Do all those patterns look the same?

Chapter Content

- Repeating motive
- Differences
- Discussion

Repeating motive

InterBit Training & Consulting Ltd.

- Wrapper class holds some instance and forwards operations to it (Adapter, proxy, decorator, bridge...).
  - Advantage: flexibility.
    - The enclosed instance may be any sub-class.
  - Disadvantage: wrapper object is not the exact same thing as the enclosed class.
    - Might hinder type-checks.
    - Inexperienced RMI coders may synchronize on the proxy (stub) instead of the real remote object.

Difference #1

InterBit
Training & Consulting Ltd.

- When attempting to recognize a design pattern, consider:
  - Does the wrapper class both extend and contain the enclosed class?
    - Patterns: Composite, Decorator, Proxy.
    - So wrapper class can be used whenever the enclosed class is expected.
    - Allows recursive composition.
  - Or, does it only contain it?
    - Patterns: Object Adapter, Bridge.

Difference #2

InterBit Training & Consulting Ltd.

- Programmer's intention or the problem they tried to solve. (Not necessarily detectable by technical means).
  - Over-simplified summary:
    - Adapter - new method names.
    - Bridge - combine different abstractions and different implementation.
    - Composite - tree structure.
    - Decorator - enhances the enclosed object's functionalities.
    - Proxy - control access to an object.

Summary

InterBit
Training & Consulting Ltd.

- **Adapter** - used to change the interface of one class to that of another one.
- **Bridge** - intended to keep the interface to your client program constant while allowing you to change the actual kind of class you display or use. You can then change the interface and the underlying class separately.
- **Composite** - a collection of objects, any one of which may be either itself a Composite, or just a primitive object.
- **Decorator** - a class that surrounds a given class, adds new capabilities to it, and passes all the unchanged methods to the underlying class.
- **Proxy** - which provides a simple place-holder class for a more complex class which is expensive to instantiate.

Discussion

- Based on criterion #1, we note:
  - **Similarities between Bridge and Adapter**
    - However, bridge is for cases in which the designer foresees the inheritance evolution *in advance*, while adapters fix existing things (e.g. old API) on need.
  - **Similarities between Proxy, Decorator and Composite**
    - They aim to solve different problems.
    - Note Composite may encapsulate several parts.

Proxy and Related Patterns

InterBit Training & Consulting Ltd.

- Both the **Adapter** and the **Proxy** constitute a thin layer around an object. **But**, the Adapter provides a different interface for an object, while the Proxy provides the same interface for the object, but interposes itself where it can save processing effort.
- A Decorator also has the same interface as the object it surrounds, but its purpose is to add additional (usually visual) functionality to the original object. A proxy, by contrast, controls access to the contained class.