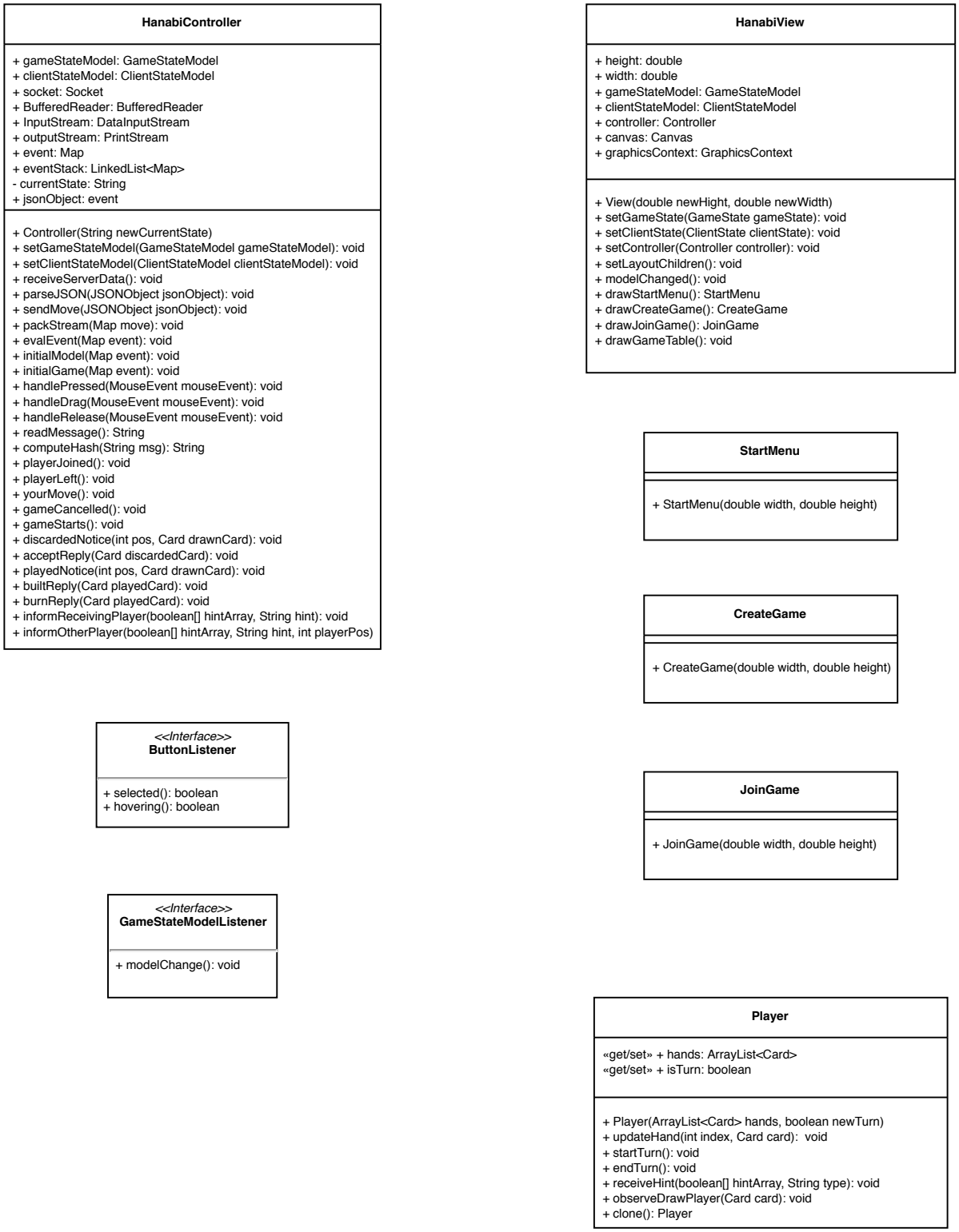
HanabiController
+ gameStateModel: GameStateModel
+ clientStateModel: ClientStateModel
+ socket: Socket
+ BufferedReader: BufferedReader
+ InputStream: DataInputStream
+ outputStream: PrintStream
+ event: Map
+ eventStack: LinkedList<Map>
- currentState: String
+ jsonObject: event
+ Controller(String newCurrentState)
+ setGameStateModel(GameStateModel gameStateModel): void
+ setClientStateModel(ClientStateModel clientStateModel): void
+ receiveServerData(): void
+ parseJSON(JSONObject jsonObject): void
+ sendMove(JSONObject jsonObject): void
+ packStream(Map move): void
+ evalEvent(Map event): void
+ initialModel(Map event): void
+ initialGame(Map event): void
+ handlePressed(MouseEvent mouseEvent): void
+ handleDrag(MouseEvent mouseEvent): void
+ handleRelease(MouseEvent mouseEvent): void
+ readMessage(): String
+ computeHash(String msg): String
+ playerJoined(): void
+ playerLeft(): void
+ yourMove(): void
+ gameCancelled(): void
+ gameStarts(): void
+ discardedNotice(int pos, Card drawnCard): void
+ acceptReply(Card discardedCard): void
+ playedNotice(int pos, Card drawnCard): void
+ builtReply(Card playedCard): void
+ burnReply(Card playedCard): void
+ informReceivingPlayer(boolean[] hintArray, String hint): void
+ informOtherPlayer(boolean[] hintArray, String hint, int playerPos)
HanabiView
+ height: double
+ width: double
+ gameStateModel: GameStateModel
+ clientStateModel: ClientStateModel
+ controller: Controller
+ canvas: Canvas
+ graphicsContext: GraphicsContext
+ View(double newHight, double newWidth)
+ setGameState(GameState gameState): void
+ setClientState(ClientState clientState): void
+ setController(Controller controller): void
+ setLayoutChildren(): void
+ modelChanged(): void
+ drawStartMenu(): StartMenu
+ drawCreateGame(): CreateGame
+ drawJoinGame(): JoinGame
+ drawGameTable(): void
StartMenu
+ StartMenu(double width, double height)
CreateGame
+ CreateGame(double width, double height)
JoinGame
+ JoinGame(double width, double height)
<<Interface>>
ButtonListener
+ selected(): boolean
+ hovering(): boolean
<<Interface>>
GameStateModelListener
+ modelChange(): void
Player
«get/set» + hands: ArrayList<Card>
«get/set» + isTurn: boolean
+ Player(ArrayList<Card> hands, boolean newTurn)
+ updateHand(int index, Card card): void
+ startTurn(): void
+ endTurn(): void
+ receiveHint(boolean[] hintArray, String type): void
+ observeDrawPlayer(Card card): void
+ clone(): Player

