Use Case Details

Version •

Table of Contents

Model

Package in package ''

<div align="right">

Model
Version   Phase 1.0 Proposed
Oliver created on 2017. 11. 22.. Last modified 2017. 11. 22.

</div>

Use Case Model

Package in package 'Model'

<div align="right">

Use Case Model
Version   Phase 1.0 Proposed
Szabi created on 2017. 02. 21.. Last modified 2017. 02. 21.

</div>

Actors

Package in package 'Use Case Model'

<div align="right">

Actors
Version 1.0 Phase 1.0 Proposed
Szabi created on 2017. 02. 21.. Last modified 2017. 02. 21.

</div>

Actors diagram

Use Case diagram in package 'Actors'

<div align="right">

Actors
Version 1.0
Oliver created on 2017. 02. 21.. Last modified 2017. 03. 23.

</div>

Figure 1: Actors

Application Start-up, Run and Shutdown

Package in package 'Use Case Model'

Application Start-up and Shutdown Use Cases diagram

Use Case diagram in package 'Application Start-up, Run and Shutdown'
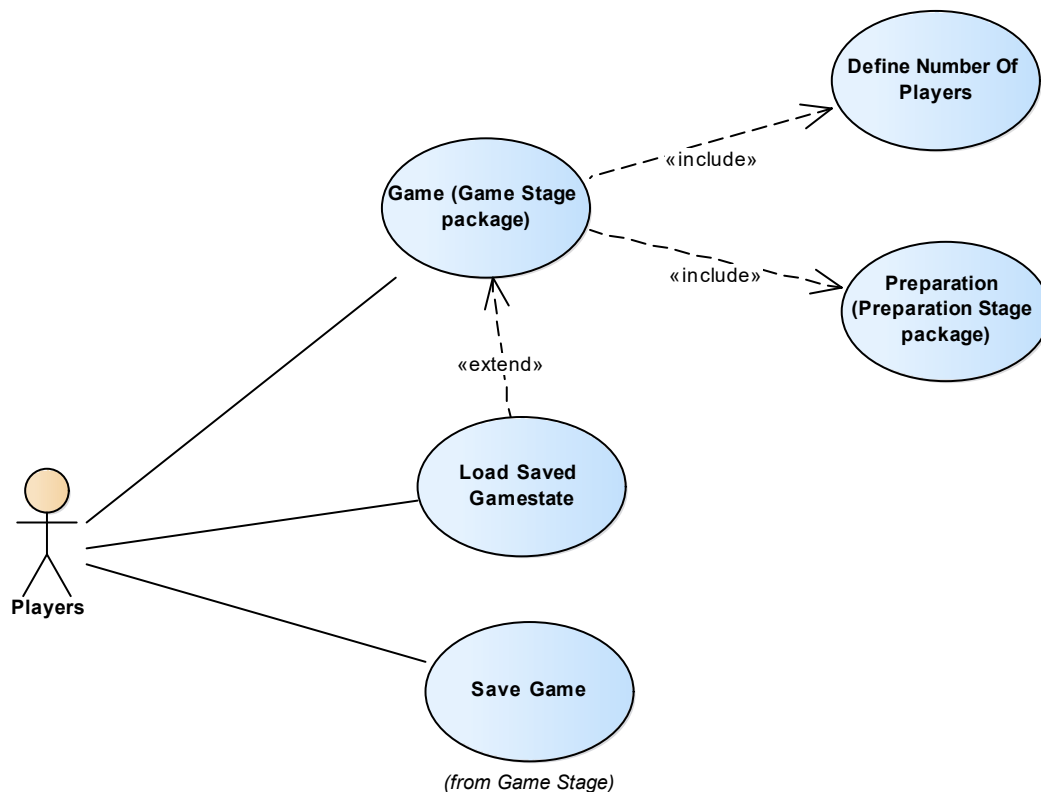
Figure 2: Application Start-up and Shutdown Use Cases

Define Number Of Players

UseCase in package 'Application Start-up, Run and Shutdown'

A jatek kezdete elott az elso jatekosnak meg kell hataroznia, hogy osszesen hany jatekosnal kezdodjon el a jatek.

