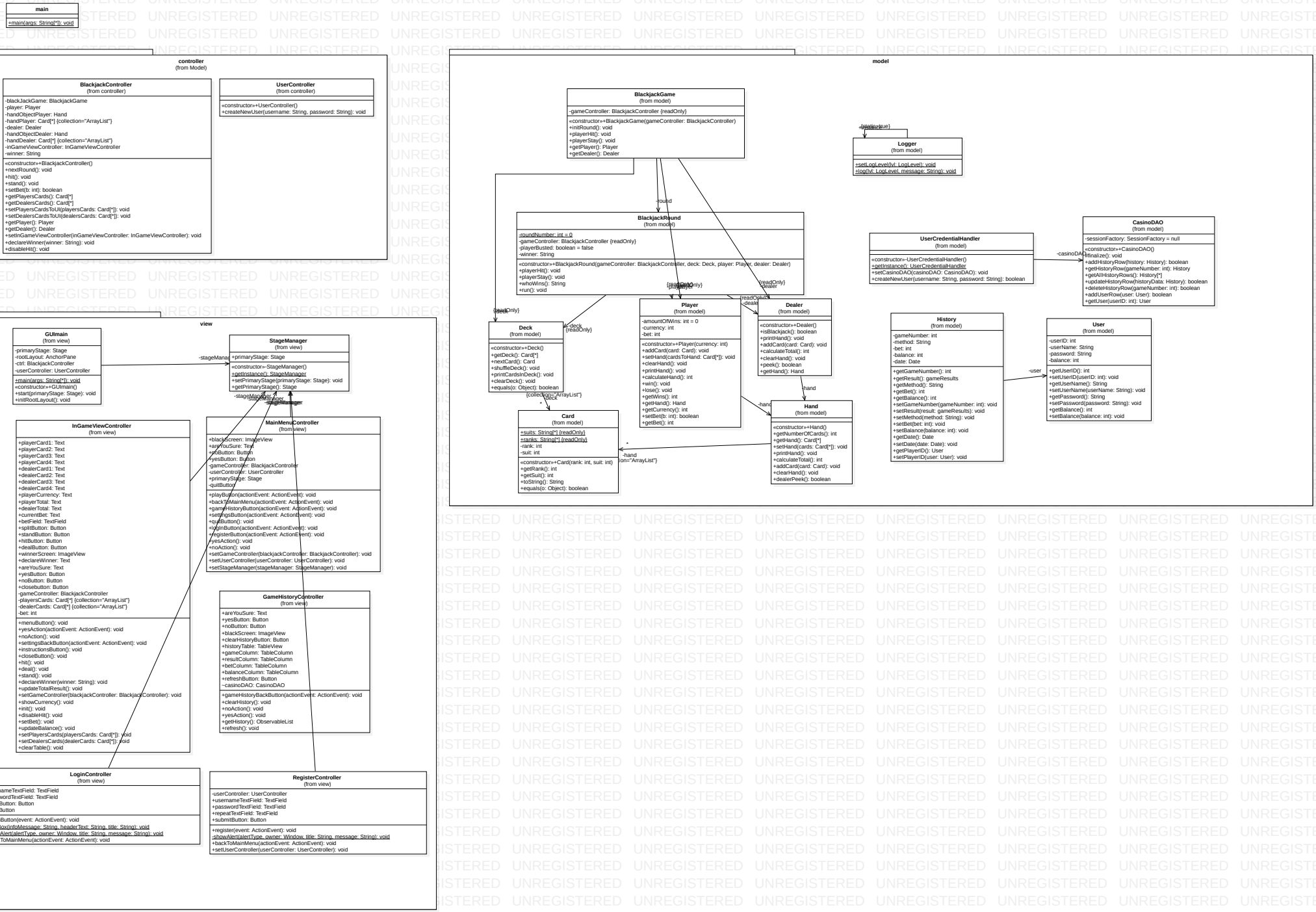
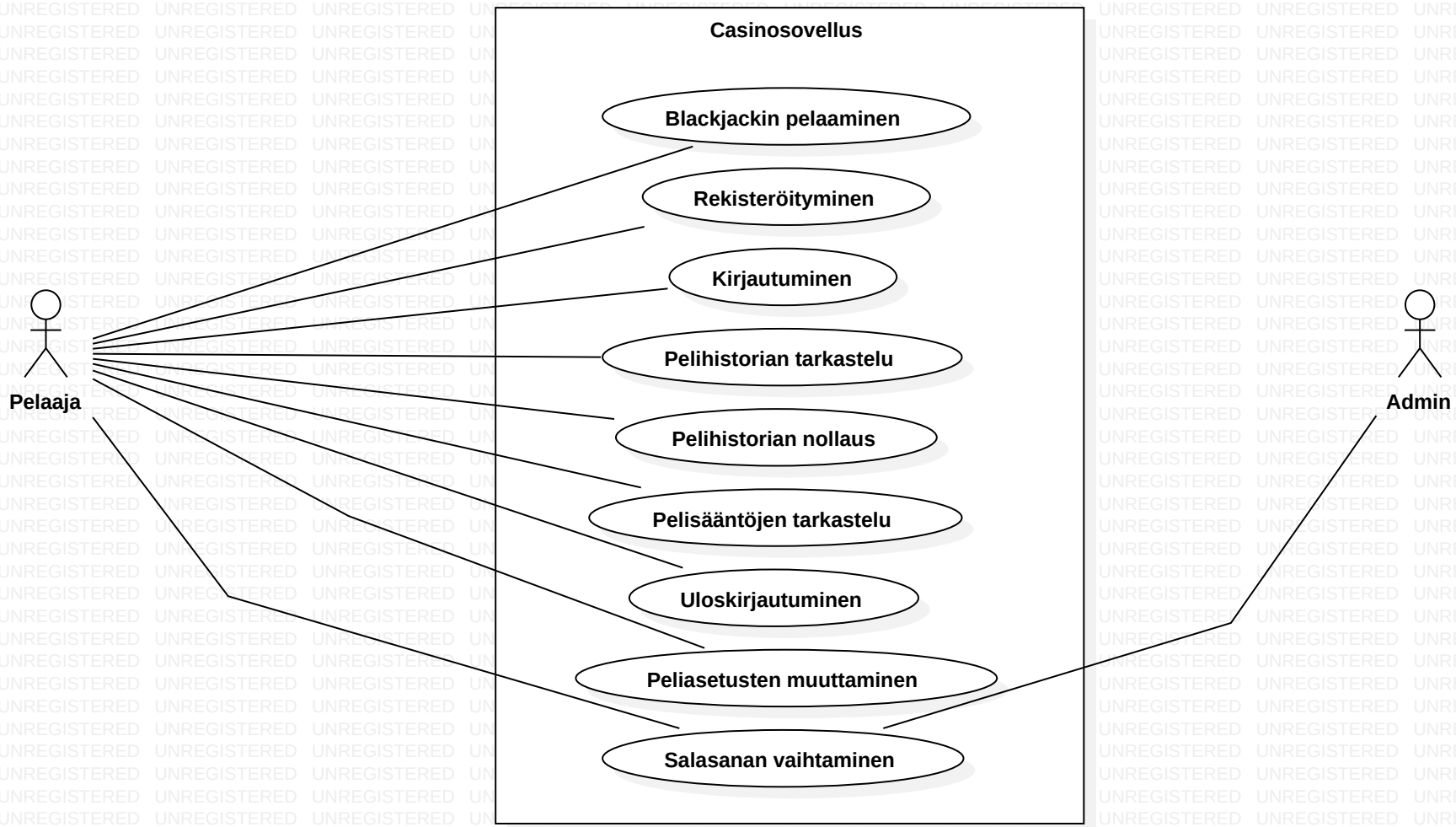
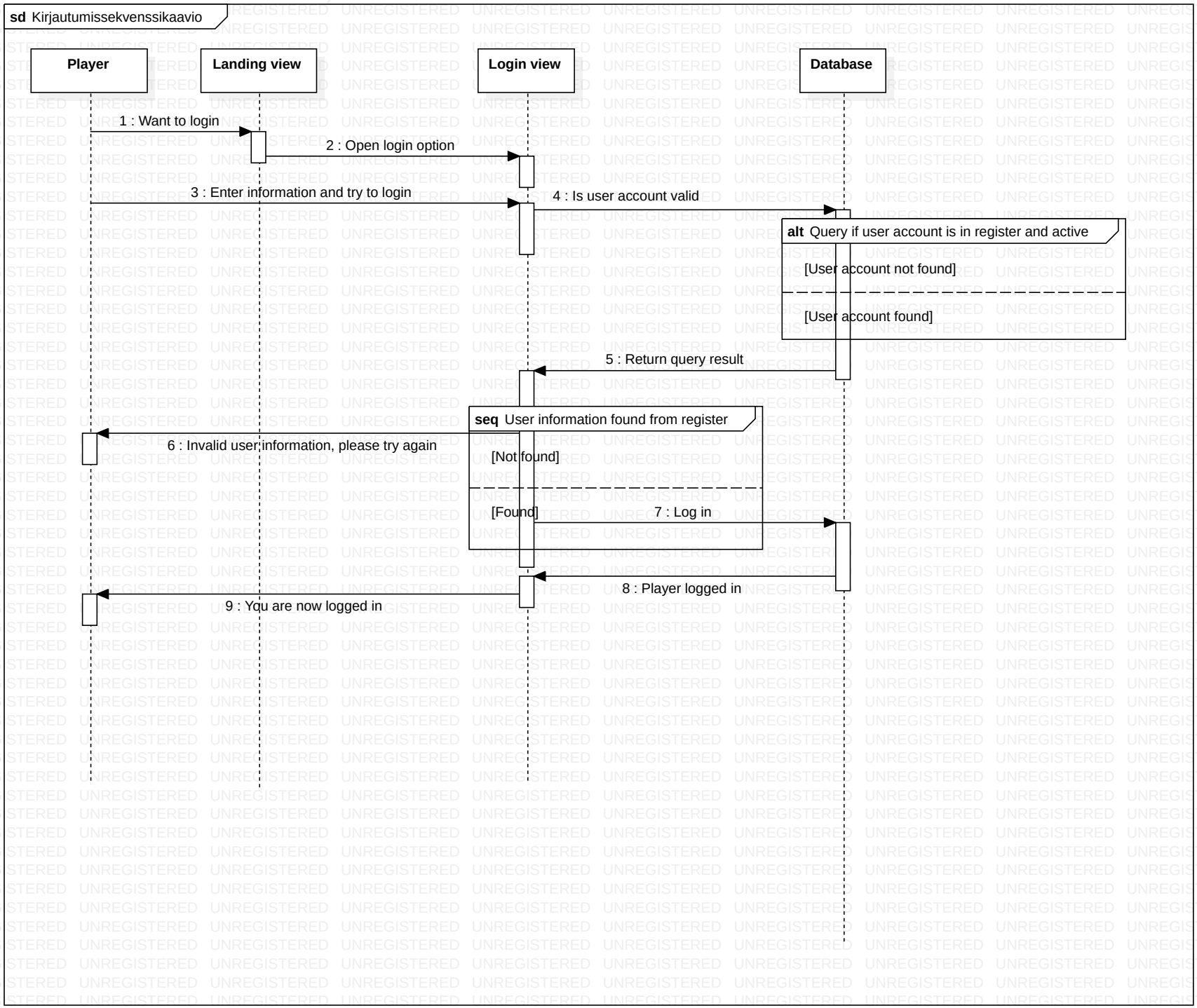
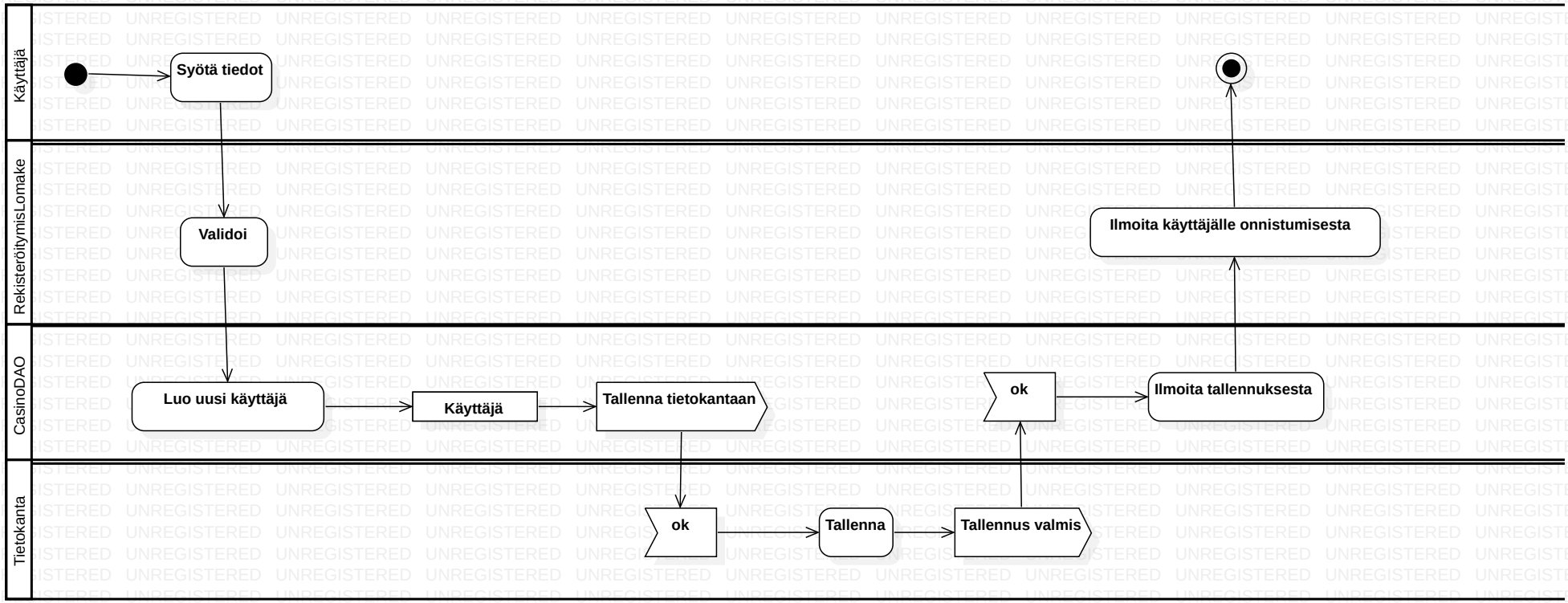

main
+main(args: String[*]): void
controller
(from Model)
BlackjackController
(from controller)
-blackJackGame: BlackjackGame
-player: Player
-handObjectPlayer: Hand
-handPlayer: Card[*] {collection="ArrayList"}
-dealer: Dealer
-handObjectDealer: Hand
-handDealer: Card[*] {collection="ArrayList"}
-inGameViewController: InGameViewController
-winner: String
«constructor»+BlackjackController()
+nextRound(): void
+hit(): void
+stand(): void
+setBet(b: int): boolean
+getPlayersCards(): Card[*]
+getDealersCards(): Card[*]
+setPlayersCardsToUI(playersCards: Card[*]): void
+setDealersCardsToUI(dealersCards: Card[*]): void
+getPlayer(): Player
+getDealer(): Dealer
+setInGameViewControlller(inGameViewController: InGameViewController): void
+declareWinner(winner: String): void
+disableHit(): void
UserController
(from controller)
«constructor»+UserController()
+createNewUser(username: String, password: String): void
model
BlackjackGame
(from model)
-gameController: BlackjackController {readOnly}
«constructor»+BlackjackGame(gameController: BlackjackController)
