Task 7 Borderland Survival

In a distorted other world known as the Borderland, some creatures are forced to play deadly games to survive. Every game tends to test intellect, brawn, teamwork, or brutality. Players must scavenge for supplies, form alliances, and solve increasingly complicated puzzles in their efforts to remain alive (forming alliances and scavenging are not modeled).

Every player has a name, a mental stat, a physical stat, and a fear level. Players may gain or lose stats based on game events. When the fear level reaches a certain threshold, the player will freeze and become inactive.

Each game has a name, a type (mental or physical), and a difficulty level. Mental games require the mental stat, and physical ones the physical stat. The game can be played by a group of players and everyone whose appropriate stat is above the difficulty level wins. Players who win will be less afraid, and players who lose will be more afraid.

Players are rewarded by cards for winning a game. All cards possess a suit and number (i.e., 7 of Spades, 3 of Hearts), and having all 52 cards will enable one to exit.

Games are occurring in an arena. The arena has a name, a series of games, and an environmental hazard which affects players (e.g., fire, building collapse, mist). When the hazard activates, players become more afraid.

The Game Master controls the games. They control the hazards in arenas and pick the next game randomly to create suspense.

The larger flow is controlled by the Borderland Manager. It sets up everything for the games, advances time by starting a game, and checks if the games are over. It can also observe the players and print them for the viewers.

The game ends when all the cards have been collected or when all the players are inactive. Play your survival, test yourself to limits, and try to get out of the Borderland!

Input file examples

players.txt

• Format
Name Mental Physical
• Sample input
Maksud 7 5
Farizi 6 8
Silva 9 3

games.txt

• Format
Name Type Difficulty
• Sample input
Bridge_Puzzle mental 6
Tag_Arena physical 5
Quiz_of_Death mental 7

arena.txt

• Format
Name Hazard
• Sample input
Abandoned_Theme_Park Fog

7.1 Additional functionality

Psychological tensions and loyalties are introduced into the Borderland to create room for player agency and strategic play. Every character has a behaviour type – Rational, Aggressive, Cautious, or Unstable – that determines the manner in which he or she responds to fear and game type. Each influences them behind the scenes to influence their decision-making, fear gain, and play. For example, Cautious characters avoid dangerous games, and Aggressive characters ignore fear but utilize stamina. The system also has trust-based relations that depreciate over time without being solidified by mutual success or card trades. Betrayals invoke fear and alter the behaviour of the characters.

7.2 State diagram

Design a UML state machine that starts every character in the Rational state and allows transitions to Aggressive, Cautious, or Unstable based on simple triggers: fear spikes push Rational to Cautious and Cautious to Unstable, betrayal drives Rational/Cautious to Aggressive, prolonged aggression with low stamina turns Aggressive to Unstable, while rest or trust-building can move Aggressive/Cautious back toward Rational and Unstable to Cautious to Rational.