CONNECTORS

CONNECTORS

↗. **Include** «include»   Source -> Destination
   From: Game (Game Stage package) : UseCase, Public
   To: Define Number Of Players : UseCase, Public

Game (Game Stage package)

UseCase in package 'Application Start-up, Run and Shutdown'

A jatekosok egymas utan kovetkeznek. Minden jatekos a sajat koreben eloszor elhelyezi az uj egysegeket, utana donthet, hogy akar-e tamadni. Korenek vegen atcsoportosithat egy teruletrol egysegeket egy masik teruletre.

<div align="right">

Game (Game Stage package)
Version 1.0 Phase 1.0 Proposed
Szabó Dávid created on 2017. 02. 22.. Last modified 2017. 03. 28.

</div>

CONNECTORS

↗. **Include** «include»   Source -> Destination
   From: Game (Game Stage package) : UseCase, Public
   To: Define Number Of Players : UseCase, Public

↗. **Include** «include»   Source -> Destination
   From: Game (Game Stage package) : UseCase, Public
   To: Preparation (Preparation Stage package) : UseCase, Public

---

↗. **Extend** «extend»   Source -> Destination
   From: Load Saved Gamestate : UseCase, Public
   To: Game (Game Stage package) : UseCase, Public

↗. **Association**   Unspecified
   From: Players : Actor, Public
   To: Game (Game Stage package) : UseCase, Public

Load Saved Gamestate

UseCase in package 'Application Start-up, Run and Shutdown'

<div align="right">

Load Saved Gamestate
Version 1.0 Phase 1.0 Proposed
Szabó Dávid created on 2017. 02. 22.. Last modified 2017. 03. 28.

</div>

CONNECTORS

↗. **Extend** «extend»   Source -> Destination
   From: Load Saved Gamestate : UseCase, Public
   To: Game (Game Stage package) : UseCase, Public

CONNECTORS

⬈. **Association**   Unspecified
   From: Players : Actor, Public
   To: Load Saved Gamestate : UseCase, Public

Preparation (Preparation Stage package)

UseCase in package 'Application Start-up, Run and Shutdown'

A jatekosok egymas utan elhelyezik a kezdeti egysegeket a teruleteken.

Preparation (Preparation Stage package)
Version 1.0 Phase 1.0 Proposed
Szabó Dávid created on 2017. 02. 22.. Last modified 2017. 03. 28.

CONNECTORS

⬈. **Include** «include»   Source -> Destination
   From: Game (Game Stage package) : UseCase, Public
   To: Preparation (Preparation Stage package) : UseCase, Public

Game Stage

Package in package 'Use Case Model'

Game Stage
Version 1.0 Phase 1.0 Proposed
Szabó Dávid created on 2017. 02. 22.. Last modified 2017. 03. 28.

Game Stage Use Cases diagram

Use Case diagram in package 'Game Stage'

