Black Jack

by Andrea Kaminski, Dajana Berthold, Michael Freiwald, Andreas Mayer,
Matthias Müller-Brockhausen, Daniel Sikeler

Index

- Black Jack
- Strategies
- AI Relevance
- System Design & Architecture
- Implementation
- Live Demo

Black Jack

What is Black Jack?

- gambling card-game against the Croupier (dealer)
- goal:

  get closer to 21 points than the Croupier without exceeding that score
- usually played with six standard 52-card decks (312 cards)

Black Jack

card values

- 2 – 9 : the value shown on the card
- kings, queens and jacks : 10 points
- ace : either 1 or 11 points (freely decidable)

moves

- Hit
- Stand
- Double / Double Down
- Split
- Surrender
- Bust / Break

Strategies

- common rules
- Soft 17
- Card Counting
- Strategy Table

Strategies: common rules

- never over 21
- dealer >= 7 & player < 16 -> HIT
- dealer <= 6 & player <= 11 -> HIT
- dealer >= 7 & player >= 17 -> STAND
- dealer <= 6 & player >= 12

Strategies: Soft 17

- ace + 6 -> STAND
- hand >= 17 -> STAND

Hit Soft: ace + 6 -> HIT

Strategies: Card Counting

- High Low:
- introduced in 1963 by Harvey Dubner
- 5 steps

Strategies: Card Counting

step 1:

assign a point value to each rank

High-Low Point Values

RANK	VALUE
2	+1
3	+1
4	+1
5	+1
6	+1
7	0
8	0
9	0
10	-1
J	-1
Q	-1
K	-1
A	-1

Strategies: Card Counting

step 2:

- start with „Running Count“ of zero
- keep adding or subtracting from the „Running Count“

Example:

3,5,K,7,Q,A,8,5,4,2 **->** 1+1-1+0-1-1+0+1+1+1 = +2

Strategies: Card Counting

step 3:

- divide the „Running Count“ by the number of decks remaining
  -> „True Count“

example:

„Running Count“ = 7
4 decks left **->** 7 /4 = 1.75 ~ 2

Strategies: Card Counting

step 4:

- the greater the „True Count“, the more to bet
- when and how much you bet depends on your own style
- try to make your play look natural ->

increase bets after win
decrease after a loss
stay the same after push

Strategies: Card Counting

step 5 (for some hands):

- play according to the „True Count“ AND a table of „Index Numbers“
- The greater the count, the more inclined you will be to STAND, DOUBLE, SPLIT or SURRENDER

example:

P: 15 , **D**: 10, **True Count**: 4 **->** STAND if True Count >= 4

Hand Player :	Hand Dealer : 2	3	4	5	6	7	8	9	10	A
8	H	H	H	H	H	H	H	H	H	H
9	H	DD	DD	DD	DD	H	H	H	H	H
10	DD	DD	DD	DD	DD	DD	DD	DD	H	H
11	DD	DD	DD	DD	DD	DD	DD	DD	DD	H
12	H	H	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H/R	H
16	S	S	S	S	S	H	H	H/R	H/R	H/R
17	S	S	S	S	S	S	S	S	S	S
A,2	H	H	H	DD	DD	H	H	H	H	H
A,3	H	H	H	DD	DD	H	H	H	H	H
A,4	H	H	DD	DD	DD	H	H	H	H	H
A,5	H	H	DD	DD	DD	H	H	H	H	H
A,6	H	DD	DD	DD	DD	H	H	H	H	H
A,7	S	DD	DD	DD	DD	S	S	H	H	H
A,8	S	S	S	S	S	S	S	S	S	S
A,9	S	S	S	S	S	S	S	S	S	S
2,2	H/P	H/P	P	P	P	P	H	H	H	H
3,3	H/P	H/P	P	P	P	P	H	H	H	H
4,4	H	H	H	H/P	H/P	H	H	H	H	H
5,5	DD	DD	DD	DD	DD	DD	DD	DD	H	H
6,6	H/P	P	P	P	P	H	H	H	H	H
7,7	P	P	P	P	P	P	H	H	H	H
8,8	P	P	P	P	P	P	P	P	P	P
9,9	P	P	P	P	P	S	P	P	S	S
10,10	S	S	S	S	S	S	S	S	S	S
A,A	P	P	P	P	P	P	P	P	P	P

Shortcuts:

S	Stand
H	Hit
P	Split
DD	Double Down
H/P	Split if Player is able to make DD next Round. Else : HIT
H/R	Surrender if possible. Else : HIT

Strategies: Table

- first time 1958
- left column hand of Player
- horizontal column hand of Dealer

AI relevance

- different types of agents
  - Simple Reflex Agent
  - Omniscient Agent
  - Model Based Reflex Agent
  - Goal Based Agent
  - Learning Agent
- predicate logic
  - Java has a class for this with „and“, „or“ and „negate“ methods

Agent‘s Performance

Name	Type	Wins	Loses	Black Jack	Profit
WallHackAgent	Omniscient	5587	3075	397	2262
AlwaysStandAgent	Simple	3872	5618	466	-1513
HitUntilAgent	Goal	3987	5105	468	-884
ReflexAgent	Simple	3115	6811	463	-4619
BasicStrategyAgent	Model	4711	4746	489	-527
HighLowAgent	Model	4091	5328	434	-6740
PredicateAgent	Model	4517	5411	433	-1321
SaveAgent	Goal	4192	5198	458	-777
LearningAgent	Learning	4643	4816	453	-598

Agent‘s Performance (2)

- 10000 games player vs. dealer
- WallHackAgent
  - for comparison reasons
  - best performance
- Black Jack doesn‘t say much about performance.
- Many wins don‘t mean high profit (wager is relevant too).
- Luck is significant for Black Jack and so it‘s difficult to construct good agents.

Software Design & Architecture

Basics

- necessity for a Black Jack Game API
  - implement own game

    + full control of the api

    + complete understanding of all game mechanisms

    - waste time

    - potential fault in game logic
  - find available open source implementation

    + more time to build agents

    + guarantees correct game implementation

    - may need modifications to fit purpose

Black Jack API

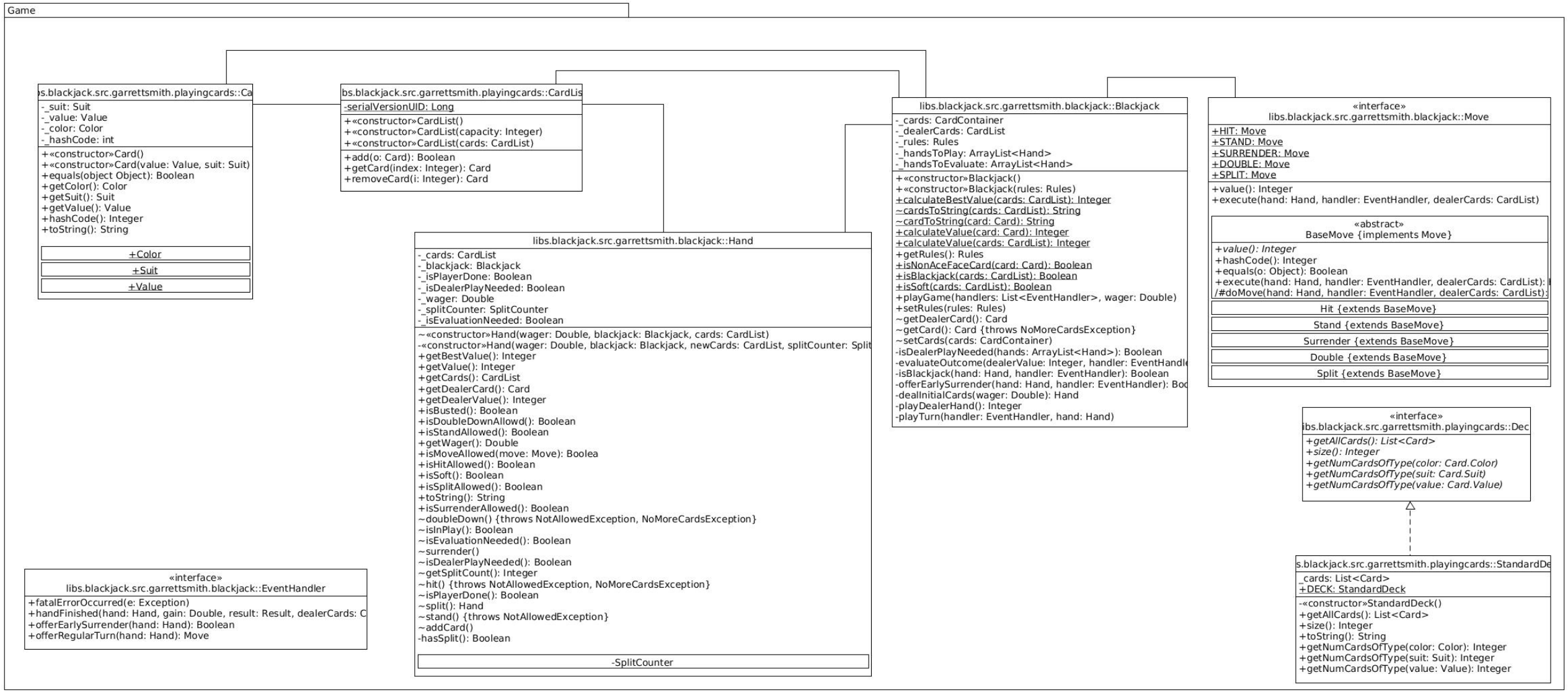
- provide an interface for players
- prevent illegal actions
- take care of game logic
- support for advanced features
  - wagers / purse
  - split & double
  - multiplayer
- Monte Jack (everything except multiplayer)
  http://garrettsmith.net/montejack/

