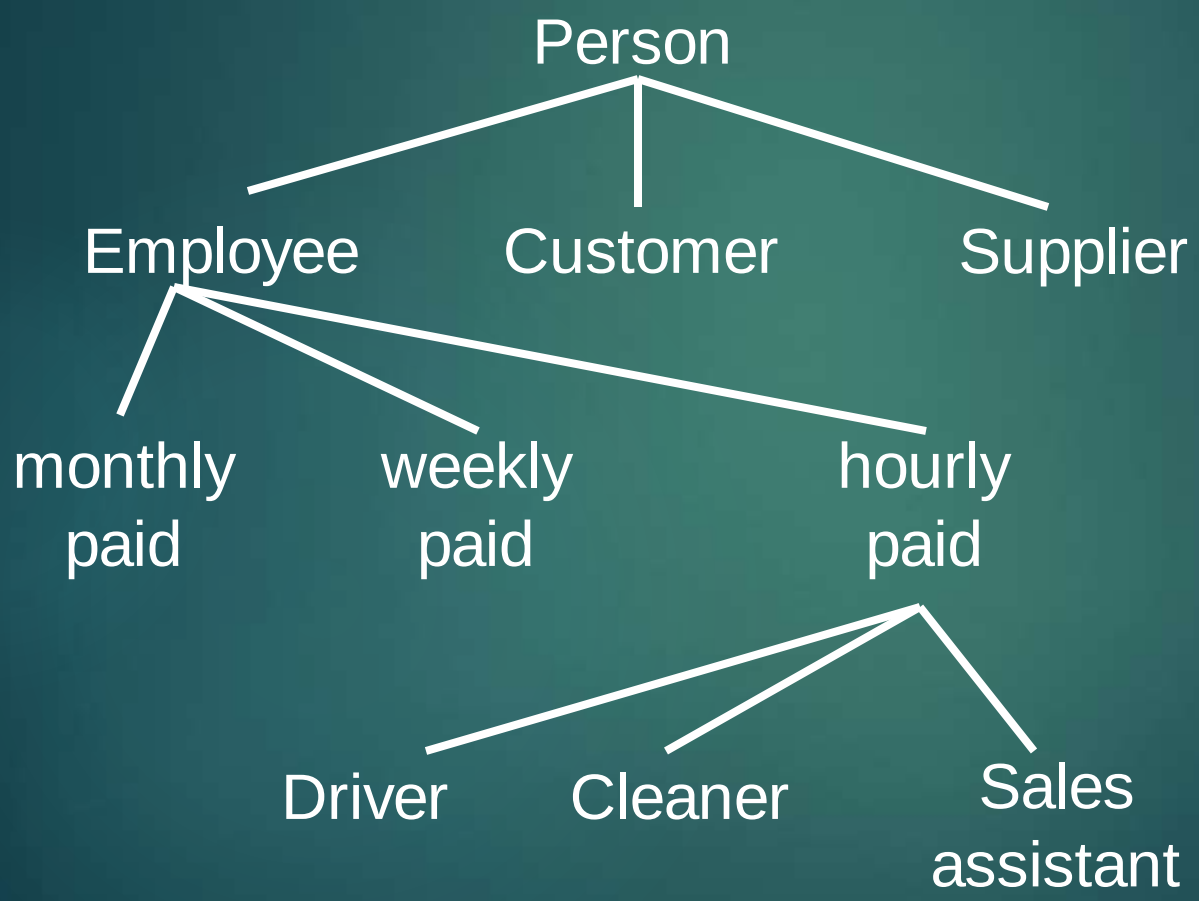
Generalization and Specialization

E-course

- Classification is hierarchic in nature
- For example, a person may be an employee, a customer, a supplier of a service
- An employee may be paid monthly, weekly or hourly
- An hourly paid employee may be a driver, a cleaner, a sales assistant

Specialization Hierarchy

E-course



More general (super classes)

More specialized (subclasses)

Generalization and Specialization

E-course

- More general bits of description are *abstracted out* from specialized classes:

General (superclass)

Person
name date of birth gender title

Specialized (subclass)

HourlyPaidDriver
startDate standardRate overtimeRate licenceType

Inheritance

E-course

- The *whole* description of a superclass applies to *all* its subclasses, including:
  - Information structure (including associations)
  - Behaviour
- Often known loosely as *inheritance*
- (But actually inheritance is how an O-O programming language *implements* generalization / specialization)

Message-passing

E-course

- Several objects may collaborate to fulfil each system action
- “Record CD sale” could involve:
  - A CD stock item object
  - A sales transaction object
  - A sales assistant object
- These objects communicate by sending each other messages

