

FILE 1

"Parent" Class

Public:

```
struct DrawCard {  
    string Number  
    string Suit (S, C, D, H)  
    string code for printing purposes  
    ostream operator<< ( )  
        - Overload insertion &  
        - Format for response  
}
```

- Constructor
- Destructor
- DrawCard* GetSuits (char s) virtual s = suits
- DrawCard* GetValue (int n) virtual n = number
- DrawCard* DrawHand () virtual
- ostream
- Template<T> BubbleSort
- void PrettyPrint virtual if needed





Protected:

- Data (same as before) virtual
- EndOfData () virtual

Private:

```
vector<char> JsonData  
DrawCard ** Card  
CardsDealt = 1
```

KEY:

-  = side notes
-  = skeleton actions/comments
-  = connects to Compare Structs
-  = using template

FILE 2

```
struct CompareSuits<T> {
```

```
    bool Compare (DrawCard* C1, DrawCard* C2)  
        - auto S1 = setSuits(C1.suits)  
        - auto S2 = setSuits(C2.suits)  
        - If (C1 == C2)  
            return CompareNumbers(C1, C2)  
        - return ((S1 > S2) ? True : False)
```

```
    int setSuit(string suit)
```

- Set the values of suit
- Maybe do this → Switch (cardSuit)
 - case H: 4
 - case D: 3
 - case C: 2
 - case S: 1

```
struct CompareNumbers<T>
```

```
    bool Compare (DrawCard* C1, DrawCard* C2)  
        - auto N1 = C1.numbers  
        - auto N2 = C2.numbers  
        - If (N1 == N2)  
            return CompareSuits(C1, C2)  
        - return ((N1 > N2) ? True : False)
```

FILE 3

- MAIN:
- Mini Games call dif. parameters for API
 - #1 - no jokers
 - #2 - include jokers
 - Use small functions to call into main
 - "jokers_enabled" "true"
 - Joker: code = X1, X2
value = JOKER
suit = RED, BLACK
 - Make inputs all uppercase (exception)

FILE 4

MiniGame 1

Private:

int userGuess
DrawCard* Card;

Public:

DrawHand (DrawCard* card)

- Draw card @ index[i]

GetGuess

- send guess to CompareXXX

CheckGuess

- do {

if (guess != answer)

if (guess > answer)

else if (guess < answer)

else throw "invalid entry"

else if (guess == answer)

} while (guess <= 5)

- return SendResponse?

SendResponse?

- You won

- You lost → max guesses reached

FILE 5

MiniGame 2

Private:

int numOfCards = 0

Public:

SetNumOfCards (int num)

- num → User asked for # of cards to draw

return numOfCards = num

DrawHand (DrawCard* card)

- Set max count of drawn cards

- exception handling

SearchForJoker

- Uses DrawHand values to iterate and check if any are Jokers

- If JokerFound = true,

"Joker found, redrawing card"

DrawCard* GetSuits (char s)^{s = suits}

DrawCard* GetValue (int n)^{n = number}

CompareNumbers <T>

- receives card numbers

- BubbleSort

CompareSuits <T>

- receives card suits

- BubbleSort