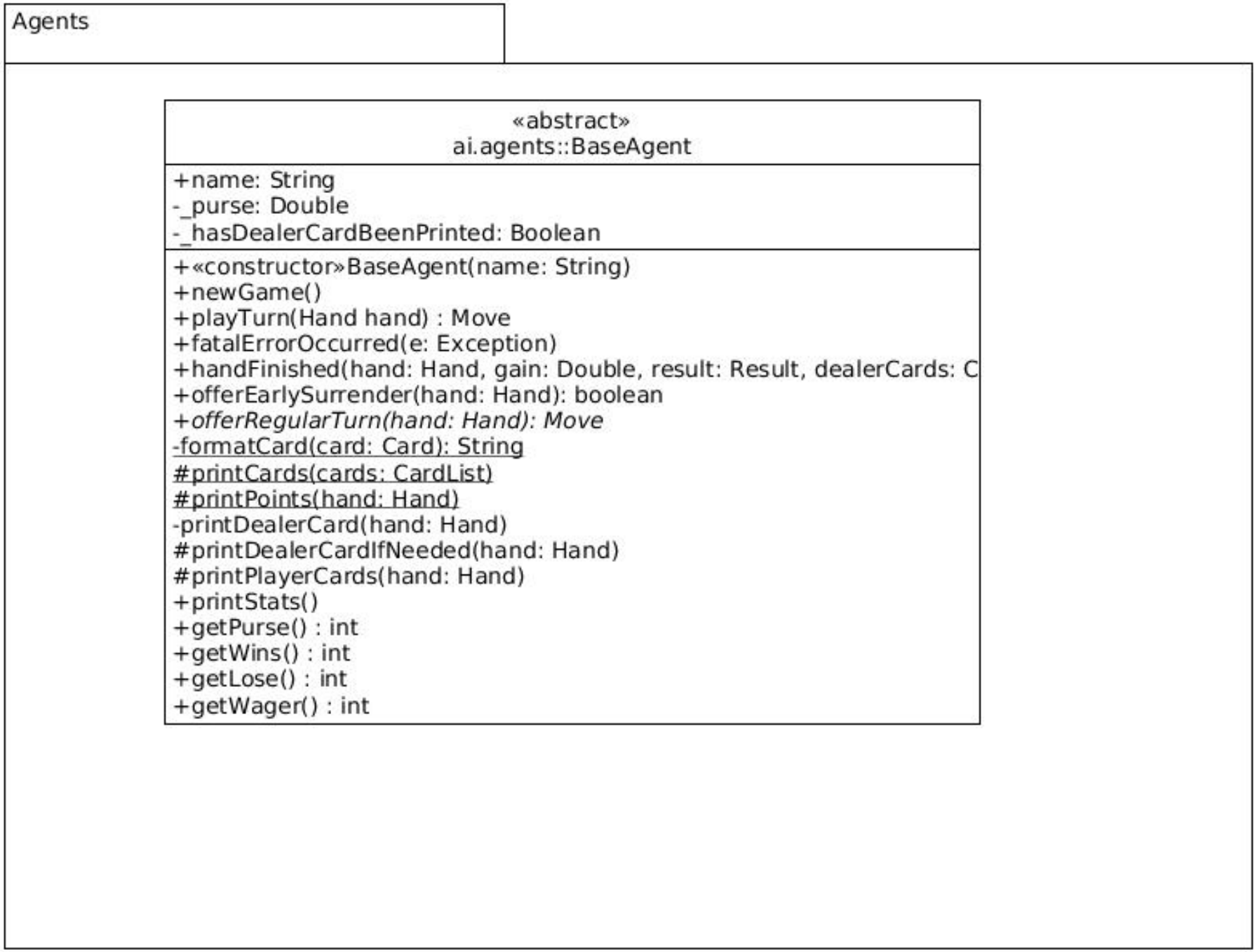
Modifications needed

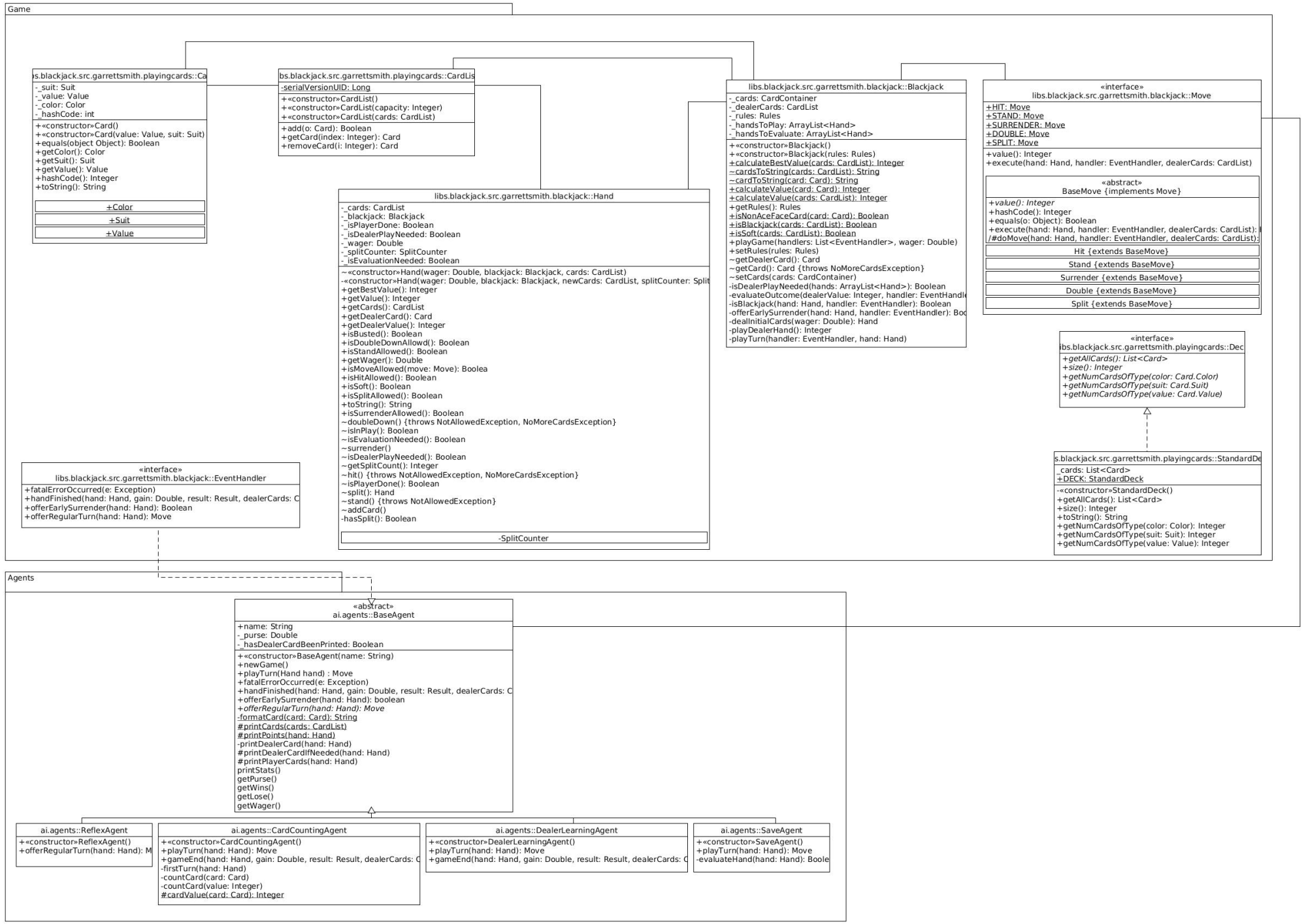
- multiplayer
- added Publish-Subscribe-Pattern
  - for observation of the game environment.
  - Example: new card decks available
- publish some methods for omniscience

BaseAgent

- implements the PlayerHandler of the API
- Base Class for our agents (DRY)
- handles logging
- Agents only have to implement playTurn(Hand).