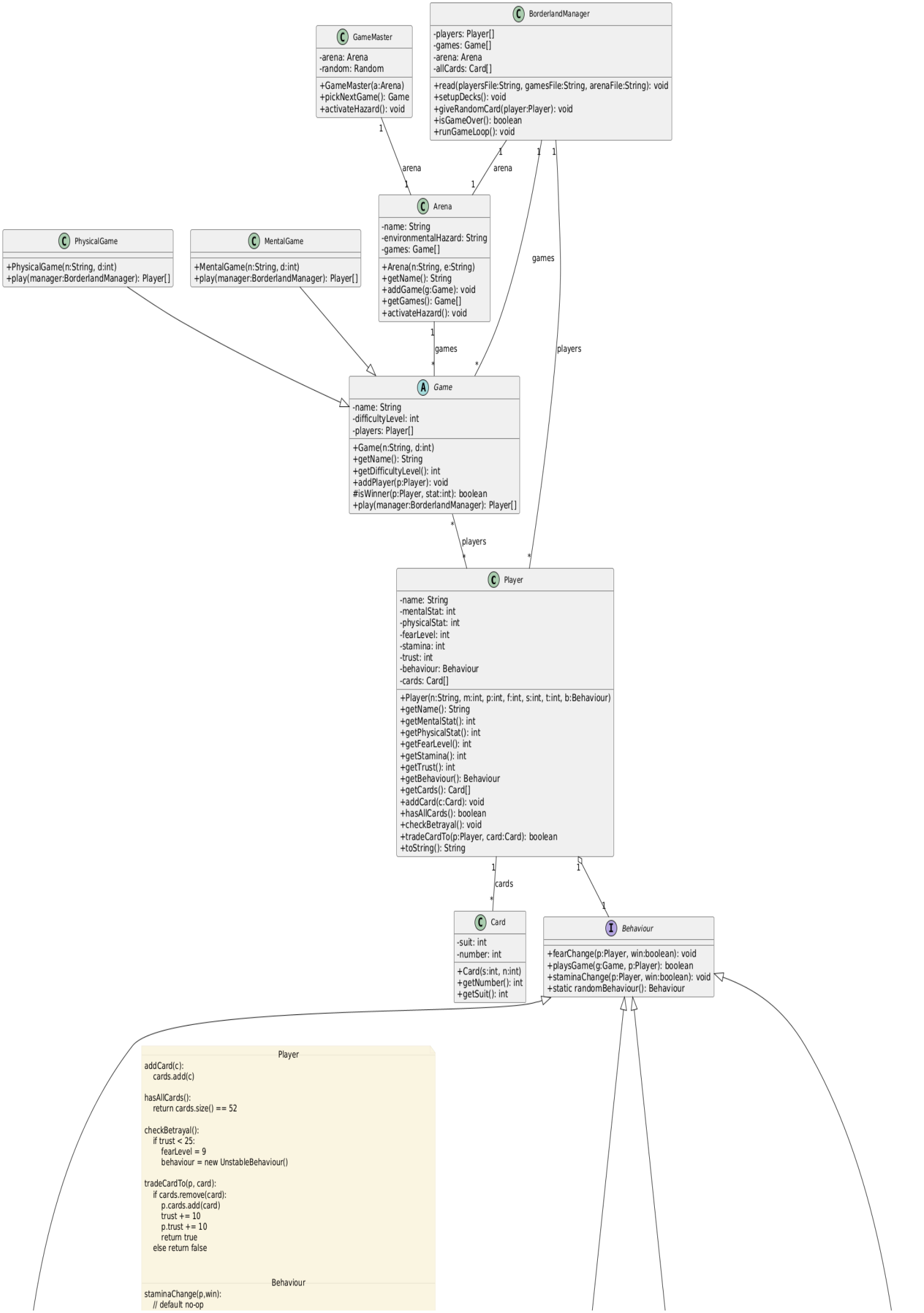
7.3 Scenarios to model with sequence diagrams

