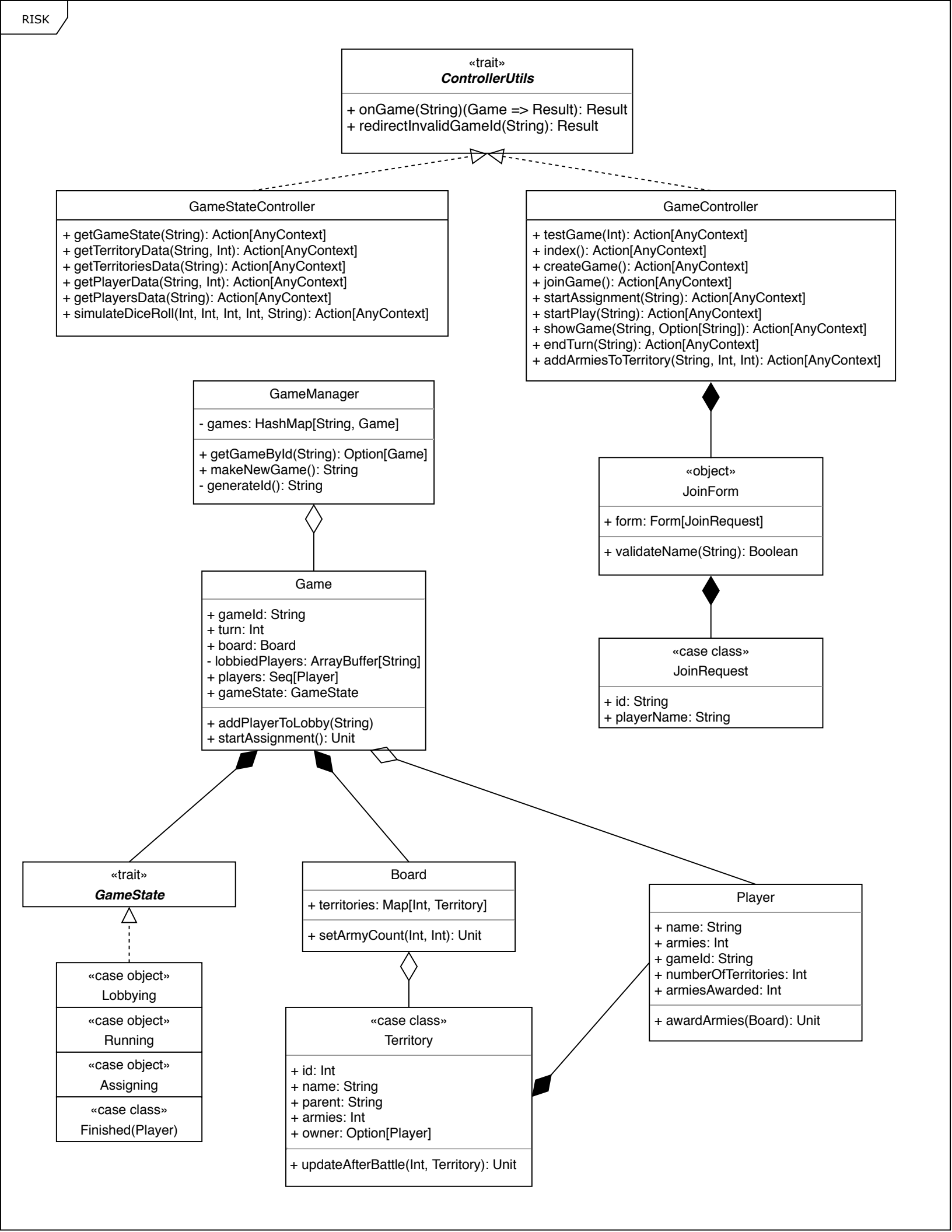
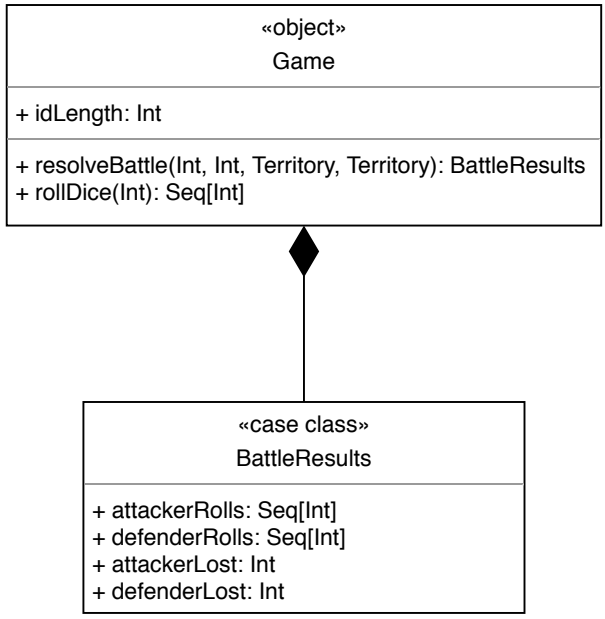
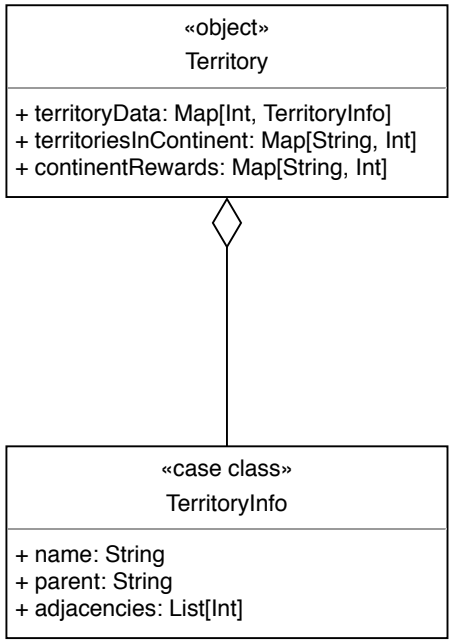

