Announcements

• Program 1 due Wednesday 9/29
• Homework 2 due Friday
• Exam 1 on Monday 9/27

Material

• Object-oriented design: Class identification
• CRC
Problem to Code

Three phases of software development:
1. Analysis
2. Design
3. Implementation

- Analysis Phase: produces the functional specification of the system detailing a precise description of the tasks the system must perform.
- Design Phase: results in the identification of the system classes, the classes’ responsibilities, and the relationships between the classes.
- Implementation Phase: tasks involve the coding of the classes, testing and system deployment.
In class exercise

The description of the project is (briefly):

Design a music catalog system to store information about music recordings. The system must support adding recordings, storing information about artist, album title, song title, song composer, etc. The user of the system should be able to search the collection for any information. It should also allow the user to browse the collection by recording title, by artist, and by year of recording. This program reads the recordings from a file stored locally on disk.
Identifying Objects

- Abbott & Booch approach:
  - Identify nouns, pronouns, noun phrases in the system specification
  - singular \(\rightarrow\) object, plural \(\rightarrow\) class
  - Other (non-noun) classes must be identified:
    - Tangible things, agents (operation classes), events/transactions, users/roles, interfaces/devices, sub-systems.
  - verbs identify services
  - CRC Class-Responsibility-Collaboration cards
Once a set of potential objects is determined

- Identify their attributes
  - these are usually expressed as data members

- Identify their responsibilities
  - these are services that the object provides, usually methods

- Identify their relationships to other objects
  - these help the object/class accomplish their responsibilities, that is, this object calls other objects
CRC Cards

- CRC = Class-Responsibility-Collaboration cards
- Take index card and use them in your design.
- Write name of class at top
- Write responsibility on left
  - these tend to be methods
- Write collaborator on right
  - member variables

<table>
<thead>
<tr>
<th>Class: PUT NAME HERE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Responsibilities</td>
</tr>
<tr>
<td>list responsibilities</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### WebLink Class

<table>
<thead>
<tr>
<th>Responsibilities</th>
<th>Collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store URL &amp; link</td>
<td>URL, string</td>
</tr>
<tr>
<td>Provide Access to path</td>
<td>string</td>
</tr>
<tr>
<td>Provide Access to name</td>
<td>string</td>
</tr>
<tr>
<td>Provide Access to server</td>
<td>string</td>
</tr>
<tr>
<td>Determine validity</td>
<td>URL, URLConnection</td>
</tr>
</tbody>
</table>

Notice that there is no mention here of the fact that WebLinks are stored in a collection or are used in the GetLink method, that is outside of this class.
Identify Objects

• Break up in groups of 2/3 and identify objects in this problem description

• Design a music catalog system to store information about music recordings. The system must support adding recordings, storing information about artist, album title, song title, song composer, etc. The user of the system should be able to search the collection for any information. It should also allow the user to browse the collection by recording title, by artist, and by year of recording. This program reads the recordings from a file stored locally on disk
Example: Music System

- Reject (for now)
  - music (refers to the type of information stored, but nothing musical stored)
  - catalog (collection, system) - all the same
  - information - general name to refer to pieces of the collection
  - user - external to the system, plays a role in the system
Other points

• Data structure support required... (organization of elements in domain)
  • Catalog has a collection of recordings
  • Recording has title, artist name, list of songs, year of recording
  • Song has title, composer name, artist name
  • What relationships exists here?
Good Enough?
Better?
Identifying Responsibilities

- Look for verb in the requirements document — usually this will define services of the object of the sentence.
- E.g. Quarterback throws the ball. This defines a service for the ball, provided by the quarterback.

```java
class Quarterback {
    void throws(Ball b) {}
}
class Ball {
    something...
}
```

- Look at each feature — require services of many objects.
Specifying Responsibilities

• Name the service to match the external request for the service (name is intended to be on the interface, make it reflect that).
  • FindArtist or GetCompositions(Artist)
  • ThrowsBall
• Identify the information and/or entities necessary to provide the service.
  • artist name, composer name, composer's songs, recording list, etc.
  • these come as arguments to the service or as values already stored in the object
• Identify the responses, if any, that the service will generate.
  • success, failure, artist name, song name, list of songs, etc.

• BE CAREFUL: separate I/O from responsibilities
  • Unless the responsibility is to print something out
  • DIFFERENCE between "get me something" and "print me something"