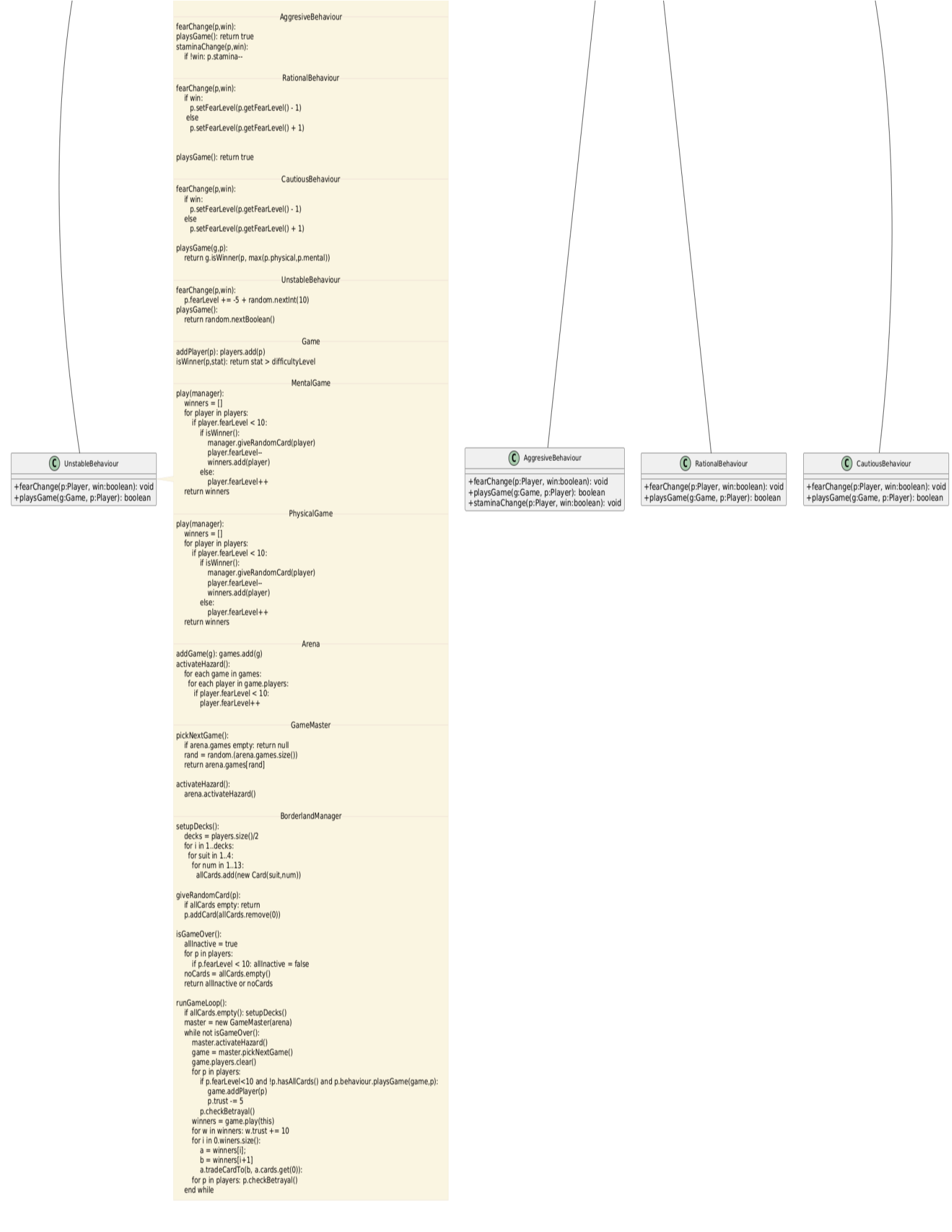
7.3.1 Scenario

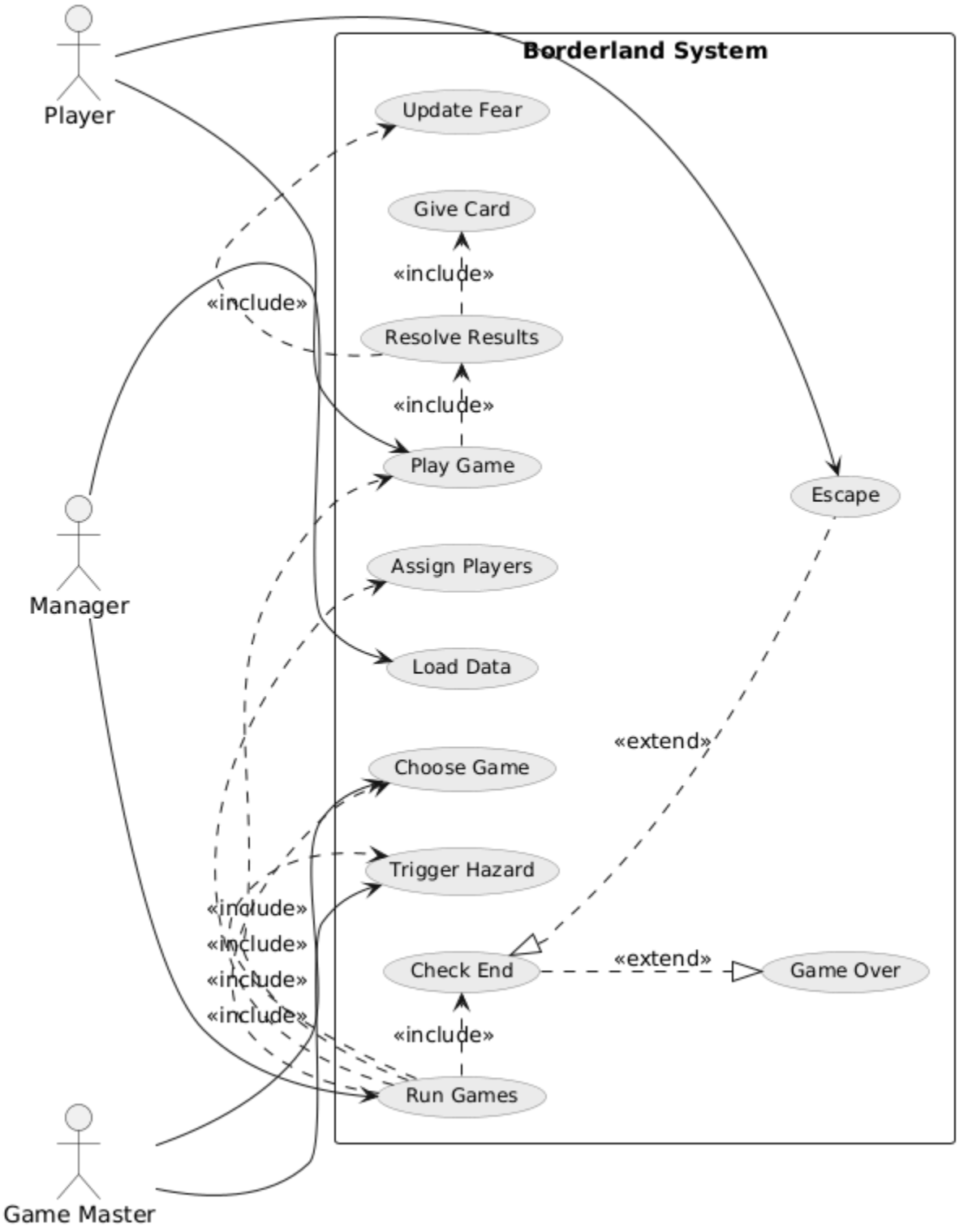
1: Typical game Draw a sequence diagram for the “Start Game” interaction: the Borderland Manager instructs the Game Master to begin the next round; the Game Master randomly selects a Game from the Arena’s queue, notifies the Arena to set itself up, and then broadcasts a “game start” message to all Players, who each respond by checking their relevant stat (mental or physical) against the Game’s difficulty and returning either “win” or “lose”; the Game Master tallies results, updates each Player’s fear level, and awards cards to winners before reporting back to the Borderland Manager.