+initRound(): void
+playerHit(): void
+playerStay(): void
+getPlayer(): Player
+getDealer(): Dealer
Logger
(from model)
+setLogLevel(lvl: LogLevel): void
+log(lvl: LogLevel, message: String): void
BlackjackRound
(from model)
-roundNumber: int = 0
-gameController: BlackjackController {readOnly}
-playerBusted: boolean = false
-winner: String
«constructor»+BlackjackRound(gameController: BlackjackController, deck: Deck, player: Player, dealer: Dealer)
+playerHit(): void
+playerStay(): void
+whoWins(): String
+run(): void
UserCredentialHandler
(from model)
«constructor»-UserCredentialHandler()
+getInstance(): UserCredentialHandler
+setCasinoDAO(casinoDAO: CasinoDAO): void
+createNewUser(username: String, password: String): boolean
CasinoDAO
(from model)
-sessionFactory: SessionFactory = null
«constructor»+CasinoDAO()
+finalize(): void
+addHistoryRow(history: History): boolean
+getHistoryRow(gameNumber: int): History
+getAllHistoryRows(): History[*]
+updateHistoryRow(historyData: History): boolean
+deleteHistoryRow(gameNumber: int): boolean
+addUserRow(user: User): boolean
+getUser(userID: int): User
Deck
(from model)
«constructor»+Deck()
+getDeck(): Card[*]
+nextCard(): Card
+shuffleDeck(): void
