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
  - 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 with attributes

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

Objects with values

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

Person

- name: String
- age: integer
- height: integer
- weight: integer
- SSN: integer

Card

- height: integer
- width: integer
- thickness: integer
- id-number: integer

Operations and Methods

• Transformation that can be applied to or performed by an object

Card

- height: integer
- width: integer
- thickness: integer
- id-number: integer
- issue()
- revoke()

• May have arguments

Shape

- height: integer
- width: integer
- rotate(angle: integer)
- move(x: integer, y: integer)
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>
</tbody>
</table>

| operation-1 (argument-list-1) : result-type-1 |
| operation-2 (argument-list-2) : result-type-2 |
| operation-3 (argument-list-3) : result-type-3 |

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)

\[
\begin{array}{ccc}
\text{Country name} & \text{Has-capital} & \text{City name} \\
\text{Person name} & \text{Is-issued} & \text{Drivers-license lic.-number: integer}
\end{array}
\]

Associations Can Have Direction

- Unified adds a direction indicator
  - Inconsistently used

\[
\begin{array}{ccc}
\text{Country name} & \text{Has-capital} & \text{City name} \\
\text{Person name} & \text{Is-issued} & \text{Drivers-license lic.-number: integer}
\end{array}
\]
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.)

<table>
<thead>
<tr>
<th>Person</th>
<th>1</th>
<th>Holds</th>
<th>0..*</th>
</tr>
</thead>
<tbody>
<tr>
<td>name: String</td>
<td></td>
<td>Authorized</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Credit-card</th>
<th>1..*</th>
<th>Holds</th>
<th>0..*</th>
</tr>
</thead>
<tbody>
<tr>
<td>card-number: integer</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

![Diagram showing associations with properties]

How to represent salary and job title?

Use a link attribute!

Folding Link Attributes

Why not this?

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

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

Manages
  0..1

boss

worker

salary: integer
job-title: String
```

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)

```
Document
  0..*

Paragraph
  0..*

Sentence
```

Aggregation symbol
Aggregation (Cont.)

- Often used in parts explosion

![UML Diagram]

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
Object Modeling Summary

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

- Aggregation
- Inheritance