Design and UML Class Diagrams

Identify and Design Classes

- Class identification from project spec / requirements / lab handout
  - nouns are potential classes, objects, properties
  - verbs are potential methods or responsibilities of a class
- Need a way to organize and visualize those classes and their relationships

UML Diagrams

- Diagrams of an OO system
  - PLs are not abstract enough for OO design
  - UML is an open standard; lots of companies use it
- Ways to utilize UML?
  - as a descriptive language: rigid formal syntax (like programming)
  - as a prescriptive language: shaped by usage and convention
- okay to omit things from UML diagrams if they aren't needed by team/supervisor/instructor
How to Use Diagrams

• As a sketch to communicate aspects of system
  – forward design: doing UML before coding
  – backward design: doing UML after coding as documentation
  – often done on whiteboard or paper
  – used to get rough selective ideas
• As a blueprint: a complete design to be implemented
• As a programming language: with right tools, code can be auto-generated from UML diagram

UML Object Diagram

• Show individual objects. (Heap Layout)
  – name : type
  – attribute = value
  – lines: references

UML Class Diagram

• A picture of:
  – the classes in an OO system
  – their properties and methods
  – connections between the classes that interact or inherit from each other
  – "abstraction" of Object Diagram
• Not represented in a UML class diagram:
  – details of how the classes interact with each other
  – algorithmic details; how a particular behavior is implemented
Anatomy of Single Class Box

- Class name in top of box
  - include protocol, struct

- Attributes (optional)
  - should include all properties of the object

- Operations / methods (optional)
  - omit trivial (get/set) methods
  - but don’t omit any methods from an protocol!
  - don't include inherited methods

Class Attributes (Properties)

- visibility name: type = default_value

- visibility:
  + public
  # protected
  - private
  ~ internal (default)
  / derived

- static properties:
  - underline

Class Operations (Methods)

- visibility name(params)
  -> return_type

- visibility:
  + public
  # protected
  - private
  ~ internal (default)

- static methods:
  - underline

Comments

changing any public property will trigger a full redraw
### Order Tracking System

<table>
<thead>
<tr>
<th>Customer</th>
<th>Order</th>
<th>OrderDetail</th>
<th>Item</th>
<th>protocol</th>
<th>amount</th>
<th>credit</th>
<th>cash</th>
<th>check</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>status</td>
<td>calcTax</td>
<td>calcTotal</td>
<td>calcWeight</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protocol</th>
<th>Payment</th>
<th>Order</th>
<th>line item</th>
<th>shippingWeight</th>
<th>priceForQuantity</th>
<th>weightForQuantity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>amount</td>
<td>date</td>
<td>status</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Credit</th>
<th>Cash</th>
<th>Check</th>
<th>protocol</th>
<th>Payment</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>type</td>
<td>expDate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>authorize</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>bankID</th>
<th>authorize</th>
</tr>
</thead>
</table>

---

### Dependency Relationships

- PhotoMatrix
  - PhotoCollection
    - PhotoSlice
      - extract(slice: PhotoSlice) -> FeatureVector

- FeatureExtractor
  - extract(photo: Photo) -> FeatureVector

- IntensityExtractor
  - extract(slice: PhotoSlice) -> FeatureVector

- RGBExtractor
  - extract(slice: PhotoSlice) -> FeatureVector

- QuadExtractor
  - extract(slice: PhotoSlice) -> FeatureVector

### Exercise: Texas Hold'em Poker Game

- 2 to 8 human or computer players
- Each player has a name and stack of chips
- Computer players have a difficulty setting: easy, medium, hard
- Summary of each hand:
  - Dealer collects ante from appropriate players, shuffles the deck, and deals each player a hand of 2 cards from the deck.
  - A betting round occurs, followed by dealing 3 shared cards from the deck.
  - As shared cards are dealt, more betting rounds occur, where each player can fold, check, or raise.
  - At the end of a round, if more than one player is remaining, players' hands are compared, and the best hand wins the pot of all chips bet so far.

### Exercise: Texas Hold'em Poker Game

- What classes are in this system?
  - What are their responsibilities?
  - Which classes collaborate?
- Draw a class diagram for this system.
- Include relationships between classes (generalization and associational).
Class Diagrams Wrap Up

+ Discover related data and attributes
+ Get a quick picture of the important entities in a system
+ See whether you have too few/many classes
+ See whether the relationships between objects are too complex, too many in number, simple enough, etc.
+ Spot dependencies between one class/object and another

Class Diagrams Wrap Up

• But...
  - Can't discover algorithmic (not data-driven) behavior
  - Can't see steps for objects to solve a given problem
  - Can't understand the app's overall control flow (event-driven? web-based? sequential? etc.)