RISK
«trait»
ControllerUtils
+ onGame(String)(Game => Result): Result
+ redirectInvalidGameId(String): Result
GameStateController
+ getGameState(String): Action[AnyContext]
+ getTerritoryData(String, Int): Action[AnyContext]
+ getTerritoriesData(String): Action[AnyContext]
+ getPlayerData(String, Int): Action[AnyContext]
+ getPlayersData(String): Action[AnyContext]
+ simulateDiceRoll(Int, Int, Int, Int, String): Action[AnyContext]
GameController
+ testGame(Int): Action[AnyContext]
+ index(): Action[AnyContext]
+ createGame(): Action[AnyContext]
+ joinGame(): Action[AnyContext]
+ startAssignment(String): Action[AnyContext]
+ startPlay(String): Action[AnyContext]
+ showGame(String, Option[String]): Action[AnyContext]
+ endTurn(String): Action[AnyContext]
+ addArmiesToTerritory(String, Int, Int): Action[AnyContext]
GameManager
- games: HashMap[String, Game]
+ getGameById(String): Option[Game]
+ makeNewGame(): String
- generateId(): String
«object»
JoinForm
+ form: Form[JoinRequest]
+ validateName(String): Boolean
«case class»
JoinRequest
+ id: String
+ playerName: String
Game
+ gameId: String
+ turn: Int
+ board: Board
- lobbiedPlayers: ArrayBuffer[String]
+ players: Seq[Player]
+ gameState: GameState
+ addPlayerToLobby(String)
+ startAssignment(): Unit
«trait»
GameState
«case object»
Lobbying
«case object»
Running
«case object»
Assigning
«case class»
Finished(Player)
Board
+ territories: Map[Int, Territory]
+ setArmyCount(Int, Int): Unit
«case class»
Territory
+ id: Int
+ name: String
+ parent: String
+ armies: Int
+ owner: Option[Player]
+ updateAfterBattle(Int, Territory): Unit
Player
+ name: String
+ armies: Int
+ gameId: String
+ numberOfTerritories: Int
+ armiesAwarded: Int
+ awardArmies(Board): Unit


«object»
Game
+ idLength: Int
+ resolveBattle(Int, Int, Territory, Territory): BattleResults
+ rollDice(Int): Seq[Int]
«case class»
BattleResults
+ attackerRolls: Seq[Int]
+ defenderRolls: Seq[Int]
+ attackerLost: Int
+ defenderLost: Int


«object»
Territory
+ territoryData: Map[Int, TerritoryInfo]
+ territoriesInContinent: Map[String, Int]
+ continentRewards: Map[String, Int]
«case class»
TerritoryInfo
+ name: String
+ parent: String
+ adjacencies: List[Int]