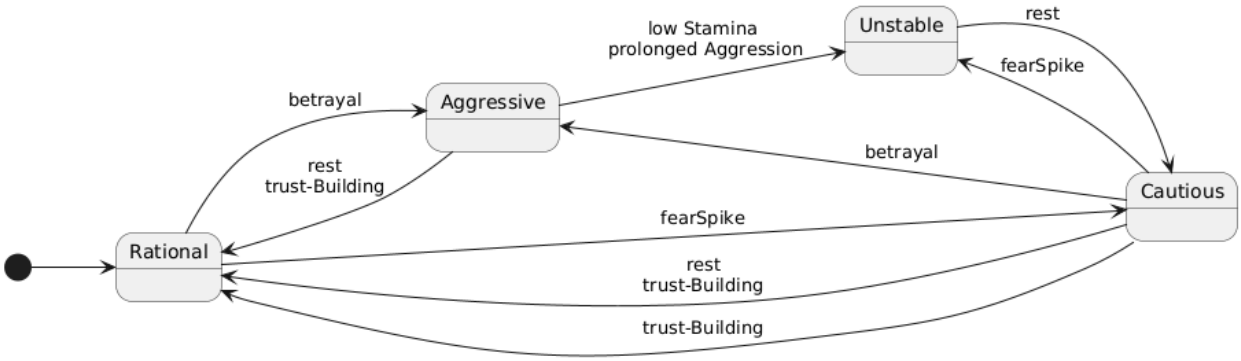
7.3.2 Scenario

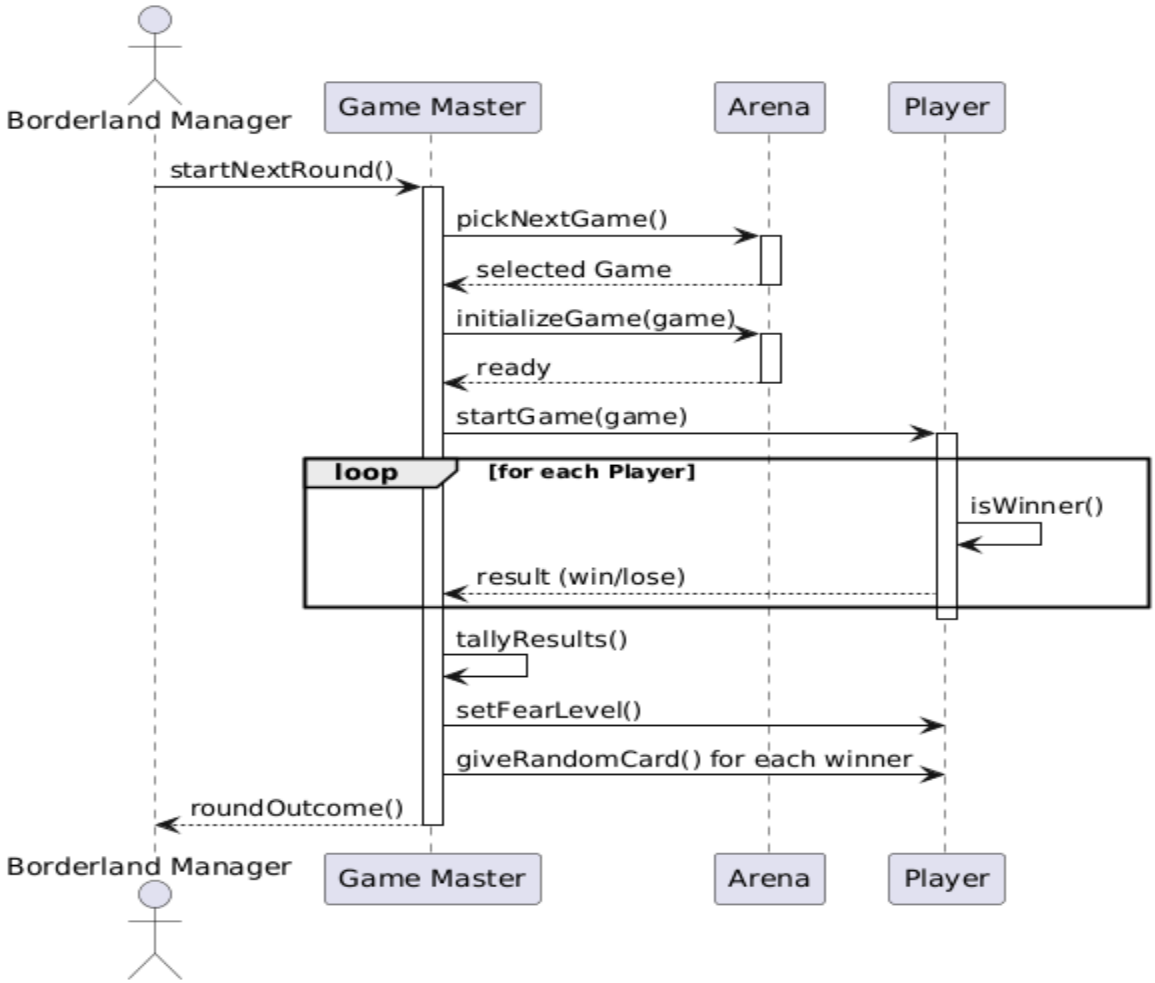
2: Victory! Create a sequence diagram depicting the moment a Player wins their last needed card: the Game object notifies the Game Master of the Player’s vic14 tory, the Game Master updates the Player’s card collection, discovers the set now contains all 52 unique cards, and immediately informs the Borderland Manager. The Manager validates the full deck, broadcasts an “exit granted” message, orders the Arena to open an exit path, and logs the event for spectators while the victorious Player sends a final “leave borderland” confirmation back to both the Game Master and Manager.