Game
s.blackjack.src.garrettsmith.playingcards::Ca
-_suit: Suit
-_value: Value
-_color: Color
-_hashCode: int
+«constructor»Card()
+«constructor»Card(value: Value, suit: Suit)
+equals(object Object): Boolean
+getColor(): Color
+getSuit(): Suit
+getValue(): Value
+hashCode(): Integer
+toString(): String
+Color
+Suit
+Value
bs.blackjack.src.garrettsmith.playingcards::CardLis
-serialVersionUID: Long
+«constructor»CardList()
+«constructor»CardList(capacity: Integer)
+«constructor»CardList(cards: CardList)
+add(o: Card): Boolean
+getCard(index: Integer): Card
+removeCard(i: Integer): Card
libs.blackjack.src.garrettsmith.blackjack::Hand
-_cards: CardList
-_blackjack: Blackjack
-_isPlayerDone: Boolean
-_isDealerPlayNeeded: Boolean
-_wager: Double
-_splitCounter: SplitCounter
-_isEvaluationNeeded: Boolean
~«constructor»Hand(wager: Double, blackjack: Blackjack, cards: CardList)
-«constructor»Hand(wager: Double, blackjack: Blackjack, newCards: CardList, splitCounter: Split
+getBestValue(): Integer
+getValue(): Integer
+getCards(): CardList
+getDealerCard(): Card
+getDealerValue(): Integer
+isBusted(): Boolean
+isDoubleDownAllowd(): Boolean
+isStandAllowed(): Boolean
+getWager(): Double
+isMoveAllowed(move: Move): Boolea
+isHitAllowed(): Boolean
+isSoft(): Boolean
+isSplitAllowed(): Boolean
+toString(): String
+isSurrenderAllowed(): Boolean
~doubleDown() {throws NotAllowedException, NoMoreCardsException}
~isInPlay(): Boolean
~isEvaluationNeeded(): Boolean
~surrender()
~isDealerPlayNeeded(): Boolean
~getSplitCount(): Integer
~hit() {throws NotAllowedException, NoMoreCardsException}
~isPlayerDone(): Boolean
~split(): Hand
~stand() {throws NotAllowedException}
~addCard()
-hasSplit(): Boolean
-SplitCounter
libs.blackjack.src.garrettsmith.blackjack::Blackjack
-_cards: CardContainer
-_dealerCards: CardList
-_rules: Rules
-_handsToPlay: ArrayList<Hand>
-_handsToEvaluate: ArrayList<Hand>
+«constructor»Blackjack()
+«constructor»Blackjack(rules: Rules)
+calculateBestValue(cards: CardList): Integer
~cardsToString(cards: CardList): String
~cardToString(card: Card): String
+calculateValue(card: Card): Integer
+calculateValue(cards: CardList): Integer
+getRules(): Rules
+isNonAceFaceCard(card: Card): Boolean
+isBlackjack(cards: CardList): Boolean
+isSoft(cards: CardList): Boolean
+playGame(handlers: List<EventHandler>, wager: Double)
+setRules(rules: Rules)
~getDealerCard(): Card
~getCard(): Card {throws NoMoreCardsException}
~setCards(cards: CardContainer)
-isDealerPlayNeeded(hands: ArrayList<Hand>): Boolean
-evaluateOutcome(dealerValue: Integer, handler: EventHandl
-isBlackjack(hand: Hand, handler: EventHandler): Boolean
-offerEarlySurrender(hand: Hand, handler: EventHandler): Boo
-dealInitialCards(wager: Double): Hand
-playDealerHand(): Integer
-playTurn(handler: EventHandler, hand: Hand)
«interface»
libs.blackjack.src.garrettsmith.blackjack::Move
+HIT: Move
+STAND: Move
+SURRENDER: Move
+DOUBLE: Move
+SPLIT: Move
+value(): Integer
+execute(hand: Hand, handler: EventHandler, dealerCards: CardList)
«abstract»
BaseMove {implements Move}
+value(): Integer
+hashCode(): Integer
+equals(o: Object): Boolean
+execute(hand: Hand, handler: EventHandler, dealerCards: CardList): I
/#doMove(hand: Hand, handler: EventHandler, dealerCards: CardList):
Hit {extends BaseMove}
Stand {extends BaseMove}
Surrender {extends BaseMove}
Double {extends BaseMove}
Split {extends BaseMove}
«interface»
libs.blackjack.src.garrettsmith.blackjack::EventHandler
+fatalErrorOccurred(e: Exception)
+handFinished(hand: Hand, gain: Double, result: Result, dealerCards: C
+offerEarlySurrender(hand: Hand): Boolean
+offerRegularTurn(hand: Hand): Move
«interface»
ibs.blackjack.src.garrettsmith.playingcards::Dec
+getAllCards(): List<Card>
+size(): Integer
+getNumCardsOfType(color: Card.Color)
+getNumCardsOfType(suit: Card.Suit)
+getNumCardsOfType(value: Card.Value)
s.blackjack.src.garrettsmith.playingcards::StandardDe
_cards: List<Card>
+DECK: StandardDeck
-«constructor»StandardDeck()
+getAllCards(): List<Card>
+size(): Integer
+toString(): String
+getNumCardsOfType(color: Color): Integer
+getNumCardsOfType(suit: Suit): Integer
+getNumCardsOfType(value: Value): Integer


