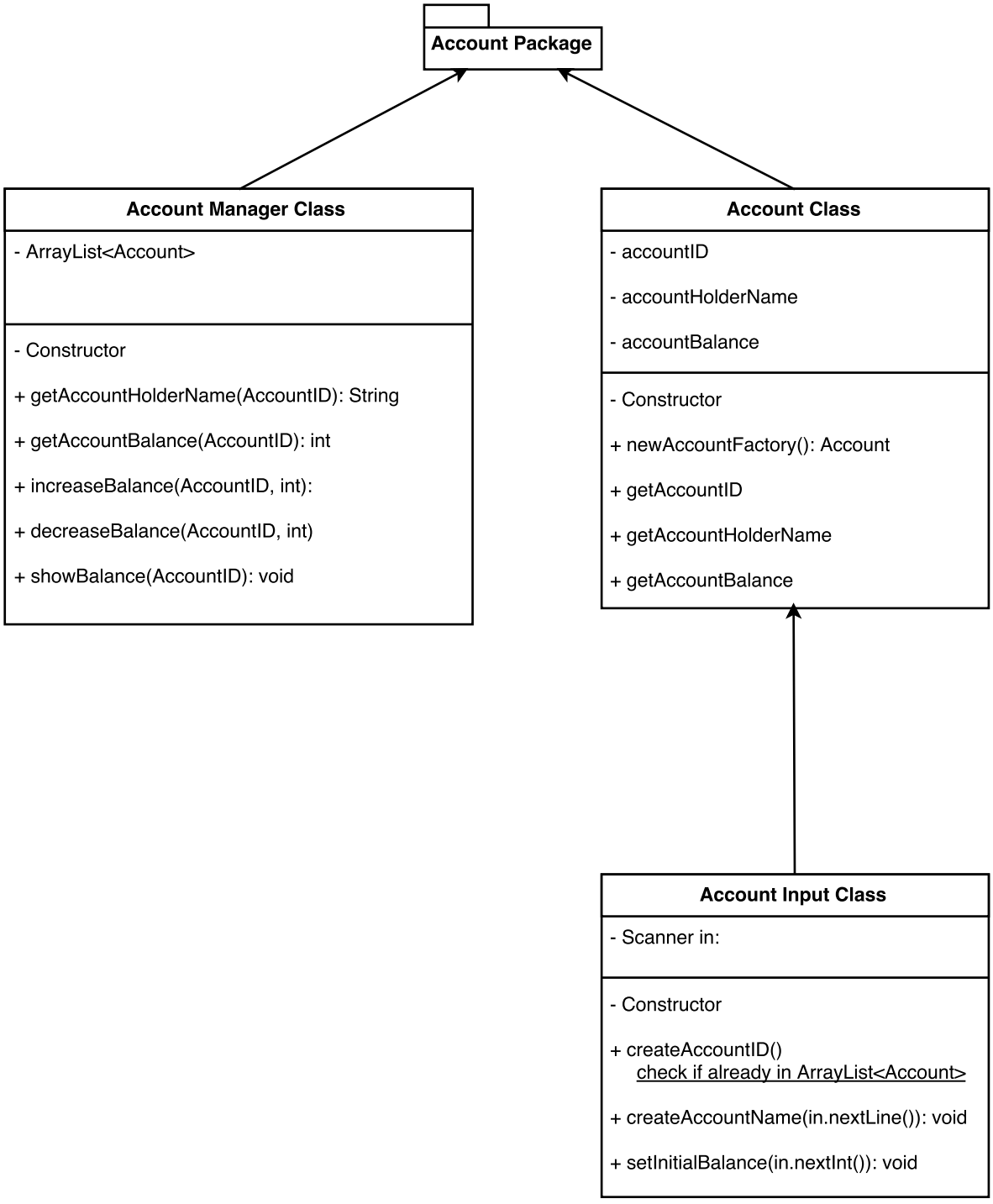
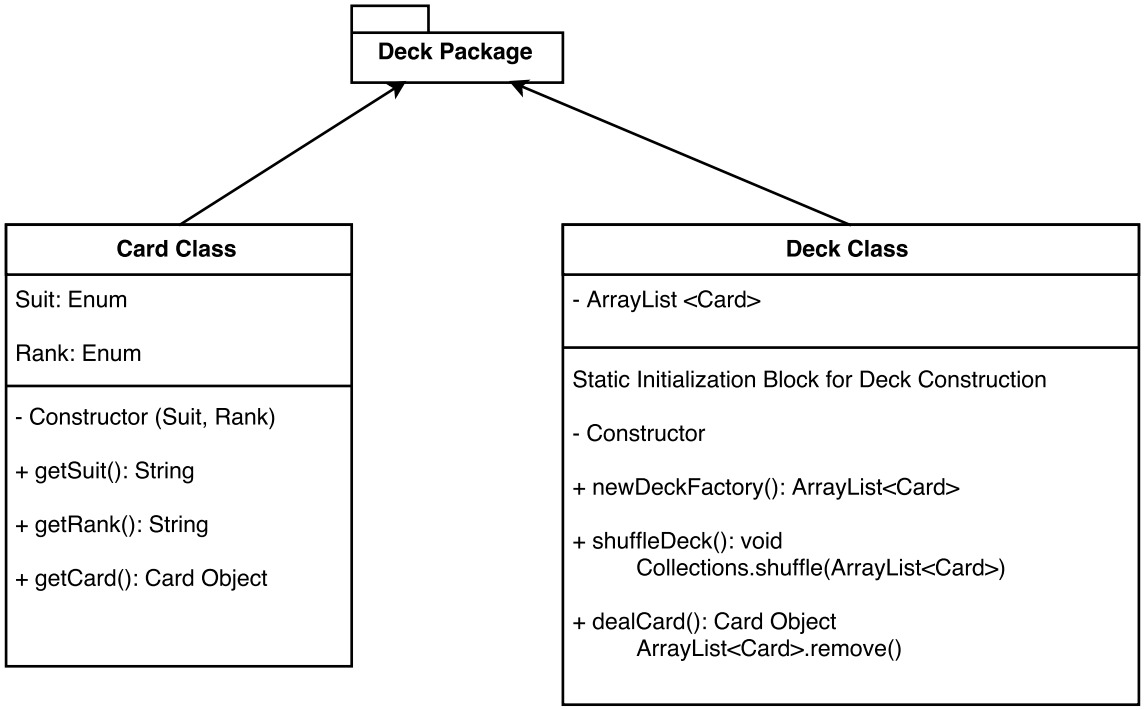
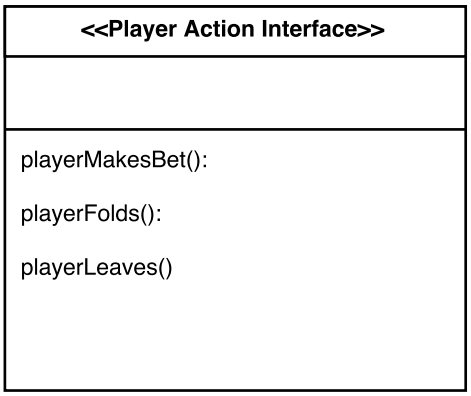
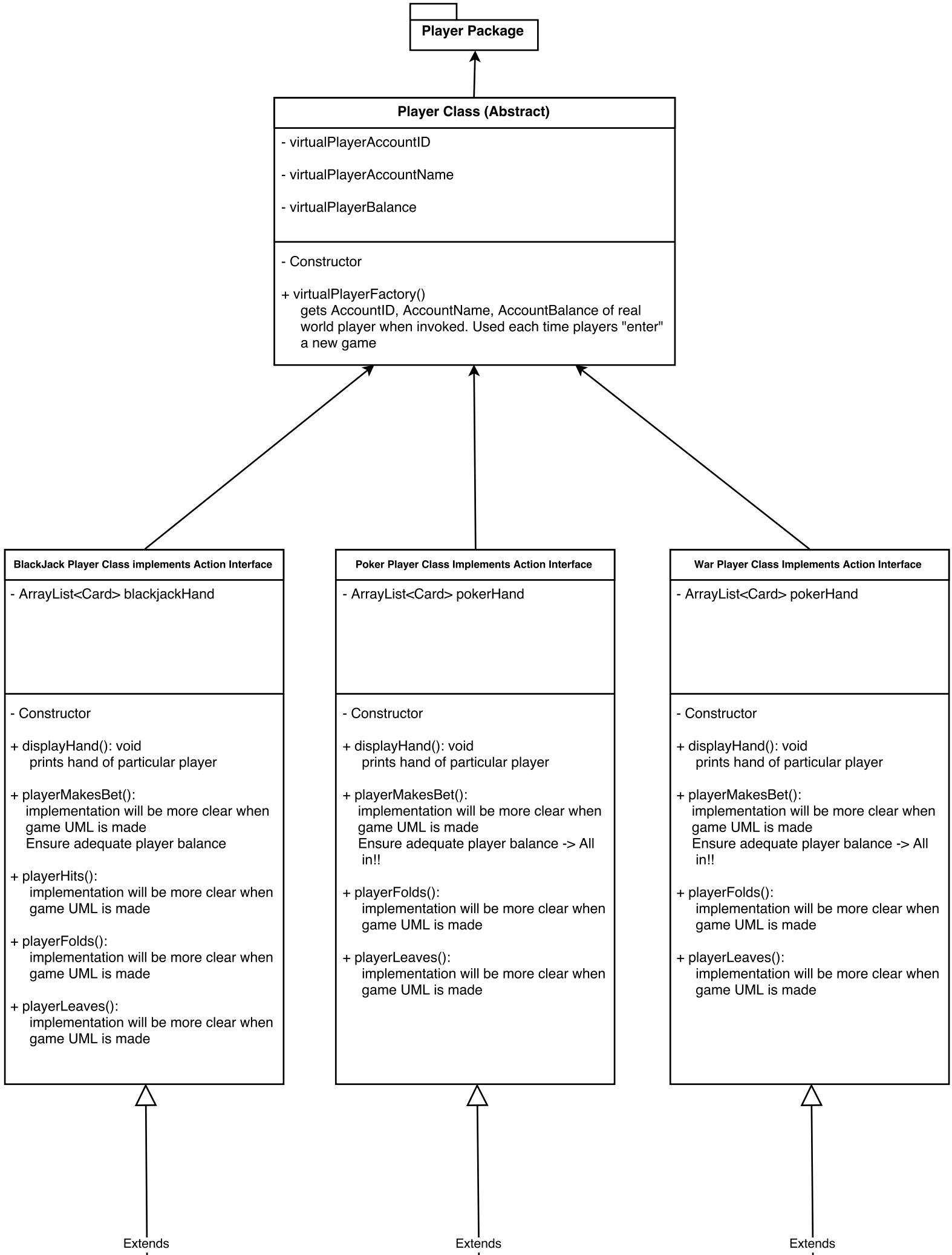
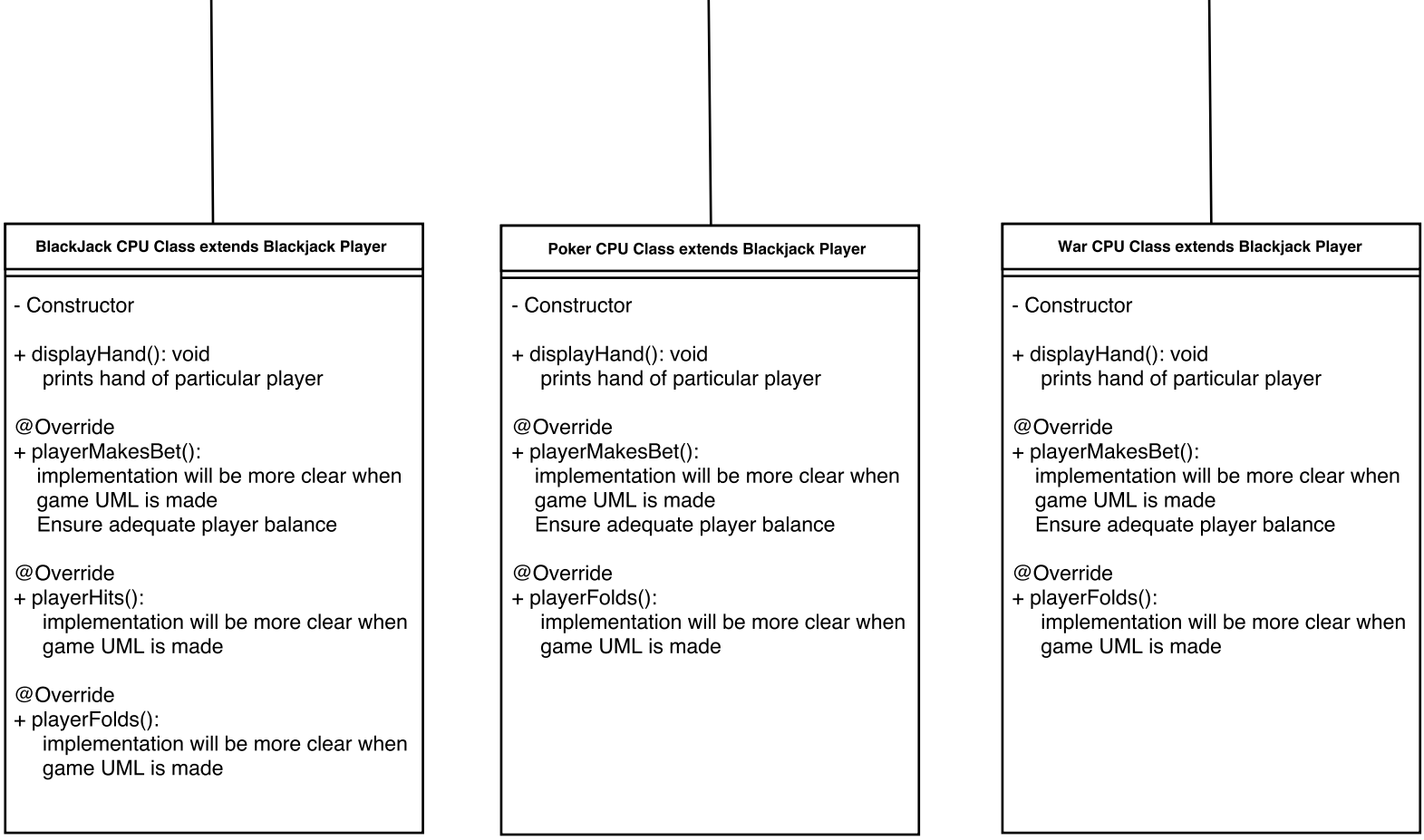

Account Package
Account Manager Class
- ArrayList<Account>
- Constructor
+ getAccountHolderName(AccountID): String
+ getAccountBalance(AccountID): int
+ increaseBalance(AccountID, int):
+ decreaseBalance(AccountID, int)
+ showBalance(AccountID): void
Account Class
- accountID
- accountHolderName
- accountBalance
- Constructor
+ newAccountFactory(): Account
+ getAccountID
+ getAccountHolderName
+ getAccountBalance
Account Input Class
- Scanner in:
- Constructor
+ createAccountID()
check if already in ArrayList<Account>
+ createAccountName(in.nextLine()): void
+ setInitialBalance(in.nextInt()): void


Deck Package
Card Class
Suit: Enum
Rank: Enum
- Constructor (Suit, Rank)
+ getSuit(): String
+ getRank(): String
+ getCard(): Card Object
Deck Class