Game Stage Use Cases
Version 1.0
Szabi created on 2017. 02. 21.. Last modified 2017. 03. 28.
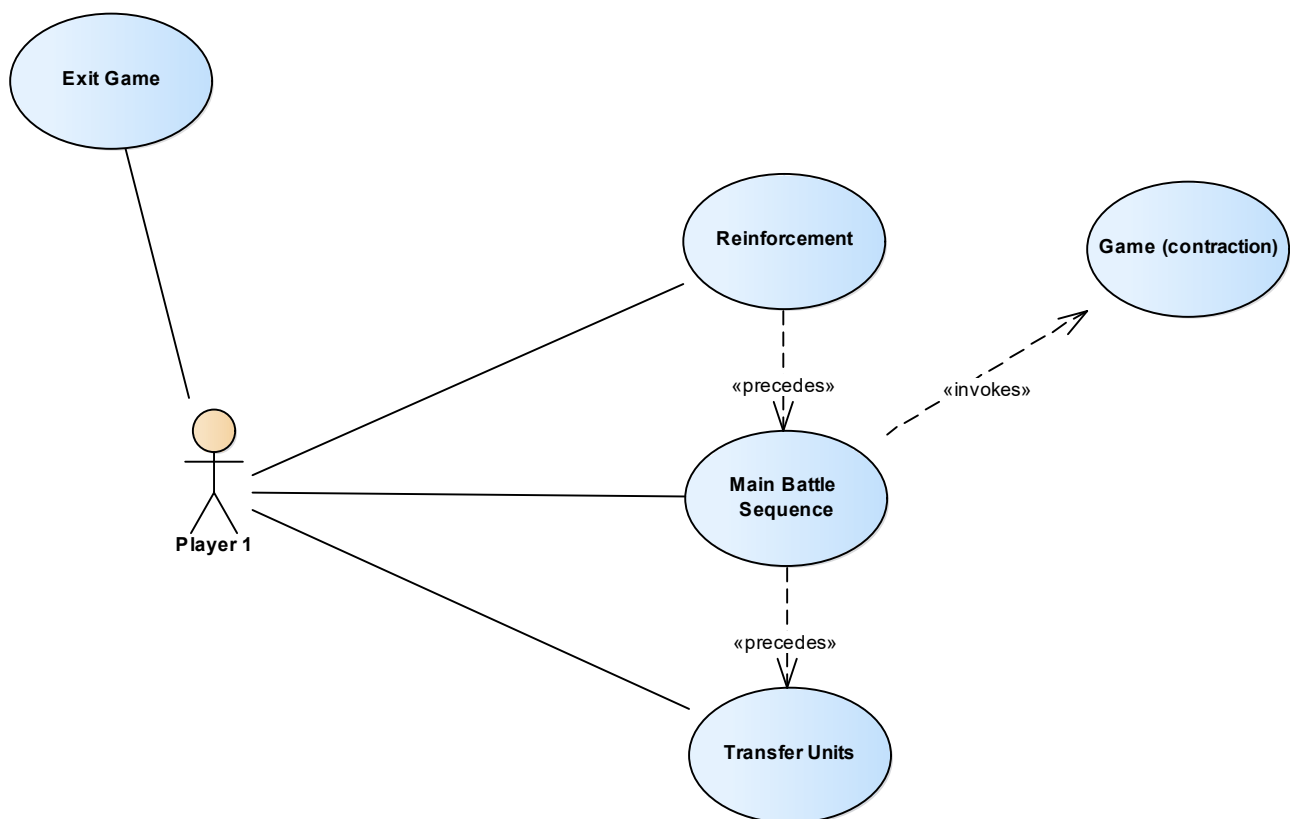
Figure 3: Game Stage Use Cases

3 Card Exchange/Using/TradeIn

UseCase in package 'Game Stage'

3 Card Exchange/Using/TradeIn
Version 1.0 Phase 1.0 Mandatory
Szabi created on 2017. 04. 04.. Last modified 2017. 04. 04.

Exit Game

UseCase in package 'Game Stage'

Aktualis jatekbol valo kilepes. Egy felugro pop-up megkerdezi a jatekost, hogy ki szeretne-e lepni. Amennyiben itt az

igen gombra kattint, a program megkerdezi, hogy menteni szeretne-e az aktualis jatekallast. Amennyiben az igen gombra kattint, abban az esetben a program elmenti a jatekallast. Amennyiben a fenti ket eset kozul barmelyiknel is "nem"-re kattint, abban az esetben a jatekbol kilep.

<div align="right">

Exit Game
Version 1.0 Phase 1.0 Proposed
Szabó Dávid created on 2017. 02. 22.. Last modified 2017. 03. 28.

