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

CSE 435: Software Engineering
Attributes and Operations

Person objects

Person class

Attributes
- name
- age
- height
- weight

Operations
- move
- change-job

Card objects

Card class

Attributes
- height
- width
- id-number

Operations
- issue
- change

Person objects:
- J. Q. Public
- VISA

Card objects:
- State of Michigan Drivers License
  - J. Q. Public
  - A-123456
  - 03-12-63
- VISA
  - J. Q. Public
  - 123 4567 887766 998
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
- Person
  - age: integer

Instance diagram
- D. Q. Public: Person
  - age = 32
- J. Q. Public: Person
  - age = 35

Class with attributes
- Person
  - person ID: integer
  - name: String
  - age: integer

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>
</tr>
</thead>
<tbody>
<tr>
<td>height: integer</td>
</tr>
<tr>
<td>width: integer</td>
</tr>
<tr>
<td>thickness: integer</td>
</tr>
<tr>
<td>id-number: integer</td>
</tr>
<tr>
<td>issue()</td>
</tr>
<tr>
<td>revoke()</td>
</tr>
</tbody>
</table>

- May have arguments

<table>
<thead>
<tr>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>height: integer</td>
</tr>
<tr>
<td>width: integer</td>
</tr>
<tr>
<td>rotate(angle: integer)</td>
</tr>
<tr>
<td>move(x: integer, y: integer)</td>
</tr>
</tbody>
</table>

CSE 435: Software Engineering
# 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)
Associations Have Direction

• Unified adds a direction indicator
  ▪ Inconsistently used

```
Country
  name
  \arrow[Has-capital] \rightarrow
City
  name

Person
  name
  \arrow[Is-issued] \rightarrow
Drivers-license
  lic.-number: integer
```
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
• One person can hold zero or more credit cards (0..*)
• Each card has zero or one holder (0..1)
Higher order associations

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

Note: hexagons should be rectangles to represent instances
Link Attributes

Associations can have properties the same way objects have properties.

<table>
<thead>
<tr>
<th>Person</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>name: String</td>
<td>name: String</td>
</tr>
<tr>
<td>age: integer</td>
<td>address: String</td>
</tr>
<tr>
<td>SSN: integer</td>
<td>salary: integer</td>
</tr>
<tr>
<td>address: String</td>
<td>job-title: String</td>
</tr>
</tbody>
</table>

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
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 (Cont.)

- Often used in parts explosion

```
Car
  ↓
Wheel  Body  Gearbox  Engine
  ↓  ↓  ↓
  Door  Hood  Trunk  Piston
  ↓  ↓  ↓
  4  (1..*)  (1..*)  (1..*)
  ↓  ↓  ↓
  0..*   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:
  - Aggregation
  - Composition
Recursive Aggregates

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

- PrimitiveDataType
- Classifier
  - name : String
  - returnType
  - type
  - parent
- Association
  - isComposition : Boolean
  - isAggregation : Boolean
- Class
  - isAbstract : Boolean
- Attribute
- Operation
  - name : String
- Parameter
  - name : String

Eclipse.org

CSE 435: Software Engineering
Use Case Metamodel I

- uu: use case association relationship
- _i: includes
- _e: extends
- _g: generalization
- aa: actor relationship

Object Modeling Summary

- **Classes**
  - Name
  - Attributes
  - Operations

- **Associations**
  - Roles
  - Link attributes

- **Aggregation**
- **Inheritance**