

Updated diagram. Still unfinished, but closer to what I want.

Using Singleton pattern for Game class. This way only one instance exists of the program and needs to exist while everything runs. Hence **instance: Game** and using **static Game\* instance** in the Game class. getInstance should also be implemented to adhere to Singleton pattern.

Remember – class methods need to have return types which are currently missing.

Logic for picking up card could be getting deck card at position 0, assigning that to a pointer called pickedUpCard for example, then adding that to bank or play area or whatever other logic.

My understanding of enum is that it’s a way to quickly check the type of card, and compare/determine if a card is a duplicate. Type() returns a CardType – CardType is the enum. So for Kraken, you would just return Kraken for type() (that’s the idea) which accesses the enum value. Also easier to compare two cards then. Could do like card1->type() == card2->type() etc.

Card ability is string for now. Subject to change – I think the play() function should be able to utilise the ability to do something. Logic undecided

Virtual method – can be overridden (for abstract classes)

Remember that outputting a card requires both the deck/collection as well as the name of the area where the cards belong like deck/play area. So you need to provide both the actual card collection (aka deck or play area) + a string to output. Subject to change.

Use a deck (and play area, bank, discard area etc) class – deck has a card collection that represents all of the cards and then player can have a relationship with the class. Maybe this isn’t necessary? As above, outputting a card can be done by getting a str() method from the card itself, and iterating over all cards in a certain collection. That collection can be held by a class as an attribute (of class collection) rather than its own class.