08 - Object Design Categories

February 1, 2023

COMP2404

Darryl Hill

Contents

Overview

We will provide one approach to software engineering

- Not the only approach

What are object design categories?

- One possible starting place for organizing objects
- Most objects would fall into one of these categories
  - Possibly more than one
- Each category has specific responsibility

Advantage:

- Easier to think of classes and their responsibilities once we narrow the scope
- Classes should
  - have a single purpose within their category
  - be reusable

Types of Object Categories

We will use four object categories:

- Entity objects
- Control objects
- Boundary objects
  - Sometimes called View or User Interface (UI) objects
  - On the boundary between our code and the user
    - User can be a person, a web service, a library, etc
- Collection objects

Types of Object Categories

Though this appears similar to Model-View-Controller (MVC), it is not the same

MVC is a specific design architecture meant to solve a specific problem

- Using the observer design pattern

Object categories are simply broad categories of objects

- Meant to help us identify what objects we need based on categories that (nearly) all applications have
- These categories become more well-defined in larger programs
  - In smaller programs we often have classes in multiple categories

Entity Objects

- These should model real-world objects or information tracked by the program
- Often represent persistent information (information that is saved between sessions)
- Examples: `University`, `Student`, `Instructor` classes
- Information stored in member variables
- Functions contain small amounts of application logic
  - Entity specific, not application specific
  - i.e. `lessThan`
- Usually the simplest form of object

Control Objects

Objects in charge of program flow

- What code is executing?
- What classes are interacting / sharing information?

Typical OO main function has two lines of code

- Create a control object
- Launch a function on the object
- This is a top-level control object

Control object takes charge of program flow

- Will often manage high-level interactions between other classes

Boundary Objects

Boundary objects manage the interaction of the application with foreign entities

- Other programs
- Users

Could be:

- An API
- A user interface

Ideally only boundary objects interact with outside entities

- Makes it easy to swap out a user interface
- Switch from console to GUI, for example

Collection Objects

Storage of multiple entities of the same type

- Along with relevant behaviours, i.e., sorting, retrieving, processing

Collections may be primitive

- Primitive arrays
- Very little associated behaviour
- Little to no error checking

Or a class

- Often backed by some other collection
- Array or another class

Collection Objects

For example we can create a class that has an array member variable.

The array is the actual collection, but remains hidden.

Our class provides an abstraction layer of common collection functions.

We can add

- behaviour (accessing, sort, find, etc)
- error checking (bounds checking, NULL checking, etc)
- automatically growing or shrinking array when necessary
- creation and deletion

The array is known as the **backing array**.

- The `vector` class uses a backing array and a generalized interface.

Collection Objects

Why not just use arrays?

- Difficult to work with
- Error prone
- Static
  - Collection classes often grow or shrink as needed
  - You'll do this in COMP2402, not in this class

Why do WE use so many arrays?

- Practice - knowledge of arrays is a necessary skill
- Understanding the process allows you to use other data structures like `vectors` effectively
- Can write our own specialized data structures
  - Specialized = faster and more resource efficient

Collection Objects

Collection objects are an excellent example of good OO-programming

- They provide encapsulation and an abstraction layer to data access
- Provide a simple interface with implementation details hidden away
  - Add x
  - Remove y
- Error checking, actual data access, etc, is handled away from the user

Programming example <p1>

Object Categories

Example: A `University` application:

- The `Student` class is an `entity` object
  - Store basic information about students
- A `View` class would handle user interactions
  - `boundary` object
  - interacts with HR at a computer terminal
- A `University` class would be a `collection` object
  - it contains a collection of `Student` objects
  - uses the `Array` to manage details
  - In more complex apps, the `University` might also be a `control` object
- A `Controller` class would be the `control` object
  - Manages the interaction between `View` and `University`

coding example <p2>

Object Categories

We can also see the heirarchy of responsibility of managing memory.

- `University` should delete the `Students`, not the `List`.