</div>

CONNECTORS

↗ **Association**   Unspecified
From: Player 1 : Actor, Public
To: Exit Game : UseCase, Public

Game (contraction)

UseCase in package 'Game Stage'

A tamado 1, 2 vagy 3 egyseggel tamad, a vedekezo 1 vagy 2 egyseggel vedekezik.
Annyi kockaval dobunk ahany egyseg harcol. A ket fel legnagyobb dobasait paronkent osszehasonlitjuk.
Minden osszehasonlitasnal, ha a vedekezo legalabb akkorat dobott mint a tamado akkor a tamado veszit egy egyseget.
Ellenekzo esetben a vedekezo veszit egy egyseget.

<div align="right">

Game (contraction)
Version 1.0 Phase 1.0 Proposed
Szabi created on 2017. 02. 21.. Last modified 2017. 03. 28.

</div>

CONNECTORS

↗ **Dependency** «invokes»   Source -> Destination
From: Main Battle Sequence : UseCase, Public
To: Game (contraction) : UseCase, Public

Main Battle Sequence

UseCase in package 'Game Stage'

A jatekos a sajat teruleteirol szomszedos teruleteket tamadhat.
Megjeloli:
-Tamadast indito terulet
-Tamadott terulet
-Tamado egysegek szama (1, 2 vagy 3. Legalabb 1 egysegnek maradnia kell a tamado teruleten)

<div align="right">

Main Battle Sequence
Version 1.0 Phase 1.0 Proposed
Szabi created on 2017. 02. 21.. Last modified 2017. 03. 28.

</div>

CONNECTORS

↗ **Dependency** «precedes»   Source -> Destination
From: Main Battle Sequence : UseCase, Public
To: Transfer Units : UseCase, Public

CONNECTORS

   ↗ **Dependency** «invokes»   Source -> Destination
       From: Main Battle Sequence : UseCase, Public
       To: Game (contraction) : UseCase, Public

   ↗ **Dependency** «precedes»   Source -> Destination
       From: Reinforcement : UseCase, Public
       To: Main Battle Sequence : UseCase, Public

   ↗ **UseCaseLink**   Source -> Destination
       From: Player 1 : Actor, Public
       To: Main Battle Sequence : UseCase, Public

Reinforcement

UseCase in package 'Game Stage'

A jatekos a fordulo kezdeten erositest kerhet
-A birtokolt terulet harmada alapjan
-Minimum 3
-Extra szabalyok..

<div align="right">

Reinforcement
Version 1.0 Phase 1.0 Mandatory
Szabi created on 2017. 02. 21.. Last modified 2017. 03. 28.

</div>

CONNECTORS

   ↗ **Dependency** «precedes»   Source -> Destination
       From: Reinforcement : UseCase, Public
       To: Main Battle Sequence : UseCase, Public

   ↗ **UseCaseLink**   Source -> Destination
       From: Player 1 : Actor, Public
       To: Reinforcement : UseCase, Public

Save Game

UseCase in package 'Game Stage'

Aktualis jatekallas mentese

<div align="right">

Save Game
Version 1.0 Phase 1.0 Proposed
Oliver created on 2017. 02. 23.. Last modified 2017. 03. 28.

</div>

CONNECTORS

CONNECTORS

↗ **Association**   Unspecified
From: Players : Actor, Public
To: Save Game : UseCase, Public

Transfer Units

UseCase in package 'Game Stage'

Egy teruletrol szomszedos teruletekre rakhatja az egysegeket.
Legalabb egy egysegnek lennie kell minden teruleten.

Transfer Units
Version 1.0 Phase 1.0 Mandatory
Szabi created on 2017. 02. 21.. Last modified 2017. 03. 28.

CONNECTORS

↗ **Dependency** «precedes»   Source -> Destination
From: Main Battle Sequence : UseCase, Public
To: Transfer Units : UseCase, Public

↗ **UseCaseLink**   Source -> Destination
From: Player 1 : Actor, Public
To: Transfer Units : UseCase, Public

Preparation Stage

Package in package 'Use Case Model'

Preparation Stage
Version 1.0 Phase 1.0 Proposed
Szabó Dávid created on 2017. 02. 22.. Last modified 2017. 03. 28.