+printCardsInDeck(): void
+clearDeck(): void
+equals(o: Object): boolean
Player
(from model)
-amountOfWins: int = 0
-currency: int
-bet: int
«constructor»+Player(currency: int)
+addCard(card: Card): void
+setHand(cardsToHand: Card[*]): void
+clearHand(): void
+printHand(): void
+calculateHand(): int
+win(): void
+lose(): void
+getWins(): int
+getHand(): Hand
+getCurrency(): int
+setBet(b: int): boolean
+getBet(): int
Dealer
(from model)
«constructor»+Dealer()
+isBlackjack(): boolean
+printHand(): void
+addCard(card: Card): void
+calculateTotal(): int
+clearHand(): void
+peek(): boolean
+getHand(): Hand
History
(from model)
-gameNumber: int
-method: String
-bet: int
-balance: int
-date: Date
+getGameNumber(): int
+getResult(): gameResults
+getMethod(): String
+getBet(): int
+getBalance(): int
+setGameNumber(gameNumber: int): void
+setResult(result: gameResults): void
+setMethod(method: String): void
+setBet(bet: int): void
+setBalance(balance: int): void
+getDate(): Date
+setDate(date: Date): void
+getPlayerID(): User
+setPlayerID(user: User): void
User
(from model)
-userID: int
-userName: String
-password: String
-balance: int
+getUserID(): int
+setUserID(userID: int): void
+getUserName(): String