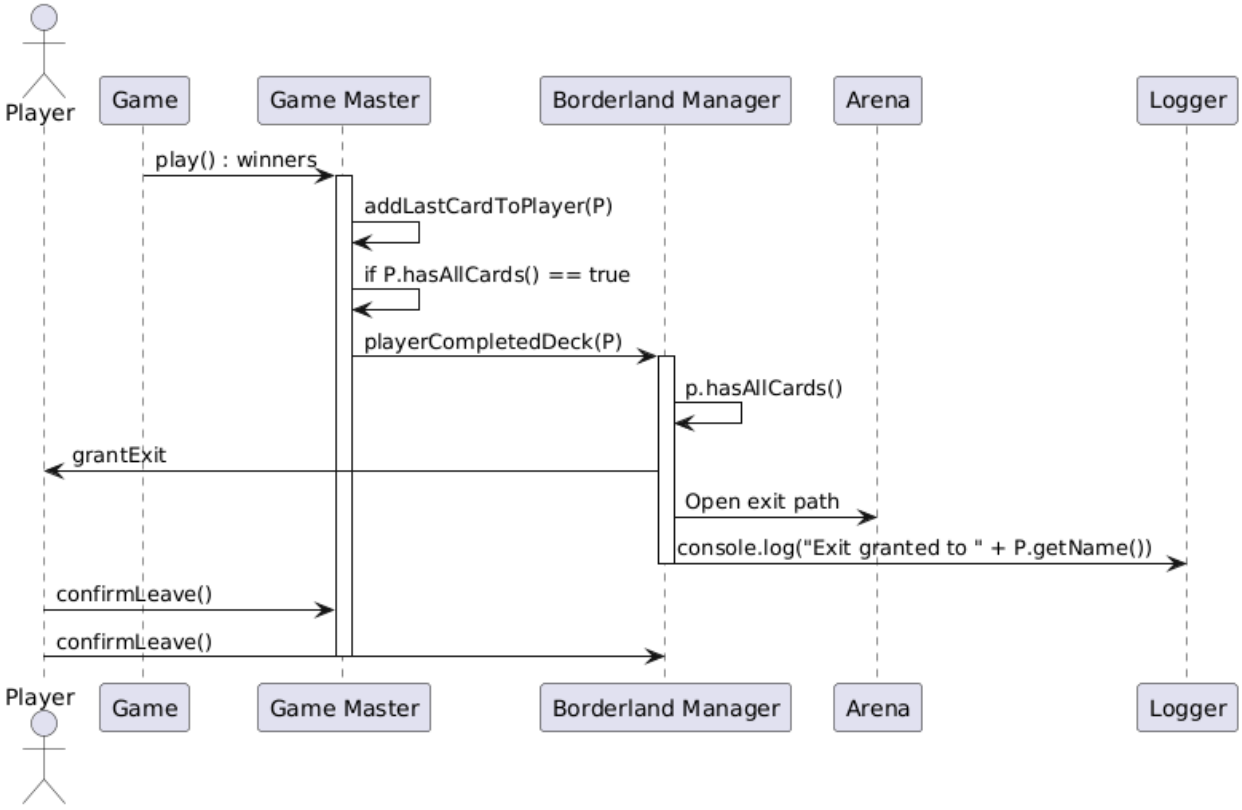

BorderlandManager
-players: Player[]
-games: Game[]
-arena: Arena
-allCards: Card[]
+read(playersFile:String, gamesFile:String, arenaFile:String): void
+setupDecks(): void
+giveRandomCard(player:Player): void
+isGameOver(): boolean
+runGameLoop(): void
GameMaster
-arena: Arena
-random: Random
+GameMaster(a:Arena)
+pickNextGame(): Game
+activateHazard(): void
Arena
-name: String
-environmentalHazard: String
-games: Game[]
+Arena(n:String, e:String)
+getName(): String
+addGame(g:Game): void
+getGames(): Game[]
+activateHazard(): void
PhysicalGame
+PhysicalGame(n:String, d:int)
+play(manager:BorderlandManager): Player[]
MentalGame
+MentalGame(n:String, d:int)
+play(manager:BorderlandManager): Player[]
Game
-name: String
-difficultyLevel: int
-players: Player[]
+Game(n:String, d:int)
+getName(): String
+getDifficultyLevel(): int
+addPlayer(p:Player): void
#isWinner(p:Player, stat:int): boolean
+play(manager:BorderlandManager): Player[]
Player
-name: String
-mentalStat: int
-physicalStat: int
