The OO Solution

- The OO model closely resembles the problem domain
  - Base your model on the objects in the problem domain
- Iteratively refine the high-level model until you have an implementation
  - Attempt to avoid big conceptual jumps during the development process

Objects

State of Michigan
Drivers License
J. Q. Public
A-123456
03-12-63

VISA
J. Q. Public
123 4567 887766 998

Attributes and Operations

Person objects

Card objects

Card class

Person class

Attributes

- name
- age
- height
- weight

Operations

- move
- change-job

Attributes

- height
- width
- id-number

Operations

- issue
- change

Characteristics of Objects

- Identity
  - Discrete and distinguishable entities
- Classification
  - Abstract entities with the same structure (attributes) and behavior (operations) into classes
- Polymorphism
  - The same operation may behave differently on different classes
- Inheritance
  - Sharing of attributes and operations based on a hierarchical relationship
The Class Diagrams

Objects

- Something that makes sense in the application context (application domain)
  - J.Q. Public
  - Joe’s Homework Assignment 1
  - J. Q. Public’s drivers license
- All objects have identity and are distinguishable
- NOT objects
  - Person
  - Drivers license
Classes

- Describes a group of objects with:
  - similar properties (attributes),
  - common behavior (operations),
  - common relationships to other classes, and
  - common semantics

- Person
  - J. Q. Public
  - Joe Smith
  - D. Q. Public
- Card
  - Credit card
  - Drivers license
  - Teller card

Class Diagrams

Class diagram

<table>
<thead>
<tr>
<th>Person</th>
<th>age: integer</th>
</tr>
</thead>
</table>

Instance diagram

<table>
<thead>
<tr>
<th>D. Q. Public: Person</th>
<th>J. Q. Public: Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>age= 32</td>
<td>age= 35</td>
</tr>
</tbody>
</table>

Class with attributes

Objects with values

| Person
| name: String |
| person ID: integer |
| age: integer |

Objects have an identity
- Do not explicitly list object identifiers
- SSN OK!
Examples

<table>
<thead>
<tr>
<th>Person</th>
<th>Card</th>
</tr>
</thead>
<tbody>
<tr>
<td>name: String</td>
<td>height: integer</td>
</tr>
<tr>
<td>age: integer</td>
<td>width: integer</td>
</tr>
<tr>
<td>height: integer</td>
<td>thickness: integer</td>
</tr>
<tr>
<td>weight: integer</td>
<td>id-number: integer</td>
</tr>
<tr>
<td>SSN: integer</td>
<td></td>
</tr>
</tbody>
</table>

Operations and Methods

- Transformation that can be applied to or performed by an object
- May have arguments

<table>
<thead>
<tr>
<th>Card</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>height: integer</td>
<td>height: integer</td>
</tr>
<tr>
<td>width: integer</td>
<td>width: integer</td>
</tr>
<tr>
<td>thickness: integer</td>
<td>rotate(angle: integer)</td>
</tr>
<tr>
<td>id-number: integer</td>
<td>move(x: integer, y: integer)</td>
</tr>
<tr>
<td>issue()</td>
<td></td>
</tr>
<tr>
<td>revoke()</td>
<td></td>
</tr>
</tbody>
</table>
Object Notation - Summary

<table>
<thead>
<tr>
<th>Class name</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute-1 : data-type-1 = default-value-1</td>
</tr>
<tr>
<td>attribute-2 : data-type-2 = default-value-2</td>
</tr>
<tr>
<td>attribute-3 : data-type-3 = default-value-3</td>
</tr>
<tr>
<td>operation-1(argument-list-1) : result-type-1</td>
</tr>
<tr>
<td>operation-2(argument-list-2) : result-type-2</td>
</tr>
<tr>
<td>operation-3(argument-list-3) : result-type-3</td>
</tr>
</tbody>
</table>

Associations

- Conceptual connection between classes
  - A credit card is issued-by a bank
  - A person works-for a company

Class diagrams

Instance diagram
Associations are Bi-directional

- There is no direction implied in an association (Rumbaugh - OMT)

  ![Diagram showing associations between Country, Has-capital, City, Person, Is-issued, Drivers-license, lic.-number: integer]

Associations Can Have Direction

- Unified adds a direction indicator
  - Inconsistently used

  ![Diagram showing associations between Country, Has-capital, City, Person, Is-issued, Drivers-license, lic.-number: integer]
One person holds one credit card

- One object can be related to many objects through the same association

One person can hold zero or more credit cards

Multiplicity (Cont.)

