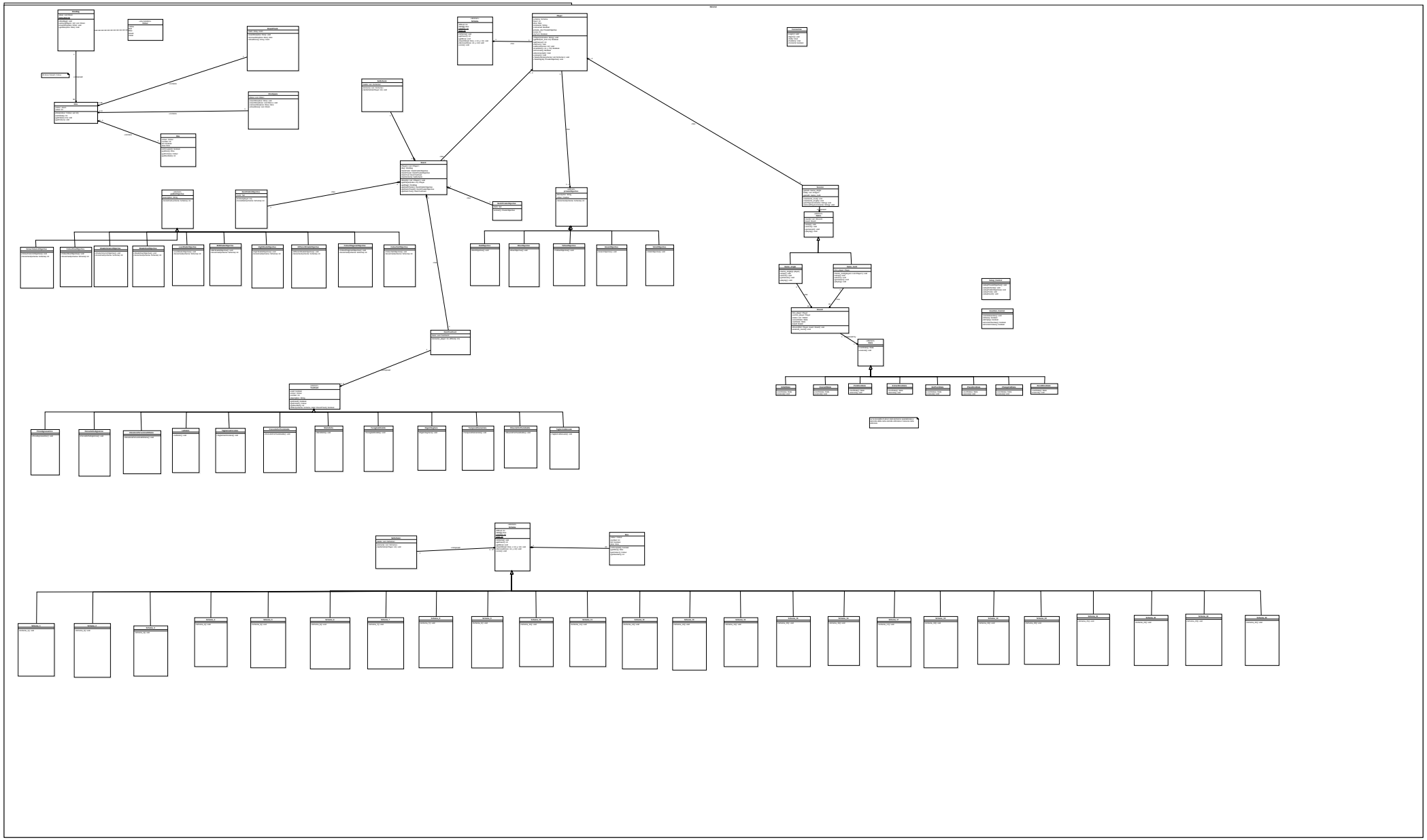
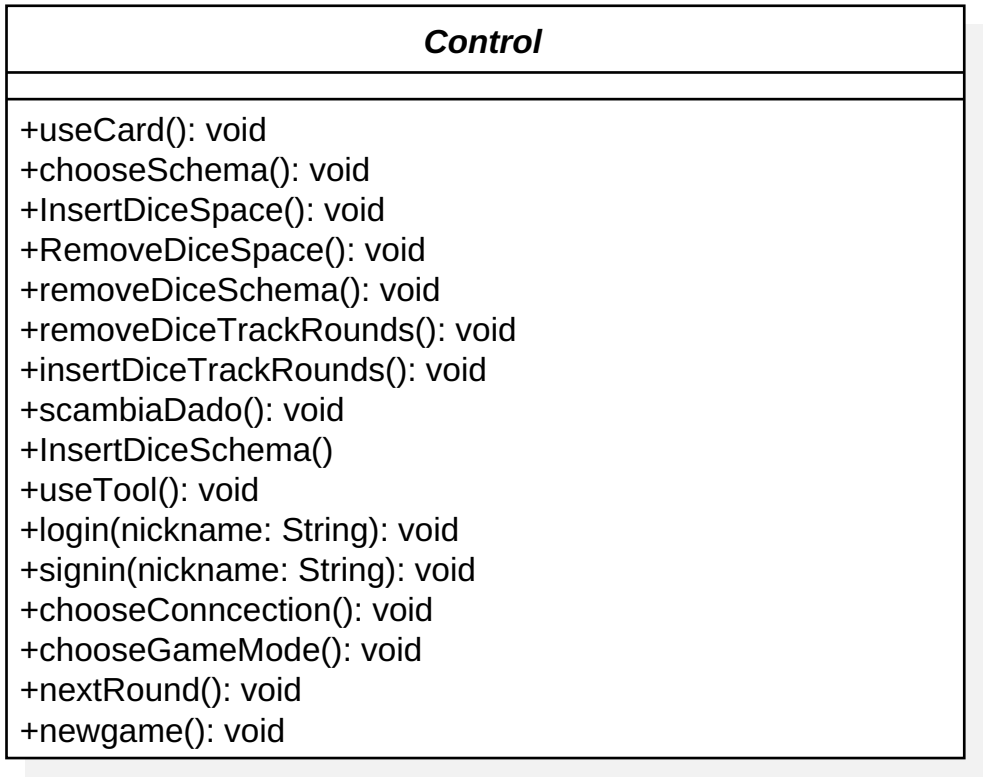
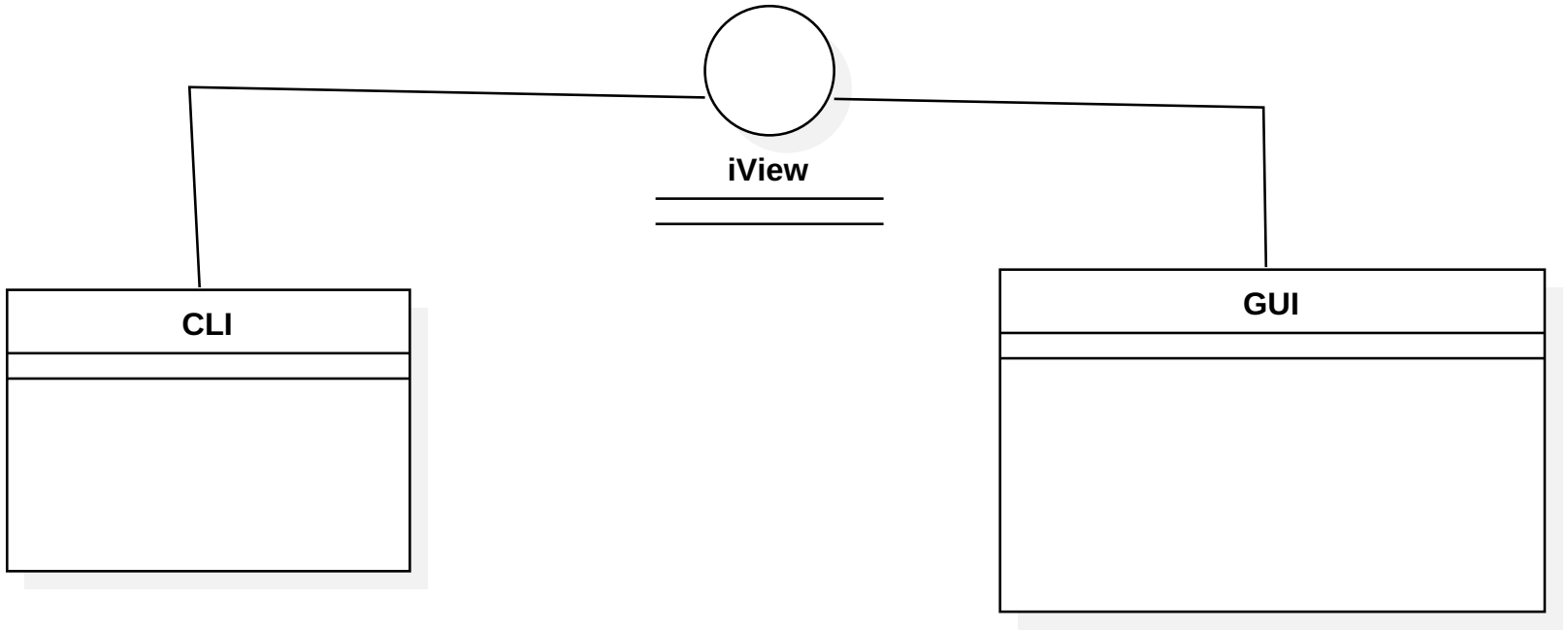
UML::All_Model



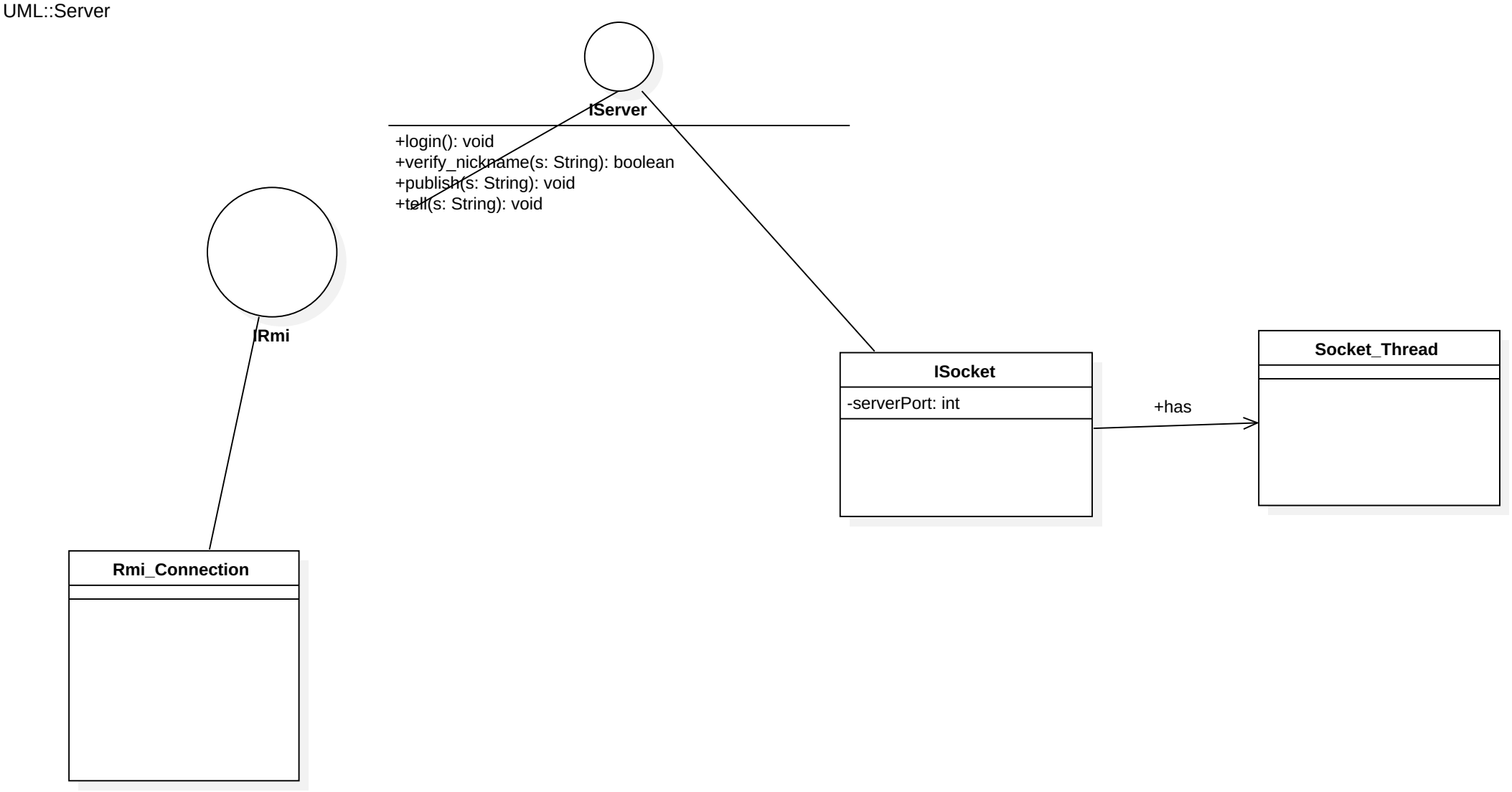
UML::Controller



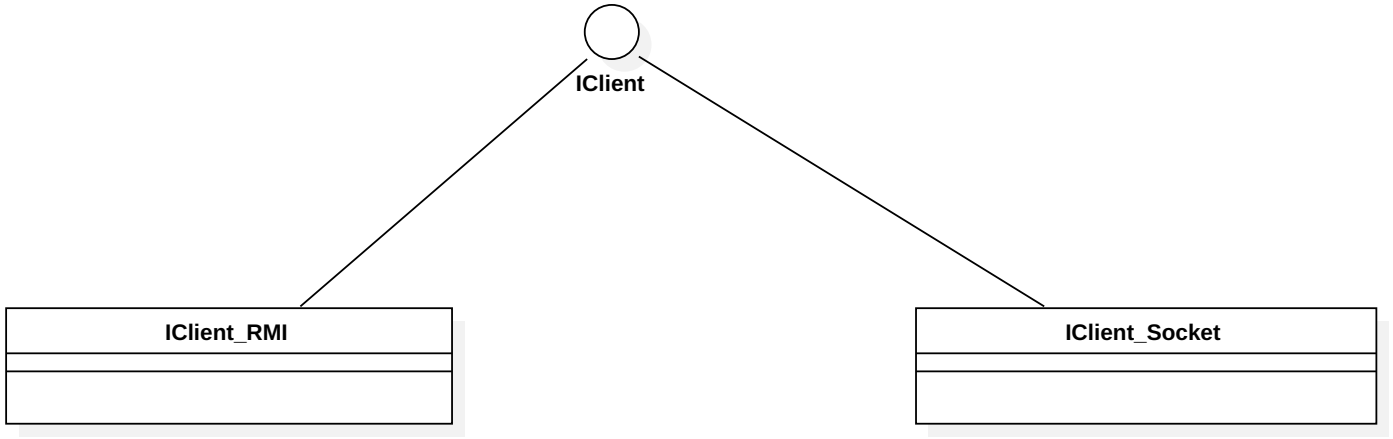
UML::View



UML::Server

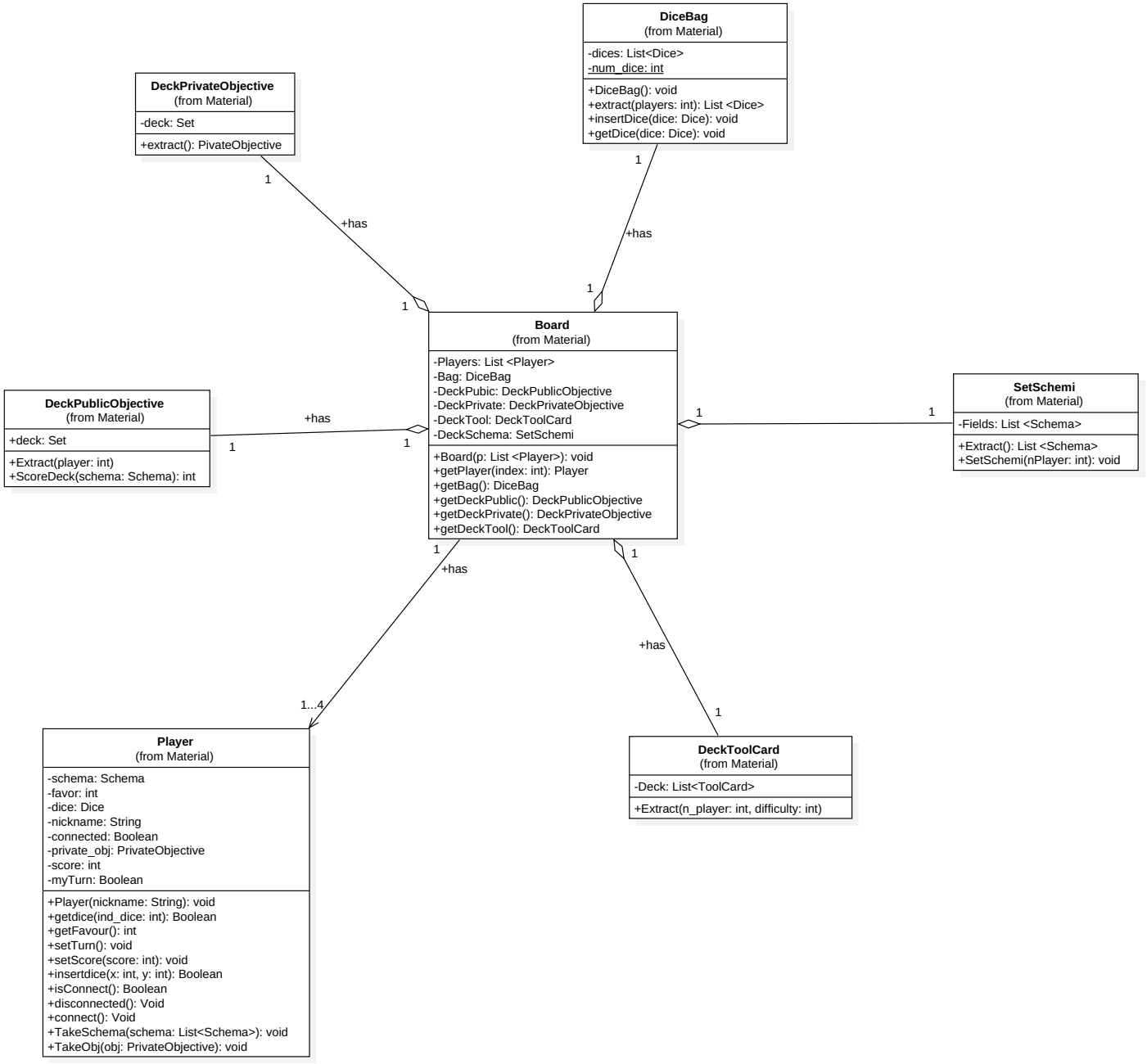

IServer
+login(): void
+verify_nickname(s: String): boolean
+publish(s: String): void
+tell(s: String): void
IRmi
ISocket