+setUserName(userName: String): void
+getPassword(): String
+setPassword(password: String): void
+getBalance(): int
+setBalance(balance: int): void
Card
(from model)
+suits: String[*] {readOnly}
+ranks: String[*] {readOnly}
-rank: int
-suit: int
«constructor»+Card(rank: int, suit: int)
+getRank(): int
+getSuit(): int
+toString(): String
+equals(o: Object): boolean
Hand
(from model)
«constructor»+Hand()
+getNumberOfCards(): int
+getHand(): Card[*]
+setHand(cards: Card[*]): void
+printHand(): void
+calculateTotal(): int
+addCard(card: Card): void
+clearHand(): void
+dealerPeek(): boolean
view
GUImain
(from view)
-primaryStage: Stage
-rootLayout: AnchorPane
-ctrl: BlackjackController
-userController: UserController
+main(args: String[*]): void
«constructor»+GUImain()
+start(primaryStage: Stage): void
+initRootLayout(): void
StageManager
(from view)
+primaryStage: Stage
«constructor»-StageManager()
+getInstance(): StageManager
+setPrimaryStage(primaryStage: Stage): void
+getPrimaryStage(): Stage
InGameViewController
(from view)
+playerCard1: Text
+playerCard2: Text
+playerCard3: Text
+playerCard4: Text
+dealerCard1: Text
+dealerCard2: Text
+dealerCard3: Text
+dealerCard4: Text
+playerCurrency: Text