GameStateModel

«get/set» - hintsRemain: int
«get/set» - fusesRemain: int
«get/set» - cardsRemain: int
«get/set» - timeRemain: int
«get/set» - score: int
«get/set» - cardsOnTable: ArrayList<Card>
«get/set» - discardPile: LinkedList<Card>
«get/set» - players: ArrayList<Player>
«get/set» - turn: int
«get/set» - subscribers: ArrayList<GameStateListener>
«get/set» - mode: String
«get/set» - timePerTurn: Integer
«get/set» - cardsInHand: Integer
«get/set» - playersInGame: Integer
«get/set» - position: Integer
«get/set» + topCard: CPS
«get/set» - myValidityMatrix: ValidityMatrix
«get/set» - myDisardGradient: GradientMatrix
«get/set» - myPlayGradient: GradientMatrix
«get/set» - discardVector: double[]
«get/set» - playVector: double[]
«get/set» + allCards: ArrayList<Card>
«get/set» + selectedCard: Card
«get/set» + isDragging: boolean
«get/set» + playBox: PlayDiscardBox
«get/set» + discardBox: PlayDiscardBox
«get/set» + turnBox: PlayDiscardBox
«get/set» + hint: boolean
«get/set» + hintX: double
«get/set» + hintY: double

+ GameStateModel(Array<Player> newPlayers)
+ appendDiscard (Card card): void
+ updateCardsOnTable(Card card): void
+ updatePlayerHand(Player player, Card card): void
+ incrementHints(): void
+ decrementHints(): void
+ decrementFuses(): void
+ decrementCards(): void
+ decrementTime(): void
+ addSubscriber(GameStateModelListener Sub): void
+ notifySubscribers(): void
+ appendDiscard(Card Card): void
+ updateCardsOnTable(Card card): void
+ updatePlayerHand(Player player, Card card, Integer pos): void
+ updatePlayerHand(Player player, String myColor, Integer myNumber, Integer pos): void
+ clone(): GameState
+ handContain(Card[] hand, Card card): boolean
+ setTurnBox(double canvasWidth, double canvasHeight): void
+ setPlayDiscardBox(double canvasWidth, double canvasHeight): void
+ setCoordinates(double canvasWidth, double canvasHeight): void
+ addToAllCard(): void
+ checkHit(double clickX, double clickY): void
+ witchCard(double x, double y): Card
+ moveCard(Card card, double dX, double dY): void
+ setCardXY(Card card, double newX, double newY): void
+ addSubscriber(GameStateListener sub): void
+ notifySubscribers(): void

ClientStateModel

«get/set» - playerSelected: int
«get/set» - cardsSelected: ArrayList<int>
«get/set» - menusOpen: Map<String, boolean>
«get/set» - actionSelected: String
«get/set» - notes: ArrayList<int>
«get/set» - buttons: ArrayList<Button>
«get/set» - players: ArrayList<Player>

+ ClientStateModel(Player newPlayerSelected, Map<String, boolean> newMenusOpen, String newActionSelected, ArrayList<int> notes, ArrayList<Button> newButtons)
+ toggleMenu(String name): void
+ selectAction(String action): void
+ getSelectedPlayer(): void
+ getSelectedCards(): void
+ getAction(): void
- seletePlayer(Integer index): void
- deseletePlayer(): void
- addCardSelected(Integer index): void
- deselectedCard(Integer cardIndex): void
- clearCardsSelected(): void
+ selectCard(Integer playerIndex, Integer cardIndex): void
+ clone(): Card

Card

«get/set» + color: String
«get/set» + number: String
«get/set» + clickable: boolean
«get/set» + selected: boolean
«get/set» + hovering: boolean
«get/set» + x: double
«get/set» + y: double
«get/set» + width: double
«get/set» + height: double
«get/set» - myCPS: CPS
«get/set» - myGameTyple: String
«get/set» - myDiscardEv: double
«get/set» - myPlayEv: double

+ Card(String newColor, String newNumber)
+ cardToIndex(): int[]
+ getString(): String
+ checkHit(double clickX, double clickY): Boolean
+ moveCard(double dx, double dy): void
+ getCardAsString(): String
+ receiveHint(boolean isPositive, String hint): void
+ calculateMyEVs(GradientMatrix discardGradient, GradientMatrix playGradient): void