-serverPort: int
+has
Socket_Thread
Rmi_Connection

UML::Client

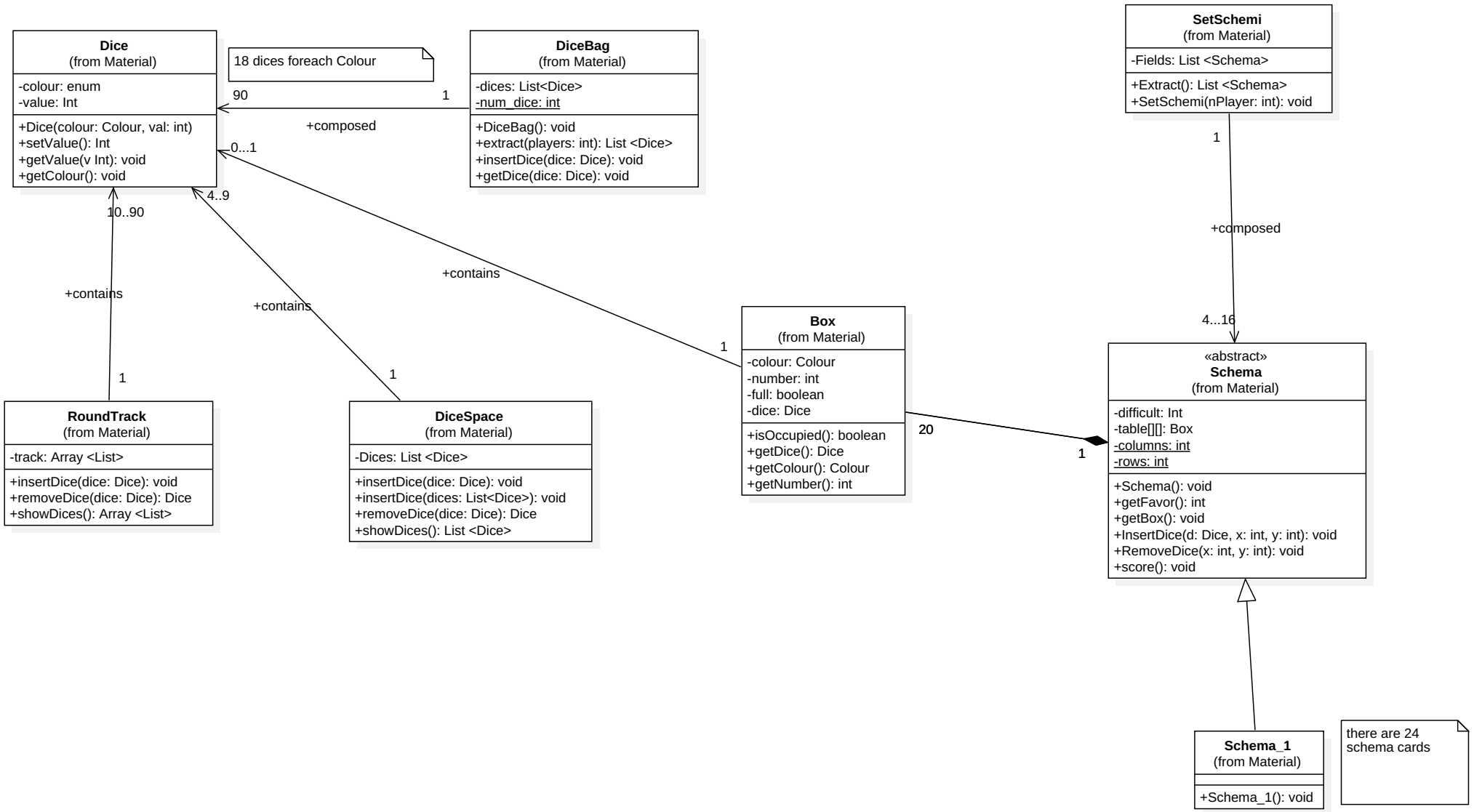


tell(s: String): void
socket: Socket
inSocket: BufferedReader
outSocket: PrintWriter
inKeyBoards: BufferedReader
outVideo: PrintWriter
insert_nickname(s: String)
-connected: boolean
login(): void
play(): void

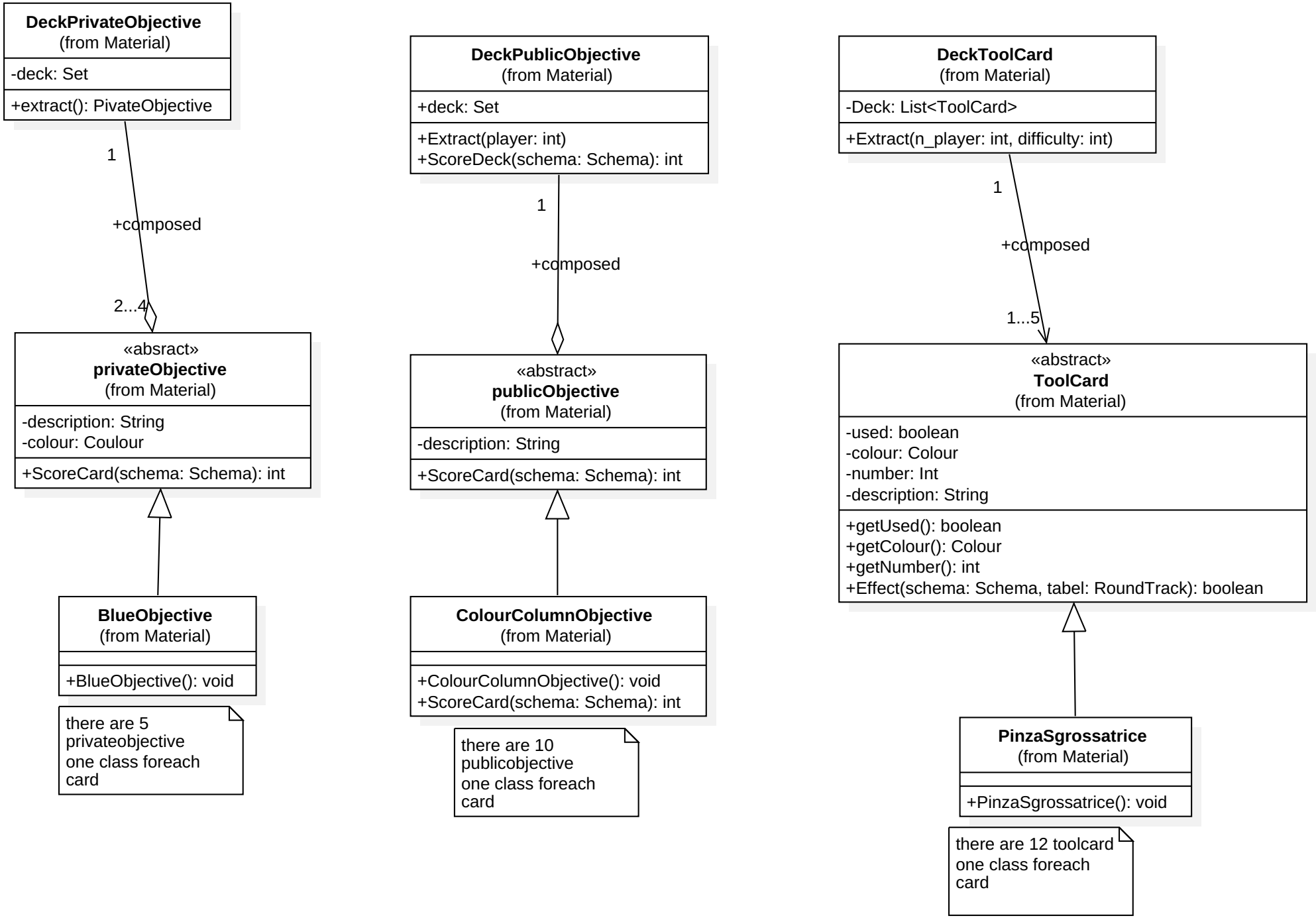
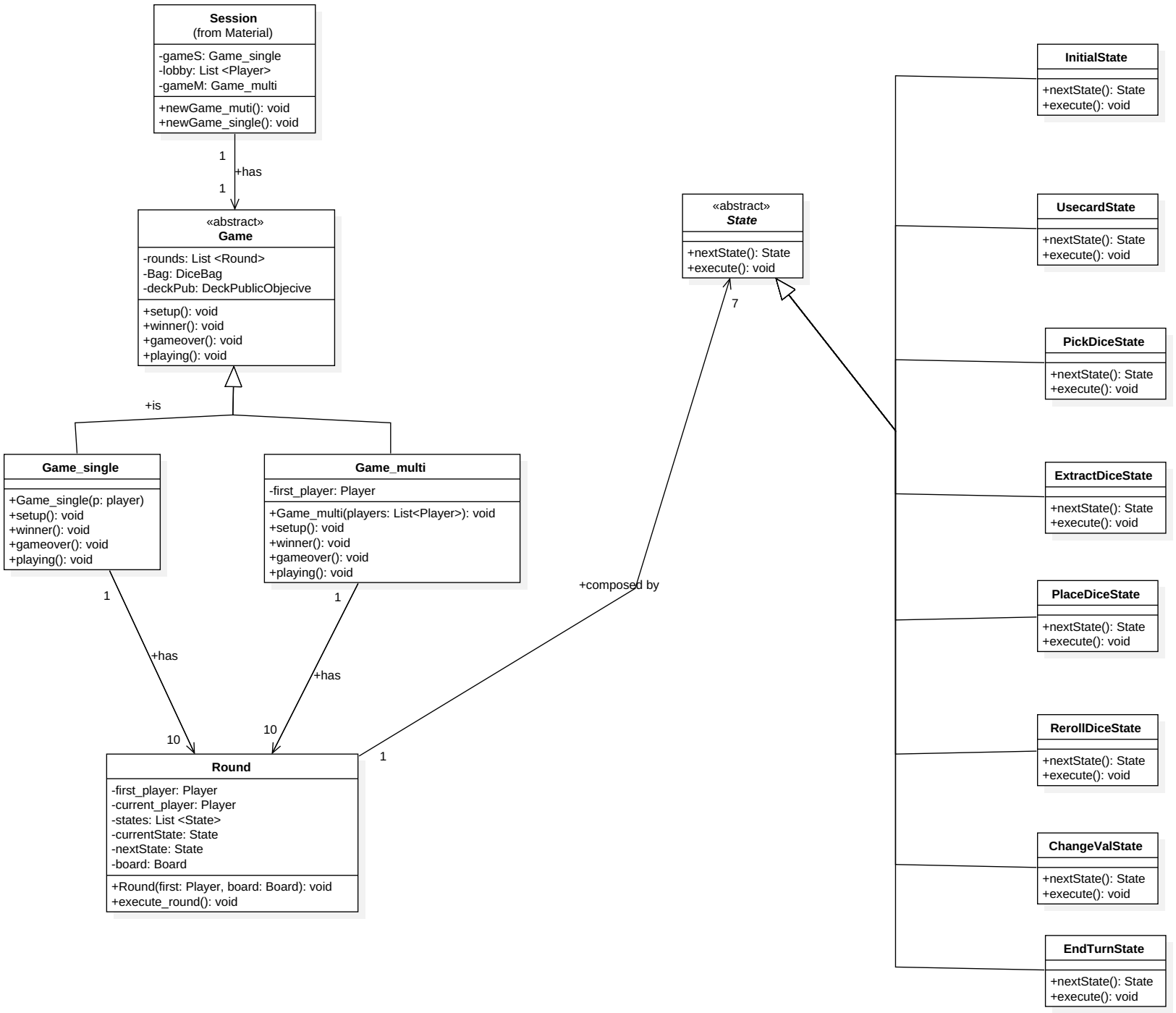
UML::Model


DiceBag