+playerTotal: Text
+dealerTotal: Text
+currentBet: Text
+betField: TextField
+splitButton: Button
+standButton: Button
+hitButton: Button
+dealButton: Button
+winnerScreen: ImageView
+declareWinner: Text
+areYouSure: Text
+yesButton: Button
+noButton: Button
+closeButton: Button
-gameController: BlackjackController
-playersCards: Card[*] {collection="ArrayList"}
-dealerCards: Card[*] {collection="ArrayList"}
-bet: int
+menuButton(): void
+yesAction(actionEvent: ActionEvent): void
+noAction(): void
+settingsBackButton(actionEvent: ActionEvent): void
+instructionsButton(): void
+closeButton(): void
+hit(): void
+deal(): void
+stand(): void
+declareWinner(winner: String): void
+updateTotalResult(): void
+setGameController(blackjackController: BlackjackController): void
+showCurrency(): void
+init(): void
+disableHit(): void
+setBet(): void
+updateBalance(): void
+setPlayersCards(playersCards: Card[*]): void
+setDealersCards(dealerCards: Card[*]): void
+clearTable(): void
MainMenuController
(from view)
+blackScreen: ImageView
+areYouSure: Text
+noButton: Button
+yesButton: Button
-gameController: BlackjackController
-userController: UserController
+primaryStage: Stage
-quitButton()
