Activity 13-1 - Design Patterns

1. **What is a *pattern*?**

   - What Merriam Webster says[1]
   - Why are patterns *useful*?
     - They provide shortcuts.
     - They are a type of generalization.
     - Useful for reasoning.
     - Human brains are wired to recognize patterns, often to the point of finding patterns where there aren't any.
   - Our definition:
     - **pattern** ~ a *named* solution applied more than once to a problem scenario when it occurs in a particular context
     - i.e., a known strategy for solving a problem
   - Examples: Bad (but useful) driving patterns
     - "Pittsburgh Left"
     - "Texas Exit"

2. **Design Patterns (for software developnment)**

   - Known solutions to problems that frequently arise in software development.
   - Goal of most design patterns w.r.t. software development –> dependency management.

3. **Types of design patterns (for software development)**

   - **creational pattern** pattern that helps with the creation of an instance of an object; **example: Factory pattern**
   - **structural pattern** pattern that helps with the setup of relationships (i.e., communication pathways) between different groups of objects (entities); **example: Decorator pattern**
   - **behavioral pattern** pattern that helps with the partitioning of system behaviors into discrete classes; **examples: Observer pattern, Command pattern**

4. **First design pattern: *Command pattern***

   - **command** (in software development) ~ an object whose job is to store all the information needed to execute a particular action. See https://www.baeldung.com/java-command-pattern.
   - **Command pattern** ~ an interface, typically named `Command`, with a single method, typically named `Execute`.

[1] https://www.merriam-webster.com/dictionary/pattern

- Useful for:
  - decoupling WHAT is done from WHO does it (i.e., it decouples actors from actions)
  - decouple WHAT is done from WHEN it is done (temporal decoupling)
  - implementing “undo” functionality
- How is the Command pattern useful for “undoing” commands?
  - Commands are objects and objects can hold state; so a comman object that executes a function could save information about what it did (i.e., it remembers the relevant details about its actions); this means it can undo those actions at a later time
- Example: imagine creating an interface called `Task` with a single method, `execute`
- Create a derivative of command object for every button in the pallete.
- Advantages: makes our code extensible because we can add new commands without changing existing code (Supports open-closed principle)

5. **Components of the Command pattern**

   - command (see above)
   - receiver
   - invoker
   - client

   From Wikipedia[2]:

   In object-oriented programming, the command pattern is a behavioral design pattern in which an object is used to encapsulate all information needed to perform an action or trigger an event at a later time. This information includes the method name, the object that owns the method and values for the method parameters.

   Four terms always associated with the command pattern are command, receiver, invoker and client. A command object knows about receiver and invokes a method of the receiver. Values for parameters of the receiver method are stored in the command. The receiver object to execute these methods is also stored in the command object by aggregation. The receiver then does the work when the execute() method in command is called. An invoker object knows how to execute a command, and optionally does bookkeeping about the command execution. The invoker does not know anything about a concrete command, it knows only about the command interface. Invoker object(s), command objects and receiver objects are held by a client object, the client decides which receiver objects it assigns to the command objects, and which commands it assigns to the invoker. The client decides which commands to execute at which points. To execute a command, it passes the command object to the invoker object.

6. When should you use the Command pattern?

   - When you need undo/redo capabilities

[2]https://en.wikipedia.org/wiki/Command_pattern

- When requests need to be handled at various times or in various orders

7. Example: https://gameprogrammingpatterns.com/command.html