Preparation Stage Use Cases diagram

Use Case diagram in package 'Preparation Stage'

Preparation Stage Use Cases
Version 1.0
Szabó Dávid created on 2017. 02. 22.. Last modified 2017. 03. 28.

Figure 4: Preparation Stage Use Cases

Define The Starting Player

UseCase in package 'Preparation Stage'

A rendszer kivalasztja, hogy melyik jatekos kezdheti elhelyezni az egysegeket.

Define The Starting Player
Version 1.0 Phase 1.0 Proposed
Szabó Dávid created on 2017. 02. 22.. Last modified 2017. 03. 28.

CONNECTORS
↗ **Dependency** «precedes»   Source -> Destination From: Define The Starting Player : UseCase, Public To: Preparation Stage (contraction) : UseCase, Public

Preparation Stage (contraction)

UseCase in package 'Preparation Stage'

A jatekosok egyesevel elhelyezik az osszes kezdeti egyseget a szabad teruletekre.
Eloszor egy teruletre csak egy egyseget lehet tenni. Ha mar nincs tobb szabad terulet akkor minden jatekos csak a sajat
teruleteire tehet egyseget.

Preparation Stage (contraction)
Version 1.0 Phase 1.0 Proposed
Szabó Dávid created on 2017. 02. 22.. Last modified 2017. 03. 28.

CONNECTORS

CONNECTORS

Association   Unspecified
From: Player : Actor, Public
To: Preparation Stage (contraction) : UseCase, Public

Dependency «precedes»   Source -> Destination
From: Define The Starting Player : UseCase, Public
To: Preparation Stage (contraction) : UseCase, Public

Primary Use Cases

Package in package 'Use Case Model'

Primary Use Cases
Version 1.0 Phase 1.0 Proposed
Szabi created on 2017. 02. 21.. Last modified 2017. 02. 21.

Primary Use Cases diagram

Use Case diagram in package 'Primary Use Cases'

Primary Use Cases
Version 1.0
Oliver created on 2017. 02. 21.. Last modified 2017. 02. 21.
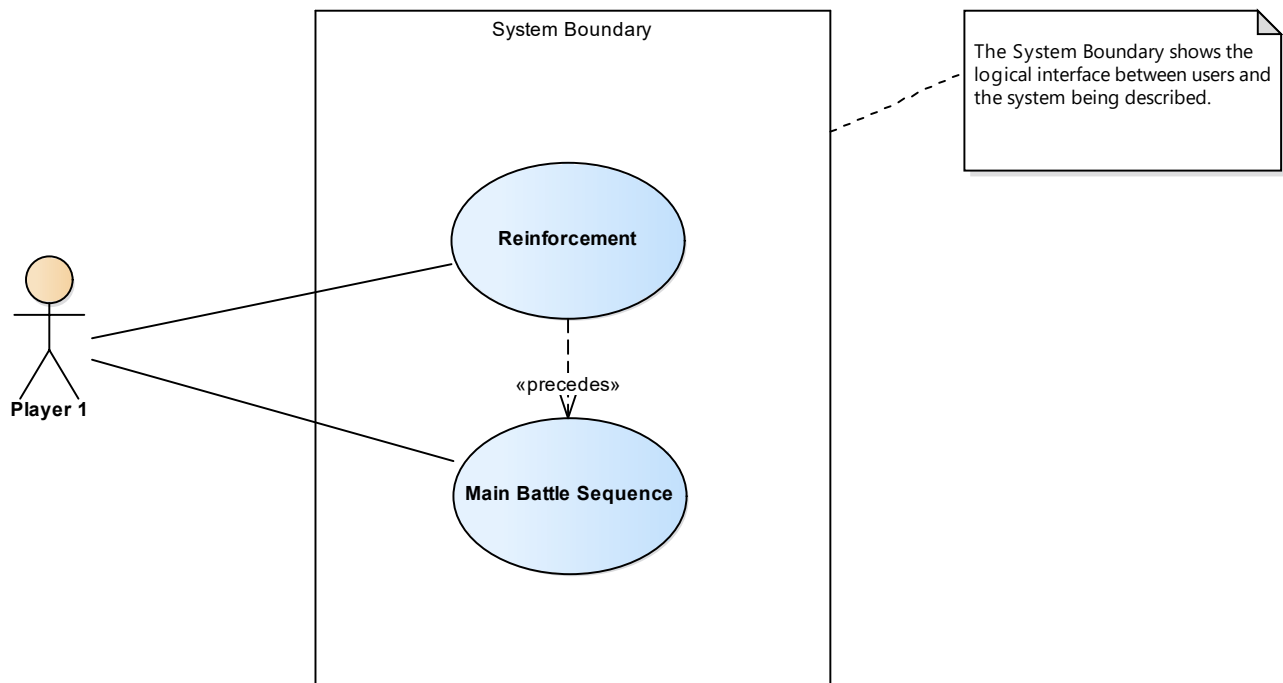
Figure 5: Primary Use Cases