+playButton(actionEvent: ActionEvent): void
+backToMainMenu(actionEvent: ActionEvent): void
+gameHistoryButton(actionEvent: ActionEvent): void
+settingsButton(actionEvent: ActionEvent): void
+quitButton(): void
+loginButton(actionEvent: ActionEvent): void
+registerButton(actionEvent: ActionEvent): void
+yesAction(): void
+noAction(): void
+setGameController(blackjackController: BlackjackController): void
+setUserController(userController: UserController): void
+setStageManager(stageManager: StageManager): void
GameHistoryController
(from view)
+areYouSure: Text
+yesButton: Button
+noButton: Button
+blackScreen: ImageView
+clearHistoryButton: Button
+historyTable: TableView
+gameColumn: TableColumn
+resultColumn: TableColumn
+betColumn: TableColumn
+balanceColumn: TableColumn
+refreshButton: Button
-casinoDAO: CasinoDAO
+gameHistoryBackButton(actionEvent: ActionEvent): void
+clearHistory(): void
+noAction(): void
+yesAction(): void
+getHistory(): ObservableList
+refresh(): void
LoginController
(from view)
-usernameTextField: TextField
-passwordTextField: TextField
-loginButton: Button
-backButton
+loginButton(event: ActionEvent): void
+infoBox(infoMessage: String, headerText: String, title: String): void