-fearLevel: int
-stamina: int
-trust: int
-behaviour: Behaviour
-cards: Card[]
+Player(n:String, m:int, p:int, f:int, s:int, t:int, b:Behaviour)
+getName(): String
+getMentalStat(): int
+getPhysicalStat(): int
+getFearLevel(): int
+getStamina(): int
+getTrust(): int
+getBehaviour(): Behaviour
+getCards(): Card[]
+addCard(c:Card): void
+hasAllCards(): boolean
+checkBetrayal(): void
+tradeCardTo(p:Player, card:Card): boolean
+toString(): String
Card
-suit: int
-number: int
+Card(s:int, n:int)
+getNumber(): int
+getSuit(): int
Behaviour
+fearChange(p:Player, win:boolean): void
+playsGame(g:Game, p:Player): boolean
+staminaChange(p:Player, win:boolean): void
+static randomBehaviour(): Behaviour
arena
games
players
cards
Player
addCard(c):
cards.add(c)
hasAllCards():
return cards.size() == 52
checkBetrayal():
if trust < 25:
fearLevel = 9
behaviour = new UnstableBehaviour()
tradeCardTo(p, card):
if cards.remove(card):
p.cards.add(card)
trust += 10
p.trust += 10
return true
else return false
Behaviour
staminaChange(p,win):
// default no-op