Agents
«abstract»
ai.agents::BaseAgent
+name: String
-_purse: Double
-_hasDealerCardBeenPrinted: Boolean
+«constructor»BaseAgent(name: String)
+newGame()
+playTurn(Hand hand) : Move
+fatalErrorOccurred(e: Exception)
+handFinished(hand: Hand, gain: Double, result: Result, dealerCards: C
+offerEarlySurrender(hand: Hand): boolean
+offerRegularTurn(hand: Hand): Move
-formatCard(card: Card): String
#printCards(cards: CardList)
#printPoints(hand: Hand)
-printDealerCard(hand: Hand)
#printDealerCardIfNeeded(hand: Hand)
#printPlayerCards(hand: Hand)
+printStats()
+getPurse() : int
+getWins() : int
+getLose() : int
+getWager() : int


Game
s.blackjack.src.garrettsmith.playingcards::Ca
-_suit: Suit
-_value: Value
-_color: Color
-_hashCode: int
+«constructor»Card()
+«constructor»Card(value: Value, suit: Suit)
+equals(object Object): Boolean
+getColor(): Color
+getSuit(): Suit
+getValue(): Value
+hashCode(): Integer
+toString(): String
+Color
+Suit
+Value
bs.blackjack.src.garrettsmith.playingcards::CardLis
-serialVersionUID: Long
+«constructor»CardList()
+«constructor»CardList(capacity: Integer)
+«constructor»CardList(cards: CardList)
+add(o: Card): Boolean
+getCard(index: Integer): Card
+removeCard(i: Integer): Card
libs.blackjack.src.garrettsmith.blackjack::Blackjack
-_cards: CardContainer
-_dealerCards: CardList
-_rules: Rules
-_handsToPlay: ArrayList<Hand>
-_handsToEvaluate: ArrayList<Hand>
+«constructor»Blackjack()
+«constructor»Blackjack(rules: Rules)
+calculateBestValue(cards: CardList): Integer
~cardsToString(cards: CardList): String
~cardToString(card: Card): String
+calculateValue(card: Card): Integer
+calculateValue(cards: CardList): Integer
+getRules(): Rules
+isNonAceFaceCard(card: Card): Boolean
+isBlackjack(cards: CardList): Boolean
+isSoft(cards: CardList): Boolean
+playGame(handlers: List<EventHandler>, wager: Double)
+setRules(rules: Rules)
~getDealerCard(): Card
~getCard(): Card {throws NoMoreCardsException}
~setCards(cards: CardContainer)
-isDealerPlayNeeded(hands: ArrayList<Hand>): Boolean
-evaluateOutcome(dealerValue: Integer, handler: EventHandl
-isBlackjack(hand: Hand, handler: EventHandler): Boolean
-offerEarlySurrender(hand: Hand, handler: EventHandler): Boo
-dealInitialCards(wager: Double): Hand
-playDealerHand(): Integer
-playTurn(handler: EventHandler, hand: Hand)
«interface»
libs.blackjack.src.garrettsmith.blackjack::Move
+HIT: Move
+STAND: Move
+SURRENDER: Move
+DOUBLE: Move
+SPLIT: Move
+value(): Integer
+execute(hand: Hand, handler: EventHandler, dealerCards: CardList)
«abstract»
BaseMove {implements Move}
+value(): Integer
+hashCode(): Integer
+equals(o: Object): Boolean
+execute(hand: Hand, handler: EventHandler, dealerCards: CardList):
/#doMove(hand: Hand, handler: EventHandler, dealerCards: CardList):
Hit {extends BaseMove}
Stand {extends BaseMove}
Surrender {extends BaseMove}
Double {extends BaseMove}
Split {extends BaseMove}
libs.blackjack.src.garrettsmith.blackjack::Hand
-_cards: CardList
-_blackjack: Blackjack
-_isPlayerDone: Boolean
