ECE 452/CS 446 D4 Design Activity
Strategy Pattern

Team Members
- Derek Trider
- Quan Zhang
- Andy Yin

Overview
Strategy is a design pattern comes under behavioral software design pattern, which allows changing a class behaviour or its function implementation (algorithm) at runtime. The strategy design pattern contains a collection of predefined and encapsulated algorithms. The algorithms can be used interchangeably within the collection based on specifications and requirements.

A typical Strategy design pattern usually contains a strategy interface, various concrete strategy objects (algorithm) and a context object. The context object usually has a strategy object and the behavior of the context varies based on its strategy objects.

Vocabulary
1. **Strategy**
    a. **Strategy Interface**
        - An interface between context and concrete strategy objects
        - Common functions are defined in the interface like “numeric operation”
    b. **Concrete Strategy**
        - A concrete implementation of an algorithm
        - A concrete implementation of strategy interface
        - **Example**: Add Operation, Subtraction Operation

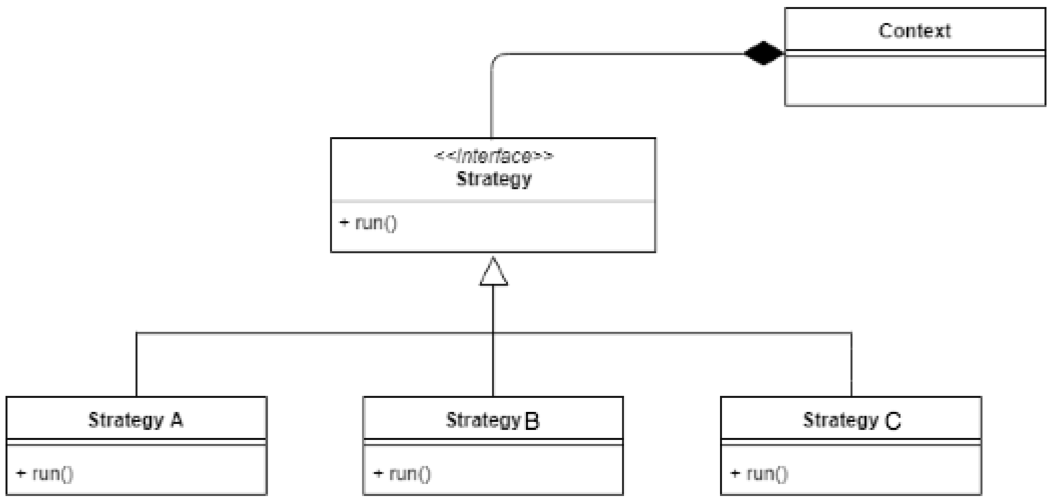
2. **Context**:
    - Contains a reference to a concrete strategy object
    - Received requests from clients and invoke corresponding concrete strategy

Purpose/Motivation
1. Reduce coupling between a strategy and the client invoking it. The pattern allows the algorithm varying independently from the clients that use it.
2. Provide convenience to add new features, functionalities to an existing program with minimum modification of client code structure.
3. Allow interchangeable class behaviours at runtime based on specifications and requirement.

Intended use cases

1. When the number of parallel functionalities (algorithms) of the program is likely to scale up.
2. When the behaviour of classes need to be changed at runtime and the client or the context has a good knowledge of which algorithm to choose.

Specific Structure



Advantage

1. Class behaviour can be decided and changed at runtime based on scenarios, which provides more flexibility to the program.
    - In the Numerical Operation example, all the potential operations can be added as a strategy and a specific algorithm will be chosen at runtime.
2. Each algorithm is encapsulated in a strategy class, which decouples the algorithms and the clients. In other words, it provides convenience to add additional functionalities with minimum client code modification.
    - An user can easily add a multiply operation with little change in context class
3. Algorithms can be used by multiple clients, which increases code usability.
4. Avoids excessive use of “switch” and nested “if-else” statements.

Disadvantage

1. Clients are required to have prior knowledge about which strategy to be chosen.
2. Strategy Interface must expose to all the clients, a failure in the strategy interface may affect all the clients using it.

3. Introducing new algorithms may result in unnecessary old algorithm accumulation.

Non-functional Properties
Supports:
1. Scalability: It is easy to add additional functionality without major code structure changes.
2. Reusability: Multiple contexts can use the same strategy, which reduces repetitive code.
3. Security: Strategies can be isolated. If an isolated strategy has a security issue, then it should only affect systems that the use that particular strategy.
4. Complexity: Algorithms are decoupled from the classes, and implemented in the strategies.

Inhibits:
1. Security: Having multiple strategies may introduce additional security vulnerabilities. A security issue in one strategy may compromise the entire system. Anonymous strategies may pose security risks if they are poorly implemented or malicious. Insecure, and legacy strategies that cannot be refactored or removed pose a permanent security vulnerability.
2. Performance: The performance of program is decreased due to bad spatial locality, as algorithms are not alongside the classes that require them. Context needs to invoke the concrete strategy to apply the algorithm.