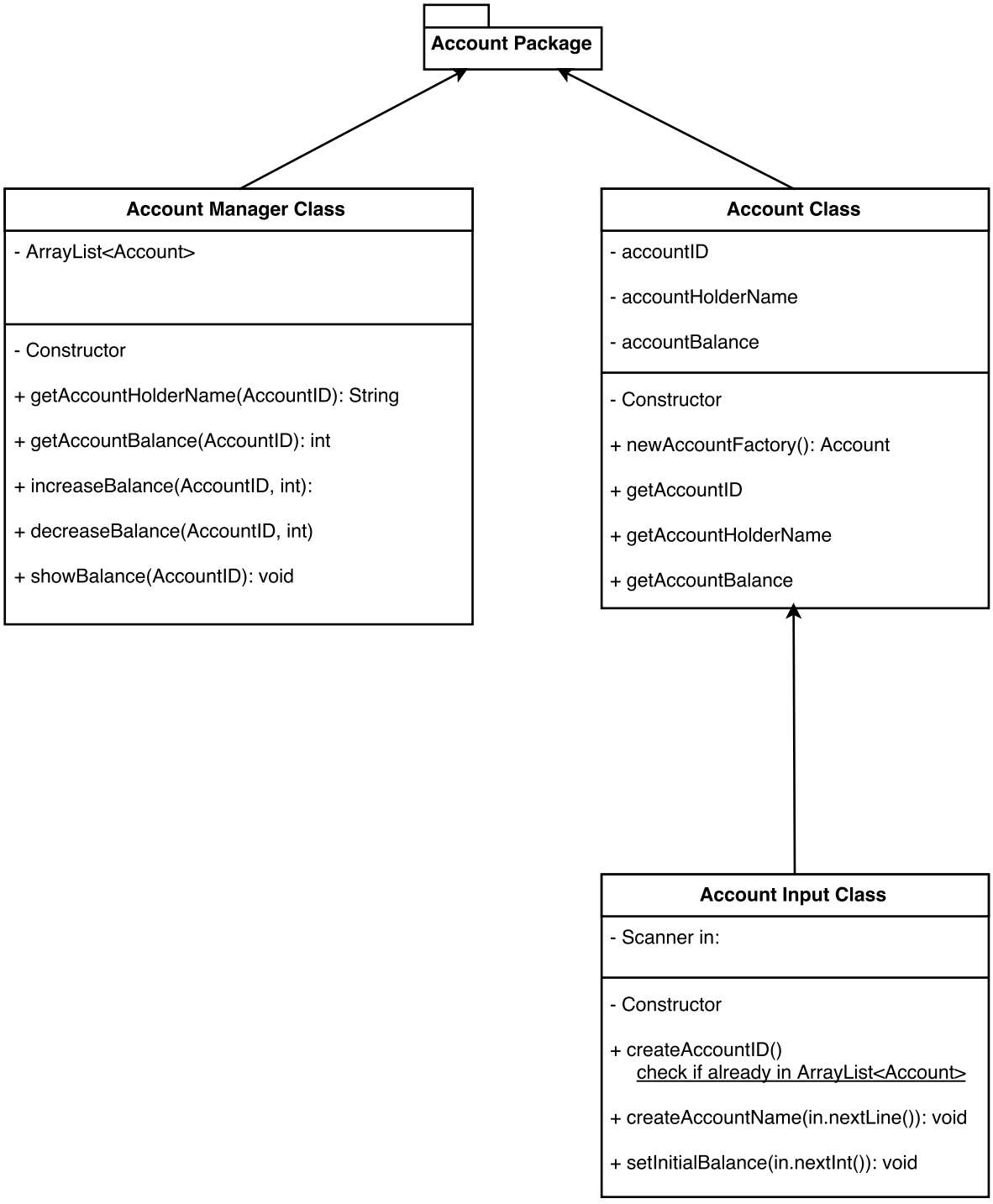
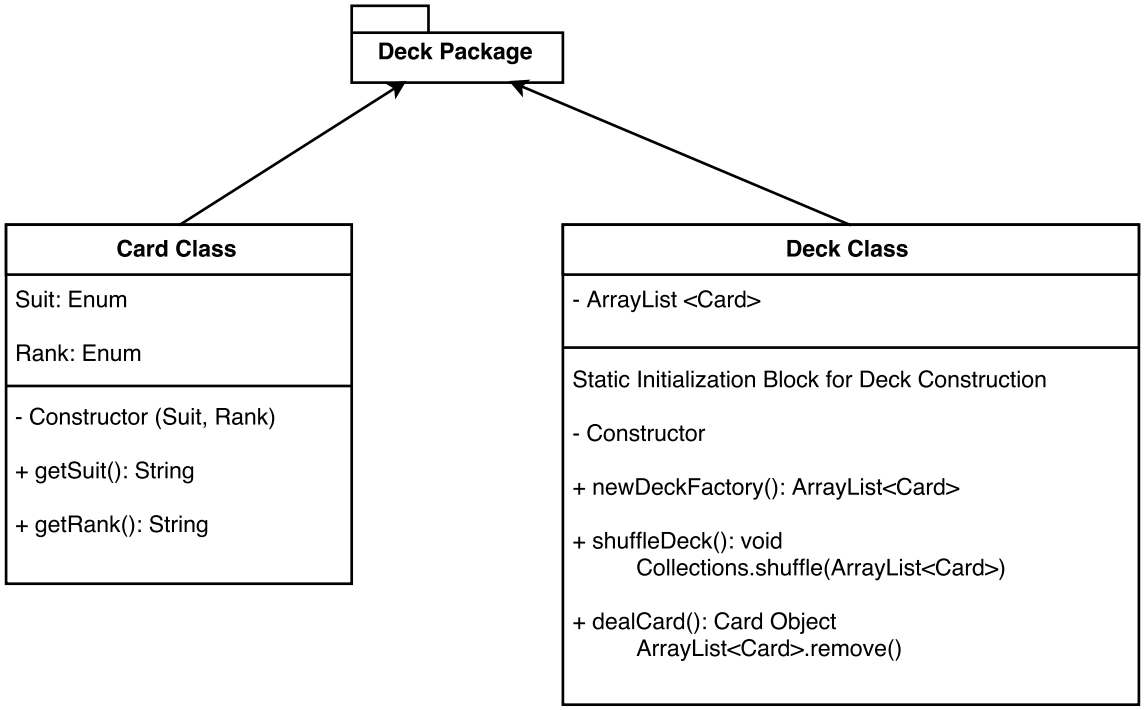
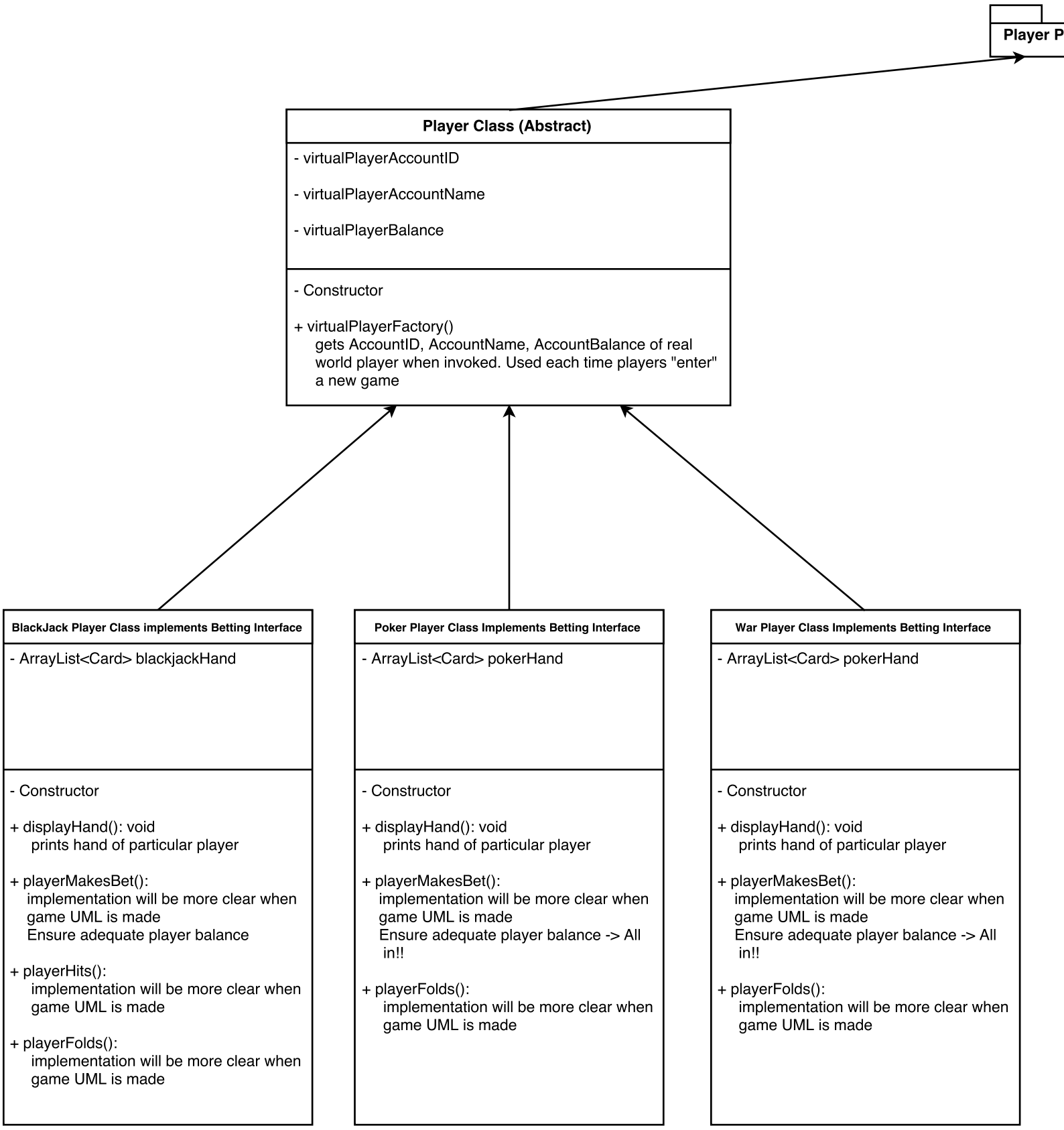
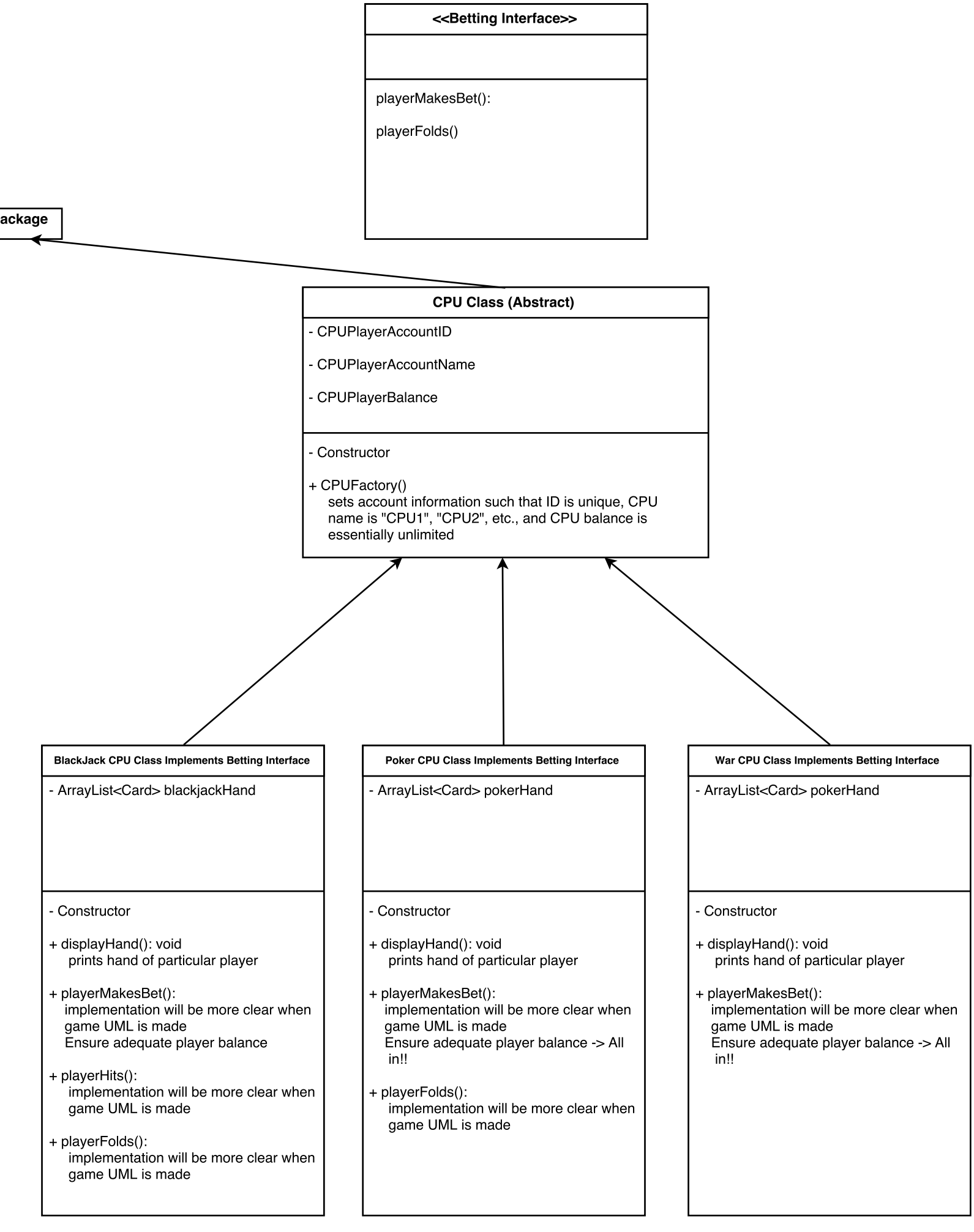

Account Package
Account Manager Class
- ArrayList<Account>
- Constructor
+ getAccountHolderName(AccountID): String
+ getAccountBalance(AccountID): int
+ increaseBalance(AccountID, int):
+ decreaseBalance(AccountID, int)
+ showBalance(AccountID): void
Account Class
- accountID
- accountHolderName
- accountBalance
- 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
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 P
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 Betting 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
Poker Player Class Implements Betting 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
War Player Class Implements Betting 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


<<Betting Interface>>
playerMakesBet():
playerFolds()
ackage
CPU Class (Abstract)
- CPUPlayerAccountID
- CPUPlayerAccountName
- CPUPlayerBalance
- Constructor
+ CPUFactory()
sets account information such that ID is unique, CPU name is "CPU1", "CPU2", etc., and CPU balance is essentially unlimited
BlackJack CPU Class Implements Betting 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
Poker CPU Class Implements Betting 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
War CPU Class Implements Betting 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!!