-_isDealerPlayNeeded: Boolean
-_wager: Double
-_splitCounter: SplitCounter
-_isEvaluationNeeded: Boolean
~«constructor»Hand(wager: Double, blackjack: Blackjack, cards: CardList)
-«constructor»Hand(wager: Double, blackjack: Blackjack, newCards: CardList, splitCounter: Split
+getBestValue(): Integer
+getValue(): Integer
+getCards(): CardList
+getDealerCard(): Card
+getDealerValue(): Integer
+isBusted(): Boolean
+isDoubleDownAllowd(): Boolean
+isStandAllowed(): Boolean
+getWager(): Double
+isMoveAllowed(move: Move): Boolea
+isHitAllowed(): Boolean
+isSoft(): Boolean
+isSplitAllowed(): Boolean
+toString(): String
+isSurrenderAllowed(): Boolean
~doubleDown() {throws NotAllowedException, NoMoreCardsException}
~isInPlay(): Boolean
~isEvaluationNeeded(): Boolean
~surrender()
~isDealerPlayNeeded(): Boolean
~getSplitCount(): Integer
~hit() {throws NotAllowedException, NoMoreCardsException}
~isPlayerDone(): Boolean
~split(): Hand
~stand() {throws NotAllowedException}
~addCard()
-hasSplit(): Boolean
-SplitCounter
«interface»
libs.blackjack.src.garrettsmith.blackjack::EventHandler
+fatalErrorOccurred(e: Exception)
+handFinished(hand: Hand, gain: Double, result: Result, dealerCards: C
+offerEarlySurrender(hand: Hand): Boolean
+offerRegularTurn(hand: Hand): Move
«interface»
ibs.blackjack.src.garrettsmith.playingcards::Dec
+getAllCards(): List<Card>
+size(): Integer
+getNumCardsOfType(color: Card.Color)
+getNumCardsOfType(suit: Card.Suit)
+getNumCardsOfType(value: Card.Value)
s.blackjack.src.garrettsmith.playingcards::StandardDe
_cards: List<Card>
+DECK: StandardDeck
-«constructor»StandardDeck()
+getAllCards(): List<Card>
+size(): Integer
+toString(): String
+getNumCardsOfType(color: Color): Integer
+getNumCardsOfType(suit: Suit): Integer
+getNumCardsOfType(value: Value): Integer
Agents
«abstract»
ai.agents::BaseAgent
+name: String
-_purse: Double
-_hasDealerCardBeenPrinted: Boolean
+«constructor»BaseAgent(name: String)
+newGame()
+playTurn(Hand hand) : Move
+fatalErrorOccurred(e: Exception)
+handFinished(hand: Hand, gain: Double, result: Result, dealerCards: C
+offerEarlySurrender(hand: Hand): boolean
+offerRegularTurn(hand: Hand): Move
-formatCard(card: Card): String
#printCards(cards: CardList)
#printPoints(hand: Hand)
-printDealerCard(hand: Hand)
#printDealerCardIfNeeded(hand: Hand)
#printPlayerCards(hand: Hand)
printStats()
getPurse()
getWins()
getLose()
getWager()
ai.agents::ReflexAgent
+«constructor»ReflexAgent()
+offerRegularTurn(hand: Hand): M
ai.agents::CardCountingAgent
+«constructor»CardCountingAgent()
+playTurn(hand: Hand): Move
+gameEnd(hand: Hand, gain: Double, result: Result, dealerCards: C
-firstTurn(hand: Hand)
-countCard(card: Card)
-countCard(value: Integer)
#cardValue(card: Card): Integer
ai.agents::DealerLearningAgent
+«constructor»DealerLearningAgent()
+playTurn(hand: Hand): Move
+gameEnd(hand: Hand, gain: Double, result: Result, dealerCards: C
ai.agents::SaveAgent
+«constructor»SaveAgent()
+playTurn(hand: Hand): Move
-evaluateHand(hand: Hand): Boole