Message-passing and Encapsulation

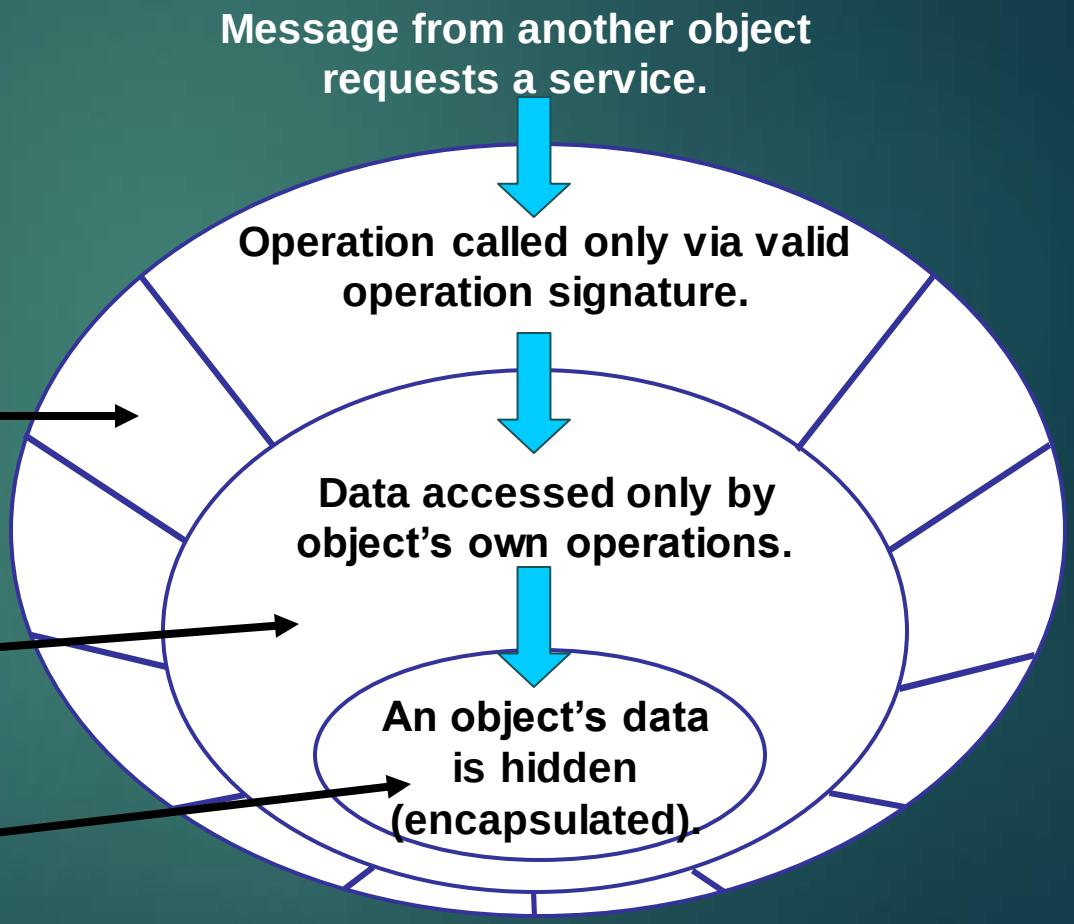
E-course

'Layers of an onion' model of an object:

An outer layer of operation signatures…

…gives access to middle layer of operations…

…which can access inner core of data



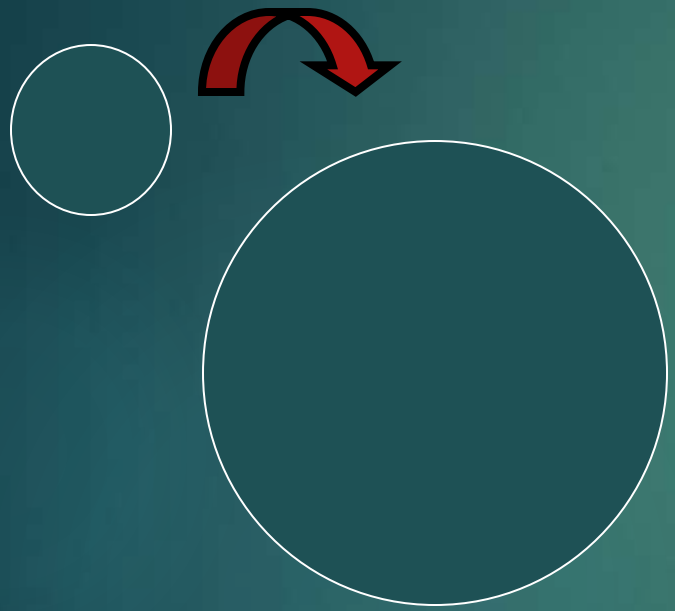
Polymorphism

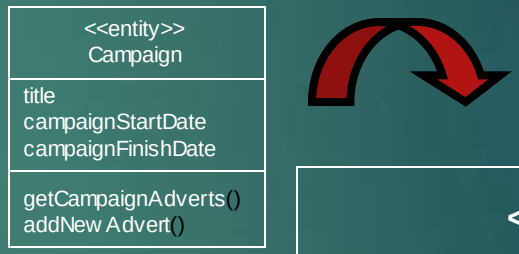
E-course

- Polymorphism allows one message to be sent to objects of different classes
- Sending object need not know what kind of object will receive the message
- Each receiving object knows how to respond appropriately
- For example, a ‘resize’ operation in a graphics package

Polymorphism in Resize Operations

E-course





<<entity>>
Campaign

title
campaignStartDate
campaignFinishDate

getCampaignAdverts()
addNewAdvert()

Advantages of O-O

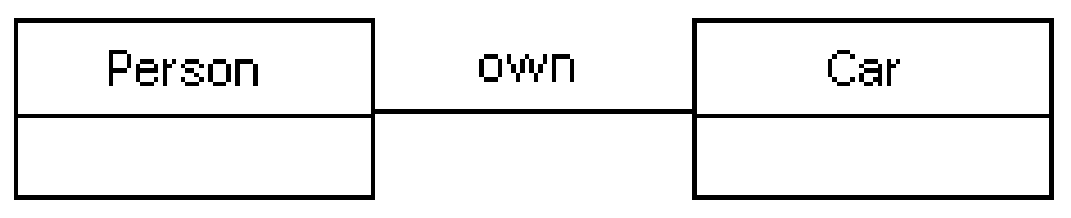
E-course

- Can save effort
  - Reuse of generalized components cuts work, cost and time
- Can improve software quality
  - Encapsulation increases modularity
  - Sub-systems less coupled to each other
  - Better translations between analysis and design models and working code

Visual Modeling

E-course

- The human brain is capable of handling and processing only a limited amount of information at any one time.
- Models can help reduce complexity by creating an abstract hierarchical representation of the real-world system.
- Visual modeling is the process of representing a system with essential parts from a particular perspective using some standard graphical notations.



Visual Modeling (cont'd)

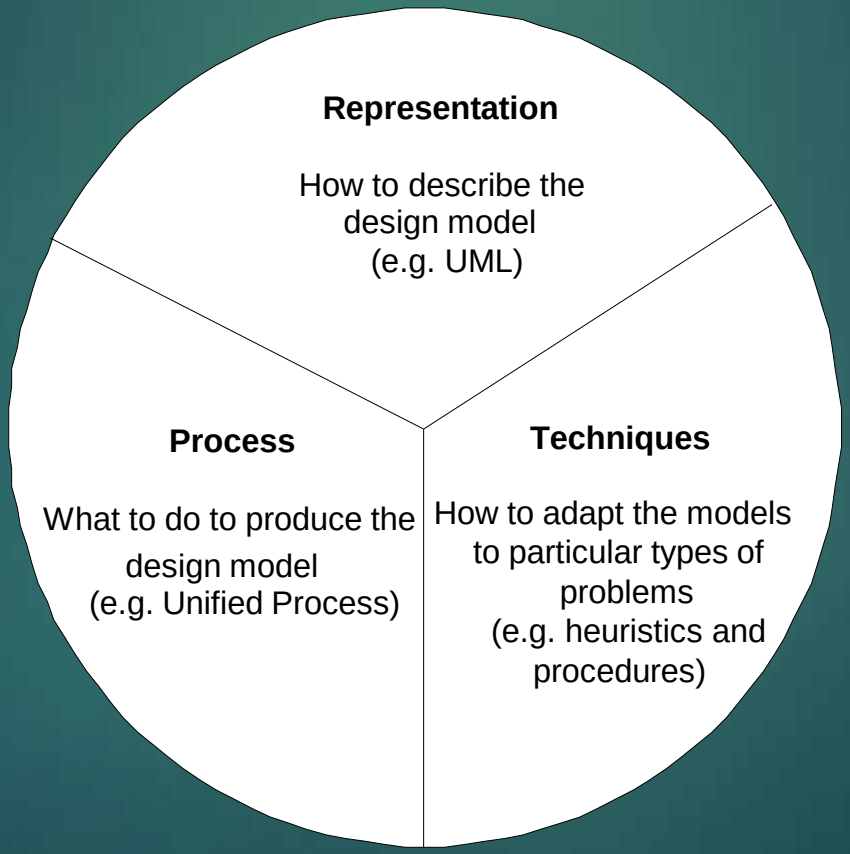
E-course

- In the software development process, visual modeling can:
  - capture business objects and logic;
  - analyze and design applications;
  - manage complexity;
  - define the software architecture; and
  - model systems independent of the implementation language.