-showAlert(alertType, owner, Window, title: String, message, String): void
+backToMainMenu(actionEvent: ActionEvent): void
RegisterController
(from view)
-userController: UserController
+usernameTextField: TextField
+passwordTextField: TextField
+repeatTextField: TextField
+submitButton: Button
+register(event: ActionEvent): void
-showAlert(alertType, owner, Window, title: String, message: String): void
+backToMainMenu(actionEvent: ActionEvent): void
+setUserController(userController: UserController): void

Model: Käyttötapauskaavio

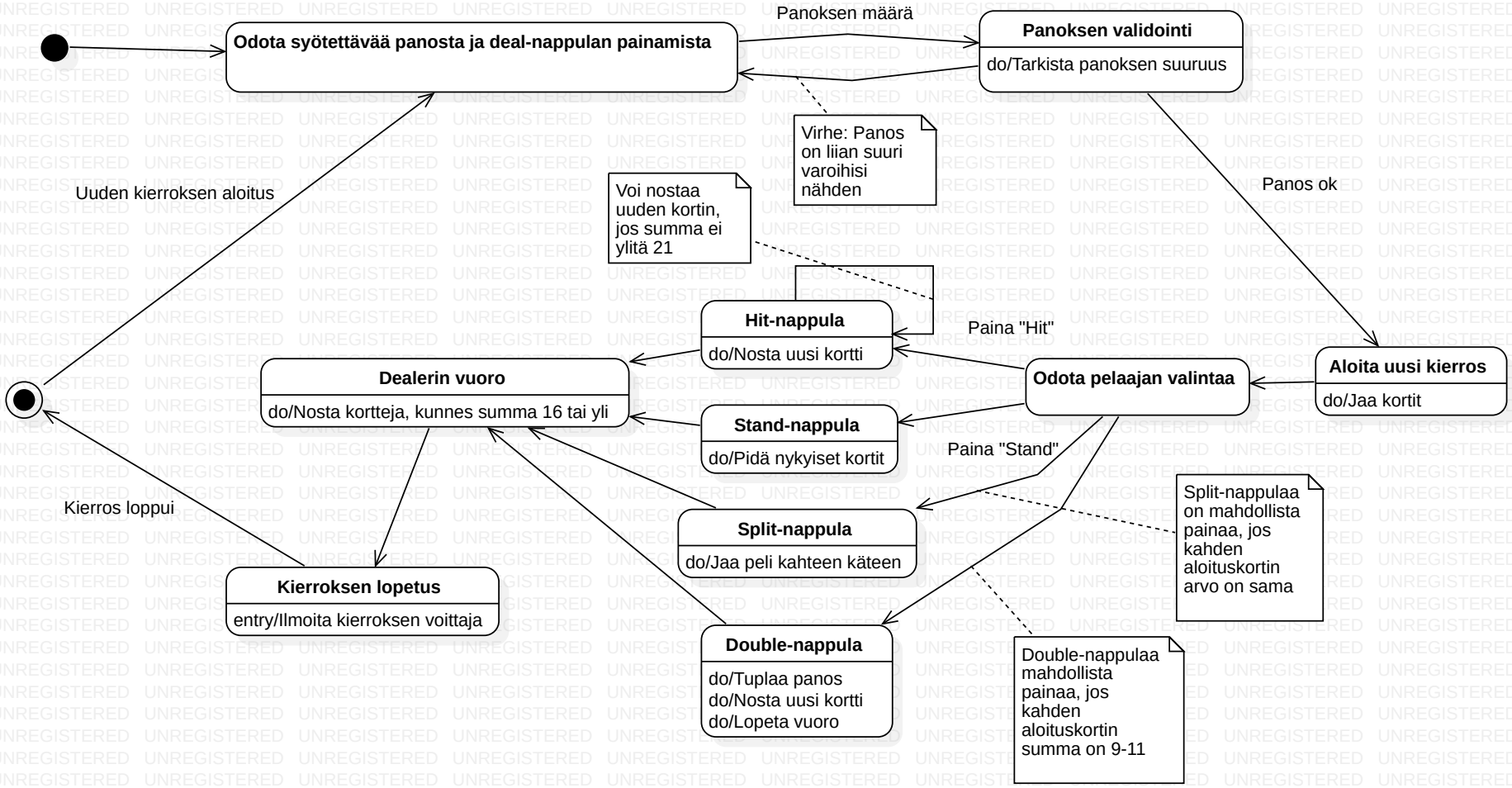


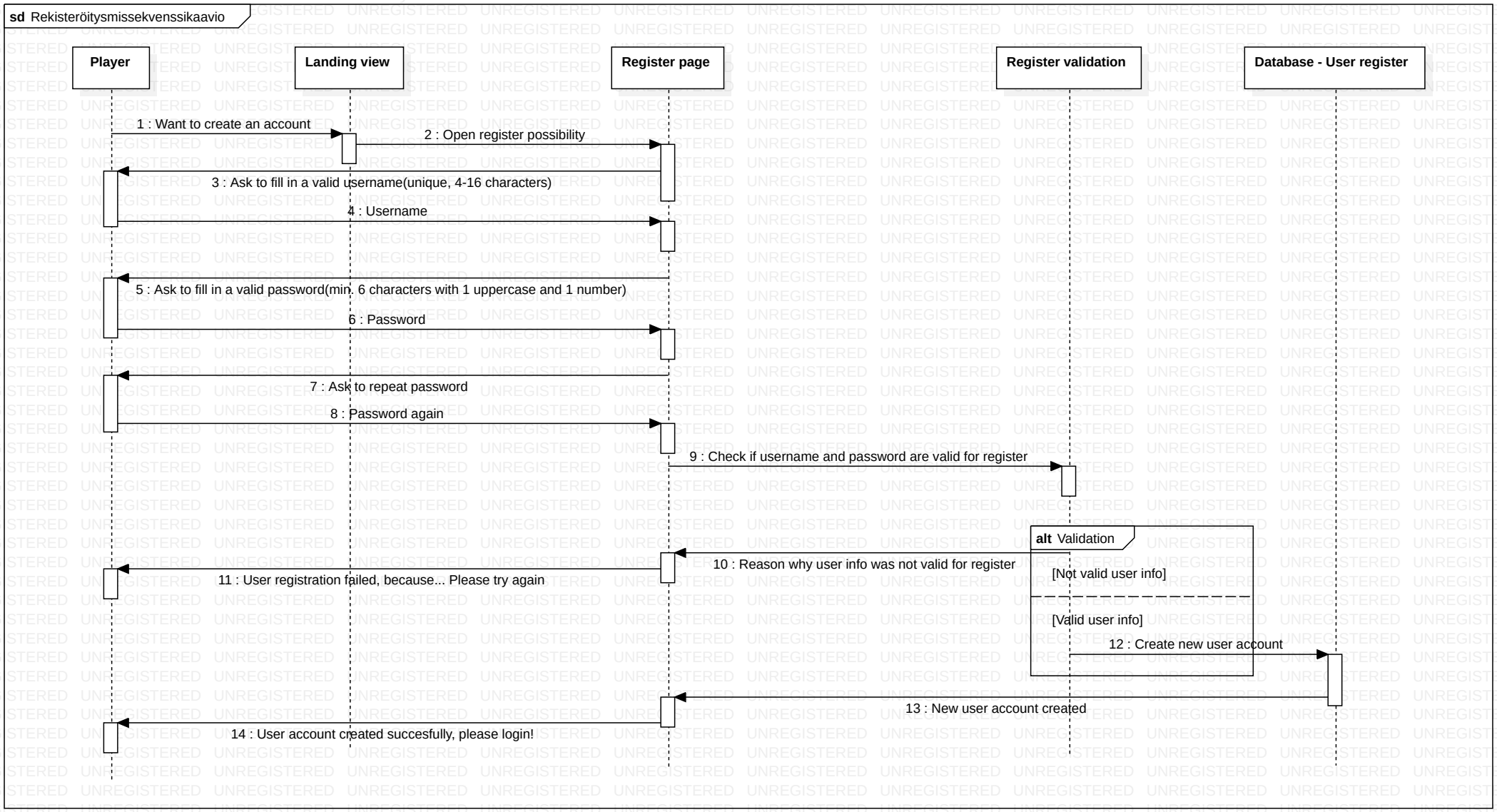

sd Kirjautumissekvenssikaavio
Player
Landing view
Login view
Database
1 : Want to login
2 : Open login option
3 : Enter information and try to login
4 : Is user account valid
alt Query if user account is in register and active
[User account not found]
[User account found]
5 : Return query result
seq User information found from register
6 : Invalid user information, please try again
[Not found]
[Found]
7 : Log in
8 : Player logged in
9 : You are now logged in

Model: Activity1: RekisteröitymisenAktiviteettikaavio



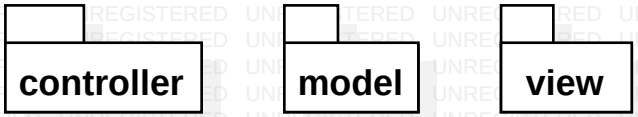
Model::StateMachine1::Peli tila automaatti