- ArrayList <Card>
Static Initialization Block for Deck Construction
- Constructor
+ newDeckFactory(): ArrayList<Card>
+ shuffleDeck(): void
Collections.shuffle(ArrayList<Card>)
+ dealCard(): Card Object
ArrayList<Card>.remove()


<<Player Action Interface>>
playerMakesBet():
playerFolds():
playerLeaves()


Player Package
Player Class (Abstract)
- virtualPlayerAccountID
- virtualPlayerAccountName
- virtualPlayerBalance
- Constructor
+ virtualPlayerFactory()
gets AccountID, AccountName, AccountBalance of real world player when invoked. Used each time players "enter" a new game
BlackJack Player Class implements Action Interface
- ArrayList<Card> blackjackHand
- Constructor
+ displayHand(): void
prints hand of particular player
+ playerMakesBet():
implementation will be more clear when game UML is made
Ensure adequate player balance
+ playerHits():
implementation will be more clear when game UML is made
+ playerFolds():
implementation will be more clear when game UML is made
+ playerLeaves():
implementation will be more clear when game UML is made
Poker Player Class Implements Action Interface
- ArrayList<Card> pokerHand
- Constructor
+ displayHand(): void
prints hand of particular player
+ playerMakesBet():
implementation will be more clear when game UML is made
Ensure adequate player balance -> All in!!
+ playerFolds():
implementation will be more clear when game UML is made
+ playerLeaves():
implementation will be more clear when game UML is made
War Player Class Implements Action Interface
- ArrayList<Card> pokerHand
- Constructor
+ displayHand(): void
prints hand of particular player
+ playerMakesBet():
implementation will be more clear when game UML is made
Ensure adequate player balance -> All in!!
+ playerFolds():
implementation will be more clear when game UML is made
+ playerLeaves():
implementation will be more clear when game UML is made
Extends
Extends
Extends


BlackJack CPU Class extends Blackjack Player
- Constructor
+ displayHand(): void
prints hand of particular player
@Override
+ playerMakesBet():
implementation will be more clear when game UML is made
Ensure adequate player balance
@Override
+ playerHits():
implementation will be more clear when game UML is made
@Override
+ playerFolds():
implementation will be more clear when game UML is made
Poker CPU Class extends Blackjack Player
- Constructor
+ displayHand(): void
prints hand of particular player
@Override
+ playerMakesBet():
implementation will be more clear when game UML is made
Ensure adequate player balance
@Override
+ playerFolds():
implementation will be more clear when game UML is made
War CPU Class extends Blackjack Player
- Constructor
+ displayHand(): void
prints hand of particular player
@Override
+ playerMakesBet():
implementation will be more clear when game UML is made
Ensure adequate player balance
@Override
+ playerFolds():
implementation will be more clear when game UML is made