UnstableBehaviour
+fearChange(p:Player, win:boolean): void
+playsGame(g:Game, p:Player): boolean
AggresiveBehaviour
+fearChange(p:Player, win:boolean): void
+playsGame(g:Game, p:Player): boolean
+staminaChange(p:Player, win:boolean): void
RationalBehaviour
+fearChange(p:Player, win:boolean): void
+playsGame(g:Game, p:Player): boolean
CautiousBehaviour
+fearChange(p:Player, win:boolean): void
+playsGame(g:Game, p:Player): boolean
AggresiveBehaviour
fearChange(p,win):
playsGame(): return true
staminaChange(p,win):
if !win: p.stamina--
RationalBehaviour
fearChange(p,win):
if win:
p.setFearLevel(p.getFearLevel() - 1)
else
p.setFearLevel(p.getFearLevel() + 1)
playsGame(): return true
CautiousBehaviour
fearChange(p,win):
if win:
p.setFearLevel(p.getFearLevel() - 1)
else
p.setFearLevel(p.getFearLevel() + 1)
playsGame(g,p):
return g.isWinner(p, max(p.physical,p.mental))
UnstableBehaviour
fearChange(p,win):
p.fearLevel += -5 + random.nextInt(10)
playsGame():
return random.nextBoolean()
Game
addPlayer(p): players.add(p)
isWinner(p,stat): return stat > difficultyLevel
MentalGame
play(manager):
winners = []
for player in players:
if player.fearLevel < 10:
if isWinner():
manager.giveRandomCard(player)
player.fearLevel--
winners.add(player)
else:
player.fearLevel++
return winners
PhysicalGame
play(manager):
winners = []
for player in players:
if player.fearLevel < 10:
if isWinner():
manager.giveRandomCard(player)
player.fearLevel--
winners.add(player)
else:
player.fearLevel++
return winners
Arena
addGame(g): games.add(g)
activateHazard():
for each game in games:
for each player in game.players:
if player.fearLevel < 10:
player.fearLevel++
GameMaster
pickNextGame():
if arena.games empty: return null
rand = random.(arena.games.size())
return arena.games[rand]
activateHazard():
arena.activateHazard()
BorderlandManager
setupDecks():
decks = players.size()/2
for i in 1..decks:
for suit in 1..4:
for num in 1..13:
allCards.add(new Card(suit,num))
giveRandomCard(p):
if allCards empty: return
p.addCard(allCards.remove(0))
isGameOver():
allInactive = true
for p in players:
if p.fearLevel < 10: allInactive = false
noCards = allCards.empty()
return allInactive or noCards
runGameLoop():
if allCards.empty(): setupDecks()
master = new GameMaster(arena)
while not isGameOver():
master.activateHazard()
game = master.pickNextGame()
game.players.clear()
for p in players:
if p.fearLevel<10 and !p.hasAllCards() and p.behaviour.playsGame(game,p):
game.addPlayer(p)
p.trust -= 5
p.checkBetrayal()
winners = game.play(this)
for w in winners: w.trust += 10
for i in 0.winers.size():
a = winners[i];
b = winners[i+1]
a.tradeCardTo(b, a.cards.get(0));
for p in players: p.checkBetrayal()
end while


Player
Manager
Game Master
Borderland System
Update Fear
Give Card
Resolve Results
Play Game
Assign Players
Load Data
Choose Game
Trigger Hazard
Check End
Run Games
Escape
Game Over
«include»
«include»
«include»
«include»
«include»
«include»
«include»
«extend»
«extend»


Unstable
low Stamina
prolonged Aggression
rest
fearSpike
Aggressive
betrayal
betrayal
rest
trust-Building
Cautious
fearSpike
Rational
rest
trust-Building
trust-Building


Borderland Manager
Game Master
Arena
Player
startNextRound()
pickNextGame()
selected Game
initializeGame(game)
ready
startGame(game)
loop
[for each Player]
isWinner()
result (win/lose)
tallyResults()
setFearLevel()
giveRandomCard() for each winner
roundOutcome()
Borderland Manager
Game Master
Arena
Player


Player
Game
Game Master
Borderland Manager
Arena
Logger
play() : winners
addLastCardToPlayer(P)
if P.hasAllCards() == true
playerCompletedDeck(P)
p.hasAllCards()
grantExit
Open exit path
console.log("Exit granted to " + P.getName())
confirmLeave()
confirmLeave()
Player
Game
Game Master
Borderland Manager
Arena
Logger