(from Material)
-dices: List<Dice>
-num_dice: int
+DiceBag(): void
+extract(players: int): List <Dice>
+insertDice(dice: Dice): void
+getDice(dice: Dice): void
DeckPrivateObject
(from Material)
-deck: Set
+extract(): PivateObjective
1
+has
1
1
+has
1
Board
(from Material)
-Players: List <Player>
-Bag: DiceBag
-DeckPubic: DeckPublicObjective
-DeckPrivate: DeckPrivateObjective
-DeckTool: DeckToolCard
-DeckSchema: SetSchemi
+Board(p: List <Player>): void
+getPlayer(index: int): Player
+getBag(): DiceBag
+getDeckPublic(): DeckPublicObjective
+getDeckPrivate(): DeckPrivateObjective
+getDeckTool(): DeckToolCard
DeckPublicObjective
(from Material)
+deck: Set
+Extract(player: int)
+ScoreDeck(schema: Schema): int
+has
1
1
1
1
SetSchemi
(from Material)
-Fields: List <Schema>
+Extract(): List <Schema>
+SetSchemi(nPlayer: int): void
1
+has
1...4
1
+has
1
Player
(from Material)
-schema: Schema
-favor: int
-dice: Dice
-nickname: String
-connected: Boolean
-private_obj: PrivateObjective
-score: int
-myTurn: Boolean
+Player(nickname: String): void
+getdice(ind_dice: int): Boolean
+getFavour(): int
+setTurn(): void
+setScore(score: int): void
+insertdice(x: int, y: int): Boolean
+isConnect(): Boolean
+disconnected(): Void
+connect(): Void
+TakeSchema(schema: List<Schema>): void