- Unified Modeling Language (UML)

Software Development Methods

E-course



Role of Notation

E-course

- Capture requirements of the system;
- Analyze the system by developing suitable analysis models - Models are expressed in an appropriate notation so that the developer can easily find the things from which he or she can quickly extract information;
- Develop the design of the system - Design models are developed and expressed in an appropriate notation that can be understood by the system designer and the programmer. The system designer may need to manipulate the analysis model and make design decisions in the process; and
- Implement, test and deploy the system - Again, the artifacts of these activities are expressed in a suitable notation that can be understood by the system designer, the programmers and system testers.

Role of Process

- Ideally, a process should offer the following features:
  - a well-managed iterative and incremental life cycle to provide the necessary control without affecting creativity;
  - embedded methods, procedures and heuristics for developers to carry out analysis, design and implementation for the whole software development life cycle (SDLC);
  - a guide through the iterative and incremental development process for the solution of complex problems;
  - a comprehensive roadmap so that designers can walk through the flexible multiple pathways of the development process depending on the nature of the problem; and
  - identification of less obvious requirements based on what have already been known or modeled.

Role of Techniques

E-course

- The main purpose of the techniques part of a method is to provide a set of guidelines and heuristics to help the developer to systematically develop the required design models and implementation.

- The techniques part of a method should include the following:
  - a set of guidelines to produce and verify the design against the original requirements and specifications;
  - a set of heuristics for the designer to ensure consistency in the structure of a design and also the design models. The latter requirement is particularly important when the design is produced by a team of designers who will need to ensure that their models are consistent and coherent;
  - a system to capture the essential features of the design so as to supplement the limited designer's domain knowledge.

Overview of the UML

E-course

- The UML is accepted by the Object Management Group (OMG) as a standard way of representing object-oriented analysis and design models.
- It has quickly become the *de facto* standard for building object-oriented software.
- The OMG specification states:
  - "The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components."
- Refer to Appendix B of the book (Curtis H.K. Tsang, Clarence S.W. Lau and Y.K. Leung (2005)) for a more detailed description of the UML.

Overview of Visual Paradigm for UML

E-course

- CASE tools can significantly help developers increase their productivity, particularly if they provide facilities that automate many model building procedures.
- Indeed, some CASE tools offer sophisticated facilities such as diagram-to-code and code-to-diagram, with real-time synchronization and consistency maintained in both directions.
- VP-UML, like most leading CASE tools, meets the following requirements:
  - Facilitate convenient model building. Models of the system should be easily developed. Editor and documentation tools should be provided and easy to use;
  - Serve as repository. Models can be saved and retrieved with ease;
  - Support navigation. Linkages between models can be maintained and traversed;

Overview of Visual Paradigm for UML

E-course

- Generate documentation automatically. Documentations can be generated for selected information of the software development project;
- Facilitate project management. The project activities can be planned and managed with ease;
- Facilitate configuration management and version control. Documentations and components of different versions of the system can be managed;
- Check model consistency. Consistency between models of the system can be checked;
- Support model verification and validation. The correctness of the models of the system can be verified and validated;
- Provide multi-user support. Multiple developers can work on the project simultaneously and coherently;
- Generate code. Code can be generated from models;

Overview of Visual Paradigm for UML

- Reverse engineering. Models can be generated from code; and
- Provide integration with other tools. The CASE tool can be integrated with domain specific systems or tools so as to accelerate the development process.

- Appendix A of the book (Curtis H.K. Tsang, Clarence S.W. Lau and Y.K. Leung (2005)) provides more information about how you can get started using the VP-UML CASE tool.

References

E-course

- Curtis H.K. Tsang, Clarence S.W. Lau and Y.K. Leung (2005)
- Coad and Yourdon (1990)
- Booch (1994)
- OMG (2004)

(For full bibliographic details, see Bennett, McRobb and Farmer)

WE FOCUS ON KNOWLEDGE-BASED ON EDUCATION

KSRA of Empowerment is a global non-profit organization committed to bringing empowerment through education by utilizing innovative mobile technology and educational research from experts and scientists. KSRA emerged in 2012 as a catalytic force to reach the hard to reach populations worldwide through Learning management system & E-learning & mobile learning.

The KSRA team partners with local under-served communities around the world to improve the access to and quality of knowledge based on education, amplify and augment learning programs where they exist, and create new opportunities for e-learning where traditional education systems are lacking or non-existent.

E-course