- One person can hold zero or more credit cards (0..*)
- Each card has zero or one holder (0..1)
Multiplicity (Cont.)

- One person can hold zero or more credit cards (0..*)
- Each card has one holder (no indication or 1)
- Each card has one or more authorized users (1..*)
- One person can be authorized to use zero or more cards

Explicit enumeration is also possible (2, 3, 2..5, etc.)

### Higher order associations

- Ternary association
  - Project, language, person
- Seldom needed (and should be avoided)

Instance of ternary association
**Link Attributes**

- Associations can have properties the same way objects have properties

```
Person
  name: String
  age: integer
  SSN: integer
  address: String

Company
  name: String
  address: String

0..* Works-for
```

**How to represent salary and job title?**

```
Person
  name: String
  age: integer
  SSN: integer
  address: String

Company
  name: String
  address: String

0..* Works-for
```

**Use a link attribute!**

```
Person
  name: String
  age: integer
  SSN: integer
  address: String
  salary: integer
  job-title: String

Company
  name: String
  address: String

0..*
```

**Folding Link Attributes**

**Why not this?**

Salary and job title are properties of the job **not** the person

```
Person
  name: String
  age: integer
  SSN: integer
  address: String
  salary: integer
  job-title: String

Company
  name: String
  address: String

0..*
```

**In this case, a link attribute is the only solution**

```
Person
  name: String
  age: integer
  SSN: integer
  address: String

Company
  name: String
  address: String

0..* Works-for 0..*
```

```
Person
  name: String
  age: integer
  SSN: integer
  address: String
  salary: integer
  job-title: String

Company
  name: String
  address: String

0..*
```

CSE870: UML Classes

19

CSE870: Advanced Software Engineering
(Cheng)
Role Names

- Attach names to the ends of an association to clarify its meaning

```
Person
  name: String
  age: integer
  SSN: integer
  address: String

Company
  name: String
  address: String

Works-for
  0..*
  employee
  0..*
  employer

salary: integer
job-title: String

Manager
  0..1
  boss

Manages
  0..*
  worker

Worker
  0..*
  employee

Aggregation

- A special association, the is-part-of association
  - A sentence is part of a paragraph (a paragraph consists of sentences)
  - A paragraph is part of a document (a document consists of paragraphs)
```

Aggregation symbol
Aggregation (Cont.)

- Often used in parts explosion

```
Car
  \downarrow
 Wheel  Body  Gearbox  Engine
  \downarrow
 Door  Hood  Trunk
  \downarrow
 Piston  Valve  Crankshaft
```

Generalization and Inheritance

- The is-a association
  - Cards have many properties in common
  - Generalize the common properties to a separate class, the base-card
  - Let all cards inherit from this class, all cards is-a base-card (plus possibly something more)
Example

Aggregation Versus Association

- Can you use the phrase is-part-of or is-made-of
- Are operations automatically applied to the parts (for example, move) - aggregation
- Not clear what it should be……
Aggregation Versus Inheritance

- Do not confuse the is-a relation (inheritance) with the is-part-of relation (aggregation)
- Use inheritance for special cases of a general concept
- Use aggregation for parts explosion

Recursive Aggregates

- A recursive aggregate contains (directly or indirectly) an instance of the same kind of aggregate
Class diagram Metamodel I

Class diagram Metamodel II

Eclipse.org
CSE 435: Software Engineering

Science direct
CSE 435: Software Engineering
Use Case Metamodel I

- **uu**: use case association relationship
- **i**: includes
- **e**: extends
- **g**: generalization
- **aa**: actor relationship

Use Case Spec Metamodel


Image: Science Direct
Use Case Metamodel II

Meta Model Architecture

- **M3**: language for describing meta-models (MOF: Meta Object Facility)
- **M2**: meta model for a given modeling language
- **M1**: class diagram
- **M0**: instance of class diagram
Interesting/Fun Metamodels

- Facebook:

Object Modeling Summary

- Classes
  - Name
  - Attributes
  - Operations
- Associations
  - Roles
  - Link attributes
- Aggregation
  - Inheritance