To facilitate the making of Texas Hold’em, we decided to create a class library called **Holdem** that contains all the classes necessary to implement a successful poker game. Below is a list of all the classes (with a list of their public methods and constructors) found in Holdem.

**Class: Card**

**Inheritance: None**

|  |  |
| --- | --- |
| **Constructors** | **Methods** |
| Card() | int rankToString(int rank) |
| Card(RANK rank, SUIT suit) | int suitToString(int suit) |
| Card(int rank, int suit) | override string ToString() |
| Card(RANK rank, SUIT suit, bool faceup) | void Highlight()/void unHighlight() |
| Card(int rank, int suit, bool faceup) | Various getter/setters & properties |
| Card(Card card) | Operator overload for ==, !=, < , >, <=, >=  the ranks of the cards will be compared |

RANK and SUIT are enums that represent rank (2-Ace) and suit (Diamonds-Spade)

**Class: Deck**

**Inheritance: None**

|  |  |
| --- | --- |
| **Constructors** | **Methods** |
| Deck() | void Add(Card card) |
| Deck(bool faceUp) | void Shuffle() |
| Deck(Deck otherDeck) | string Print() |
|  | int CardsLeft() |
|  | void Deal() |
|  | void Remove(int index) |
|  | void Remove(Card card) |

**Class: Hand**

**Inheritance: None**

|  |  |
| --- | --- |
| **Constructors** | **Methods** |
| Hand() | void Add(Card card) |
| Hand(Hand otherHand) | void Remove(int index) |
|  | void Remove(Card car) |
|  | int Count() |
|  | override string ToString() |
|  | void sortByRank() |
|  | void sortBySuit() |
|  | this[int index] (indexer for the cards in hand) |
|  | various getters/setters & propeties |
|  | Operator overload for ==, !=, < , >, <=, >=  the value of evaluated hands will be compared |

**Static Class: Hand Combinations**

**Inheritance: None**

|  |
| --- |
| **Static Methods** |
| Hand getBestHand() |
| bool isRoyalFlush() |
| Hand getRoyalFlush() |
| … (for each poker hand) |
| Hand getKickers() |

**Class: Table**

**Inheritance: None**

|  |  |
| --- | --- |
| **Constructors** | **Methods** |
| Table() | void RemovePlayer(Player player) |
| Table(PlayerList players) | void RemovePlayer(int index) |
|  | void StartNextMatch() |
|  | bool BeginNextTurn() |
|  | int incrementIndex(int index) |
|  | int decrementIndex(int index) |
|  | void DealHoleCards() |
|  | void PaySmallBlind() |
|  | void PayBigBlind() |
|  | void DealFlop() |
|  | void DealTurn() |
|  | void DealRiver() |
|  | void ShowDown() |
|  | void PlayerWon() |
|  | this[int index] (indexer for the list of players) |
|  | various getters/setters & propeties |

**Class: Pot**

**Inheritance: None**

|  |  |
| --- | --- |
| **Constructors** | **Methods** |
| Pot() | void AddPlayer(Player player) |
| Pot(int amount, PlayerList playersInPot) | void Add(int amount) |
|  | Various getter/setters & properties |

**Class: Player**

**Inheritance: None**

|  |  |
| --- | --- |
| **Constructors** | **Methods** |
| Player() | paySmallBlind(int amount, Pot mainPot) |
| Player(int buyInAmount) | payBigBlind(int amount, Pot mainPot) |
|  | Fold(Pot mainPot) |
|  | Call(Pot mainPot) |
|  | Check(Pot mainPot) |
|  | Raise(int raise, Pot mainPot) |
|  | Bet(int bet, Pot mainPot) |
|  | AllIn(Pot mainPot) |
|  | Reset() |
|  | CollectMoney(Pot mainPot) |
|  | Leave() |

**Class: AIPlayer**

**Inheritance: Inherited from Player**

|  |  |
| --- | --- |
| **Constructors** | **Methods** |
| AIPlayer() | public void MakeADecision(Pot mainPot) |
| AIPlayer(int buyInAmount, int difficulty, int playingStyle) | public void CalculateHandValue(int count) |
| AIPlayer(int buyInAmount, DIFFICULTY difficulty, PLAYINGSTYLE playingStyle) | void EasyThinking(Pot mainPot) |
|  | void MediumThinking(Pot mainPot) |
|  | void HardThinking(Pot mainPot) |
|  | void BetAmount(Pot mainPot) |

DIFFCULTY and PLAYING are enums representing integers

**User Interface and Test Programs**

The user interface for poker is quite straightforward. Most of the aspects of poker are handled by the computer. We are planning to create four buttons that the user can interact with to make his decisions. Before playing the game, the user has the option to login and customize various playing conditions (AI difficulty, number of players, name, ect…).

To be more organized, we are planning to implement 3 test programs to test various aspects of the game. Those aspects will be Evaluating Cards, Betting and the AI.