Information Systems Concepts

Fundamentals of Object Technology

Roman Kontchakov

Birkbeck, University of London

Based on Appendix of Maciaszek, L.A.:

Outline

- Object
- Class
- Association
- Aggregation and Composition
- Generalization
- Inheritance
- Polymorphism
Object: State, Behaviour & Identity

NB: equal ≠ identical

**Class**

**Class** represents a set of objects that share a common structure and a common behaviour.

**Objects** are instances of classes.

Booch, G.: *Object Oriented Analysis and Design with Applications (2nd Edition)*
Addison-Wesley, 1994
Objects: UML Notation

**c1: Module**

<table>
<thead>
<tr>
<th>attribute-name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>module_code</td>
<td>COIY016H4</td>
</tr>
<tr>
<td>module_name</td>
<td>Information Systems Concepts</td>
</tr>
</tbody>
</table>

**NB:** ‘:Module’ is an anonymous instance of class **Module**

**NB:** ‘c1’ is an object without a specified class

**NB:** there is no compartment for operations!
Classes: UML Notation

**Purchase**

- **id:** Integer
- **date:** Date
- **value:** Currency

- **reorder(product:** Product)
Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.

Encapsulation

Encapsulation hides the details of the implementation of an object

Addison-Wesley, 1994
Operations (in Communication Diagrams)

1: shipOrder(order: Order)
→ 2: getProducts(order: Order)
→ 3: checkLevels()
↓ 4: reorder(product: Product)
Operations (in Class Diagrams)

Order
- shipOrder(order: Order)

Shipment
- shipOrder(order: Order)

Product

Stock
- getProducts(order: Order)
- checkLevels()

Purchase
- reorder(product: Product)
Associations

Order
- id: Integer
- date: Date
- value: Currency

Shipment
- id: String
- date: Date
- carrier: String

shipOrder(order: Order)

multiplicity:
- 0..1: none or one
- 0..*: any number
- 1..*: at least one
- n: exactly n
Association Classes

ModuleOffering

Assessment
marks: List
totalMark: Number
grade: String

Student
Aggregation: by reference (transitive and asymmetric)

Composition: by value (transitive, asymmetric and existence-dependant)
Generalization

A subclass **inherits**
the structure and behavior
of its superclass

Addison-Wesley, 1994
Inheritance in Java

public class Person
{
    private String name;
    private Date dob;
    public int age()
    {
        return getYear() - getYear(dob);
    }
}

public class Employee extends Person
{
    private Date startDate;
    private int leaveEntitlement;
    private int leaveTaken;
    public int remainingLeave()
    {
        return leaveEntitlement - leaveTaken;
    }
}
Polymorphism in Java

public class Manager extends Employee {
    private int leaveSupplement;

    public int remainingLeave() {
        int l = super.remainingLeave();
        return l + (leaveSupplement);
    }
}
Take Home Messages

- Each **object** has a state, behaviour and identity
- **Class** defines attributes and operations
- There are three kinds of relationships between classes:
  - association,
  - aggregation/composition and
  - generalization
- **Generalization** provides basis for inheritance and polymorphism