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

Person class
Attributes:
- name
- age
- height
- weight
Operations:
- move
- change-job

abstracts to

Card objects

Card class
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

<table>
<thead>
<tr>
<th>Class diagram</th>
<th>Instance diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>D. Q. Public: Person</td>
</tr>
<tr>
<td>age: integer</td>
<td>age= 32</td>
</tr>
</tbody>
</table>

Class with attributes

Objects with values

- 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

<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

J.Q. Public
Age=35
Works-for
Michigan State Univ
Company
Works-for
Company
Issued-by
Bank

Person

Credit Card
Associations are Bi-directional

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

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

Associations Can Have Direction

- Unified adds a direction indicator
  - Inconsistently used

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

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)

```plaintext
Person
name: String

Holds

Credit-card
card-number: integer

Person
name: String

Holds

0..1

Credit-card
card-number: integer

0..*

:JQPublic:Person

name= J. Q. Public
age=35

Holds

Card789:Credit-Card
card-number= 123 456 789

:QPublic:Person

name= D. Q. Public
age=32

Holds

Card123:Credit-Card
card-number= 111 222 333

Card456:Credit-Card
card-number= 444 555 666
```

---

CSE870: Advanced Software Engineering (Cheng)
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

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
• 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
```

• Often used in parts explosion

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

Consider the example of an airline system where:
- **City**
  - name
  - Located-In
- **Airport**
  - name
  - heat()
  - clean()
- **Airline**
  - name
- **Pilot**
  - name
  - license
- **Plane**
  - model
  - serial
  - hours flown
- **Passenger**
  - name
  - location
- **Flight**
  - date
  - flight #
  - cancel()
  - delay()
- **Owens**
  - Based-In
  - Works-for
  - Offers

The relationships and operations are as follows:
- Based-In: 0..*
- Works-for: 0..*
- Offers: 0..*
- Arrives: 0..*
- Departs: 0..*
- Confirmed-for: 0..*
- heat():
- clean():
- reserve():
- cancel():
- delay():
- heat():
- refuel():
- validate():
- issue():
- revoke():
- credit-limit: integer

The diagram illustrates the hierarchical structure and relationships among different entities.
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……

![Diagram of Aggregation Versus Association with UML notations: Company connected to Division with 0..* multiplicities, Division connected to Department with 0..* multiplicities, Person connected to Company with 0..* multiplicities.]

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

![Diagram of Aggregation Versus Inheritance with UML class diagram: Car extends Wheel, Body, Gearbox, Engine. Minivan and Compact are instances of Car, Jeep is another instance of Car with a Roll Bar.]

CSE870: Advanced Software Engineering
(Cheng)
Recursive Aggregates

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

Class diagram Metamodel 1

Eclipse.org
Class diagram Metamodel II

Use Case Metamodel I

Use Case Metamodel II

- **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**