Dynamic View

Package in package 'Model'

Dynamic View
Version   Phase 1.0   Proposed
Szabi created on 2017. 03. 07.. Last modified 2017. 03. 07.

Application Start-up, Run and Shutdown

Package in package 'Dynamic View'

Application Start-up, Run and Shutdown
Version 1.0   Phase 1.0   Proposed
Oliver created on 2017. 03. 28.. Last modified 2017. 03. 28.

Define number of players diagram

Interaction diagram in package 'Application Start-up, Run and Shutdown'

Define number of players
Version 1.0
fodorad created on 2017. 03. 22.. Last modified 2017. 09. 21.
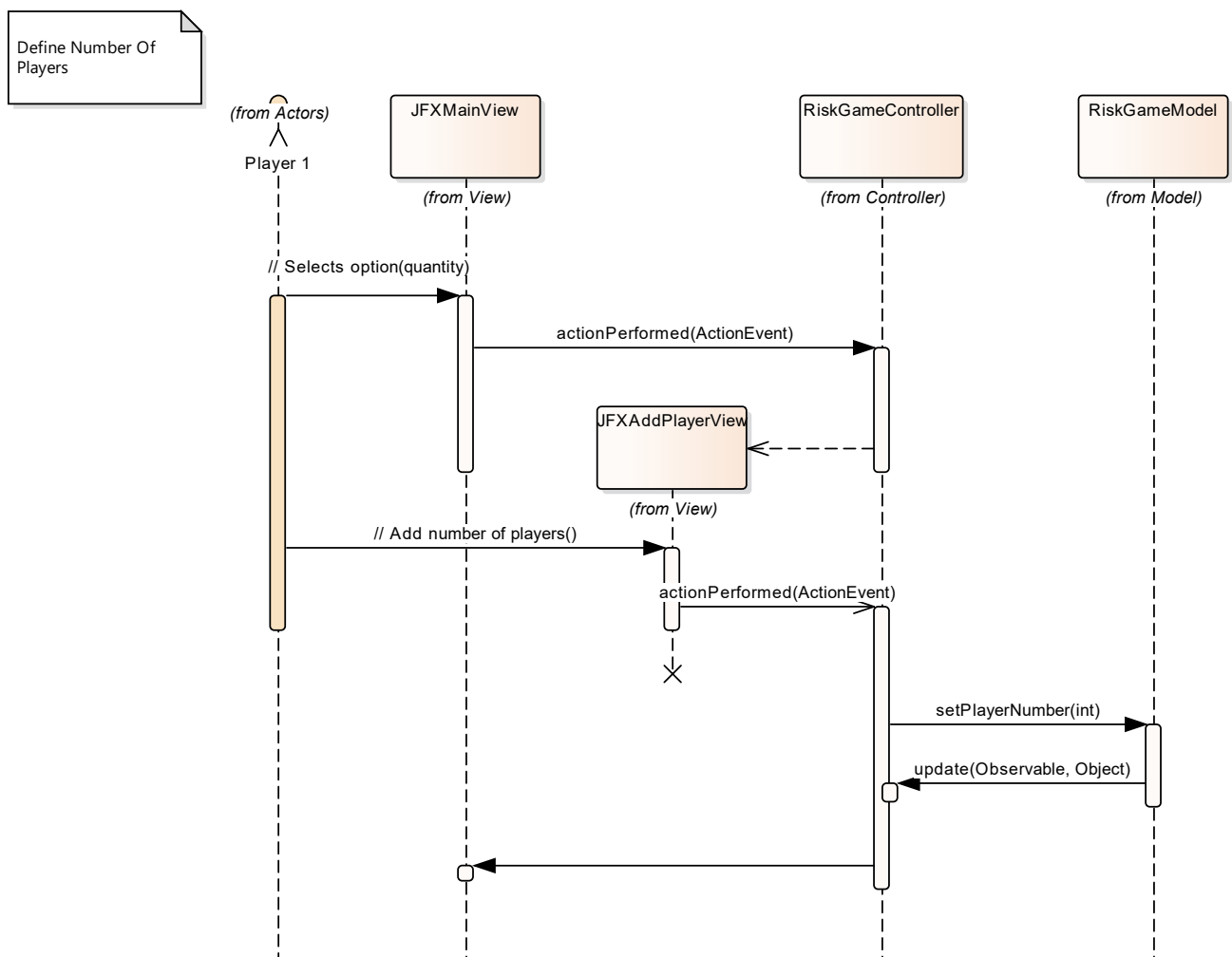
Figure 6: Define number of players