+TakeObj(obj: PrivateObjective): void
DeckToolCard
(from Material)
-Deck: List<ToolCard>
+Extract(n_player: int, difficulty: int)

UML::Dice+Schema



UML::Cards


DeckPrivateObjective
(from Material)
-deck: Set
+extract(): PivateObjective
1
+composed
2...4
«absract»
privateObjective
(from Material)
-description: String
-colour: Coulour
+ScoreCard(schema: Schema): int
BlueObjective
(from Material)
+BlueObjective(): void
there are 5 privateobjective one class foreach card
DeckPublicObjective
(from Material)
+deck: Set
+Extract(player: int)
+ScoreDeck(schema: Schema): int
1
+composed
«abstract»
publicObjective
(from Material)
-description: String
+ScoreCard(schema: Schema): int
ColourColumnObjective
(from Material)
+ColourColumnObjective(): void
+ScoreCard(schema: Schema): int
there are 10 publicobjective one class foreach card
DeckToolCard
(from Material)
-Deck: List<ToolCard>
+Extract(n_player: int, difficulty: int)
1
+composed
1...5
«abstract»
ToolCard
(from Material)
-used: boolean
-colour: Colour
-number: Int
-description: String
+getUsed(): boolean
+getColour(): Colour
+getNumber(): int
+Effect(schema: Schema, tabel: RoundTrack): boolean
PinzaSgrossatrice
(from Material)
+PinzaSgrossatrice(): void
there are 12 toolcard one class foreach card

UML::Session+Game+Round+Turn