CS2704

Topic
Class Design

Outline
- Identifying classes
- Categories of classes
- CRC Cards
- Documenting class design
- Implementing classes
- Evaluating class design

Object-Oriented Design
- Identify objects and classes
- Identify how objects interact in system
- Identify hierarchies of related classes

Identifying Objects and Classes
- Study features of system
- Look for nouns (people, places, things)
- Example features:
  - “add course grade to student record”
  - “enter rental equipment description”
  - “add frequent flyer miles to customer record”

Categories of Classes
- Tangible things – from problem domain
- System interfaces and devices
- Agents – objects to carry out operations
- Events and Transactions – something done
- Users and roles – a system user
- Systems – overall system
- Containers

Class Relationships
- Aggregation – containment of objects by other objects
- Association – use of an object by another
- Inheritance – objects share properties
CRC Cards

- When designing a class identify
  - Components – data fields
  - Responsibilities – methods
  - Collaborators – classes of objects that interact with
- Use a “card” – 3x5 card or piece of paper

Card Layout

- Front:
  - Responsibilities
  - Classes that might be needed by methods
- Back:
  - Fields

Example CRC Card (front)

<table>
<thead>
<tr>
<th>Class: Mailbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations:</td>
</tr>
<tr>
<td>Get current message</td>
</tr>
<tr>
<td>Play greeting</td>
</tr>
<tr>
<td>Relationships:</td>
</tr>
<tr>
<td>Message, MessageQueue</td>
</tr>
</tbody>
</table>

Example CRC Card (back)

Fields:

- Queue of new messages
- Queue of deleted messages
- Greeting
- Extension number
- Pass code

Using CRC Cards

- CRC cards are used to begin the design process
- Identify classes and possible relationships
- Use in group
  - Consider major features of system
  - Each person plays a role in scenario
  - Rotate roles
- Use to evaluate possible designs

Class Notation

- Common notation for class

<table>
<thead>
<tr>
<th>Passenger</th>
<th>Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Fields</td>
</tr>
<tr>
<td>Address</td>
<td></td>
</tr>
<tr>
<td>Reservation</td>
<td></td>
</tr>
<tr>
<td>changeReservation</td>
<td>Operations</td>
</tr>
</tbody>
</table>

- List only important fields and operations
Documenting Class Design

- CRC Cards and design diagrams are not enough to guide implementation of class
- Can use forms to document classes and operations for programmer
- Forms should not include decisions on types for fields, etc. – designing is not coding!
- Forms can be embedded in code as comments

Class Description Form

- Class Name
- Base class(es) - only when using inheritance
- Purpose
- States
- Constructors
- Operations
  - Mutators
  - Accessors
- Fields

Operation Description Form

- Prototype
- Purpose
- Receives
- Returns
- Remarks

Class Implementation

Implementing class may involve
  - Choosing types for fields
  - Choosing data structures for containers
  - Choosing approach to link objects
  - Programming methods
  - Adding private data and methods to help with other methods
  - Programming memory management functions

Evaluating Class Designs

- Is class general enough for future needs?
- Are there too many relationships between classes?
- Does class have too much or too little responsibility?
- Are responsibilities unused or unrelated?
- Are responsibilities equally abstract?
- Are names appropriate?