Game diagram

Interaction diagram in package 'Application Start-up, Run and Shutdown'

Game diagram

Interaction diagram in package 'Application Start-up, Run and Shutdown'

(from Actors)

Player 1

ref
Define number of players

ref
Define the starting player

alt want_load_game

[Load_Game == True]

ref
Load Game

[else]

ref
Preparation stage

loop Main Game

ref
Reinforcement

ref
Main Battle Sequence

ref
Transfer Units

ref
Exchange Cards

alt want_save_game

[Save_game == True]

ref
Save Game

ref
Exit Game
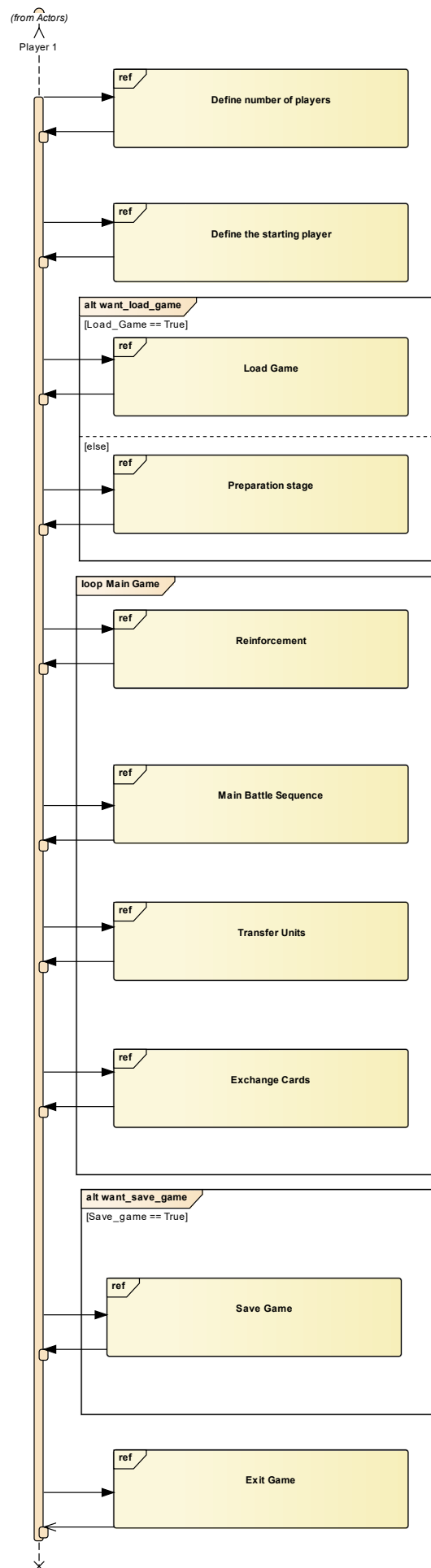