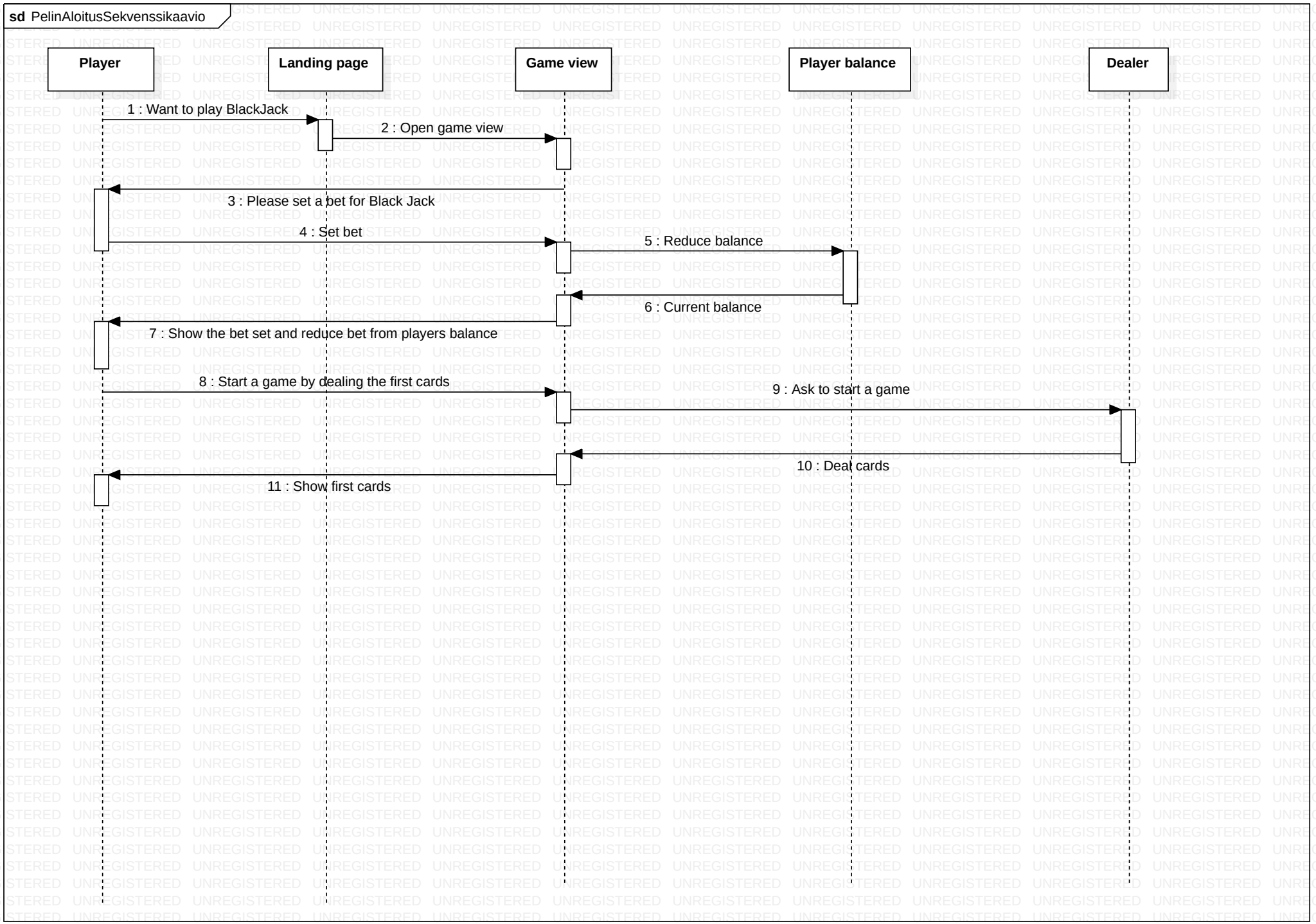



sd Rekisteröitysmissekvenssikaavio
Player
Landing view
Register page
Register validation
Database - User register
1 : Want to create an account
2 : Open register possibility
3 : Ask to fill in a valid username(unique, 4-16 characters)
4 : Username
5 : Ask to fill in a valid password(min. 6 characters with 1 uppercase and 1 number)
6 : Password
7 : Ask to repeat password
8 : Password again
9 : Check if username and password are valid for register
alt Validation
10 : Reason why user info was not valid for register
11 : User registration failed, because... Please try again
[Not valid user info]
[Valid user info]
12 : Create new user account
13 : New user account created
14 : User account created succesfully, please login!

UNREGISTERED

# JavaRevolver: Package Structure




sd PelinAloitusSekvenssikaavio
Player
Landing page
Game view
Player balance
Dealer
1 : Want to play BlackJack
2 : Open game view
3 : Please set a bet for Black Jack
4 : Set bet
5 : Reduce balance
6 : Current balance
7 : Show the bet set and reduce bet from players balance
8 : Start a game by dealing the first cards
9 : Ask to start a game
10 : Deal cards
11 : Show first cards