Agent Implementation

Simple

- AlwaysStandAgent
- ReflexAgent

Goal

- HitUntilAgent
- SaveAgent

Model

- BasicStrategyAgent
- HighLowAgent
- PredicateAgent

Learning

- LearningAgent

Omniscient

- WallHackAgent

Methods to implement

- All agents extend the abstract class *BaseAgent.*

- Our Agents need:
  - Default-Constructor
  - Method: playTurn()
  - optional: override dealerCreateNewDecks()

Simple – ReflexAgent

- 4 possible moves:
  - double down
  - stand
  - split
  - hit

- use java.util.Random to get a random integer value between 0-3
- Do this as long as one of the moves is allowed.

GoalBased – HitUntilAgent / SaveAgent

Both of the agents check whether they are above a certain value or not.

HitUntilAgent

- hits until his hand card value is above 17
- problem:
  ace has card value 1 -> the agent may hit too often

SaveAgent

- will only hit if it isn‘t possible to get a value greater than 21
- same logic as HitUntilAgent without its problem

ModelBased - BasicStrategy

- performes a move based on its and the dealer‘s hand
- separation in three different agents:
  - BasicStrategySplitAgent
  - BasicStrategySoftAgent
  - BasicStrategyHardAgent
- problems:
  No information which moves are allowed in the next round
  H/P is always a hit in our case

Hand Player : \ Hand Dealer :	2	3	4	5	6	7	8	9	10	A
8	H	H	H	H	H	H	H	H	H	H
9	H	DD	DD	DD	DD	H	H	H	H	H
10	DD	DD	DD	DD	DD	DD	DD	DD	H	H
11	DD	DD	DD	DD	DD	DD	DD	DD	DD	H
12	H	H	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H/R	H
16	S	S	S	S	S	H	H	H/R	H/R	H/R
17	S	S	S	S	S	S	S	S	S	S
A,2	H	H	H	DD	DD	H	H	H	H	H
A,3	H	H	H	DD	DD	H	H	H	H	H
A,4	H	H	DD	DD	DD	H	H	H	H	H
A,5	H	H	DD	DD	DD	H	H	H	H	H
A,6	H	DD	DD	DD	DD	H	H	H	H	H
A,7	S	DD	DD	DD	DD	S	S	H	H	H
A,8	S	S	S	S	S	S	S	S	S	S
A,9	S	S	S	S	S	S	S	S	S	S
2,2	H/P	H/P	P	P	P	P	H	H	H	H
3,3	H/P	H/P	P	P	P	P	H	H	H	H
4,4	H	H	H	H/P	H/P	H	H	H	H	H
5,5	DD	DD	DD	DD	DD	DD	DD	DD	H	H
6,6	H/P	P	P	P	P	H	H	H	H	H
7,7	P	P	P	P	P	P	H	H	H	H
8,8	P	P	P	P	P	P	P	P	P	P
9,9	P	P	P	P	P	S	P	P	S	S
10,10	S	S	S	S	S	S	S	S	S	S
A,A	P	P	P	P	P	P	P	P	P	P

Learning – LearningAgent

- 2 maps: one for loosing / one for winning history

- check whether the player cards were used in a previous game:
  - Yes:
    Check if win or loose rate was higher:
    -> perform the same action as last time
  - No:
    use the BasicStrategyAgent to do a move
- save the last result

Live Demo


€1 MIN
€40 MAX
Black Jack
Professional Series
BLACK JACK PAYS 3 TO 2
Dealer must stand on 17 and must draw to 16
INSURANCE PAYS 2 TO 1
Deal
POWERED BY NET ENTERTAINMENT™
Cash: € 5000.00 Bet: € 0.00 Win: € 0.00

Thank you for your attention and have a nice Black Jack.