*Name:* *CSC-490, Winter 2020*

# Strategy Pattern

In order to complete this worksheet you will need the files located in the folder `Strategy` in CoCalc.

1. (Motivating Example) Suppose you are working on a program for some video game. You are tasked with designing a class called `BadGuy` which has methods `fight()` and `chase()`. The game will have a variety of bad guys who fight and chase in a variety of ways (polymorphism!). For instance, you know there will be a pirate who fights with a sword and chases by boat; and there will be a ninja who fights with a kick and chases by running.

   (a) (10pts) Draw a class diagram for `BadGuy` and other classes that could be implemented to complete your task. In particular, your design should allow for creating bad guys that act like pirates and ninjas. Leave space so that you can add more to the diagram later.

*last updated: January 8, 2020*

(b) (4pts) Your boss tells you that in addition to ninjas and pirates, you will also need the ability to create all sorts of other bad guys. In particular, you need the ability to create a bad guy whose fighting and chasing behavior are any combination of the following behaviors:

| | | | |
|---|---|---|---|
| *fights with:* | sword | kick | gun |
| *chases by:* | boat | running | |

Note that there are 6 possible combinations of behaviors. For instance, you could have an admiral who chases by boat and fights with a gun, or some bad guy who chases by foot and fights with a gun. Modify your class diagram from part (a) to allow for all 6 behavior combinations.

(c) (3pts) Suppose that now you want to add a new method `reactToHit()` to each bad guy. There are two possible implementations for this new method: either the bad guy dies, or retreats. You are supposed to allow for the creation of a bad guy with any combination of fight, chase, and reactToHit behaviors. How many classes need to be added or modified to your design to make this change?

(d) (3pts) How many classes need to be modified if you need to change the implementation of running?

(e) (10pts) Use the *strategy pattern* to draw a class diagram that could be implemented to complete your task (without the `reactToHit()` method).

2. According to the GoF, the intent of the strategy pattern is to *"Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it."* For general discussions about the strategy pattern the GoF refers to class that uses the algorithm as the `Context`.

   (a) (7pts) Draw a class diagram for a strategy pattern involving a class `Context` which has method `method()`. There is also an abstract class `Strategy` with an abstract method called `algorithm()`, and some concrete subclasses `ConcreteStrategyA` and `ConcreteStrategyB`. Finally, `Context` *has a* `Strategy` which it uses in order to implement `method()`.

   (b) (3pts) Which class in part 1(d) is playing the role of `Context`? Which are the abstract and concrete strategies?

   (c) Look at one possible implementation of the class diagram above in the CoCalc folder `Strategy/Context`. Note that in that program all the methods and algorithms are void functions without parameters, but this is not always the case. Also, the implementation of the context's `method()` in that example simply calls the strategy's `algorithm()`. In general, the implementation of `method()` should make some use of the strategy's `algorithm()`, but the way in which the algorithm is used varies depending on the particular program. Also, the given implementation uses a constructor to assign a strategy. There are other ways to assign a strategy to the context; for instance one could use a setter.

3. Suppose you have a class called `List` that has a data member called `numbers` and a method `sort()` which sorts the numbers into increasing order. There are a variety of sorting algorithms that could be used to implement `sort()`. Use the strategy pattern to allow for different sorting algorithms to be used. Include a method in `List` called `setSortStrategy()` that can be used to set the strategy.

   (a) (10pts) Draw the corresponding class diagram with two concrete sorting strategies called `BubbleSortStrategy` and `QuickSortStrategy`.
   Include pseudocode for the implementation of the `sort()` method in `List`.

   (b) (10pts) In the CoCalc folder `Strategy/List` you'll find a partial implementation of this story. There are a couple missing parts to the program. First off, the concrete sorting algorithms are not implemented properly; at this point they just print a statement saying they are not yet implemented and then they add a number to the end of the list (this is to check that the sorting algorithms can modify the list). The goal of this part is **not** to implement the sorting algorithms. Instead, your job is to properly implement the method `sort()` in the class `List`. When you finish, compiling[1] and running[2] your program should result in the following terminal output:

```
8 7 2
-------------------------------------------------------------
Bubble sort is not yet implemented...
Instead, bubble sort just adds a 0 to the end of your list
-------------------------------------------------------------
8 7 2 0
-------------------------------------------------------------
Quick sort is not yet implemented...
Instead, quick sort just adds a 1 to the end of your list
-------------------------------------------------------------
8 7 2 0 1
```

[1]I compile using the command `c++ -std=c++11 -Wall -o list *.cpp`

[2]Then run using the command `./list`

4. (10pts) Describe another situation where the strategy pattern can be used. You can either think of your own example, or look up some online and pick your favorite. Your example should be significantly different than the examples covered in this worksheet and the one given in the GoF book. In addition to a brief written description of your example, you should draw a class diagram. If you did not come up with your example on your own, state where you found it.