Figure 7: Game

Load Game diagram

Interaction diagram in package 'Application Start-up, Run and Shutdown'
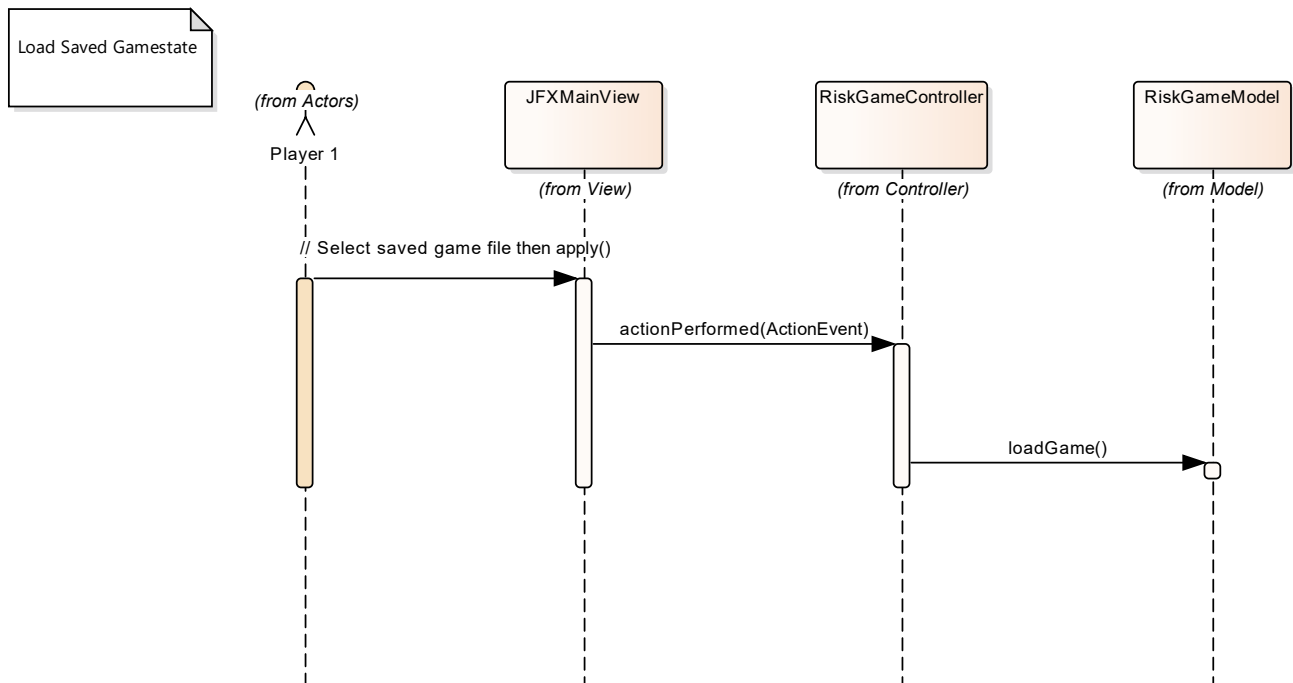
Figure 8: Load Game

Game Stage

Package in package 'Dynamic View'

<div align="right">

Game Stage
Version 1.0 Phase 1.0 Proposed
Szabó Dávid created on 2017. 03. 22.. Last modified 2017. 03. 28.

</div>

Check If Captured And Conquer diagram

Interaction diagram in package 'Game Stage'

<div align="right">

Check If Captured And Conquer
Version 1.0
Szabi created on 2017. 03. 07.. Last modified 2017. 03. 28.

</div>
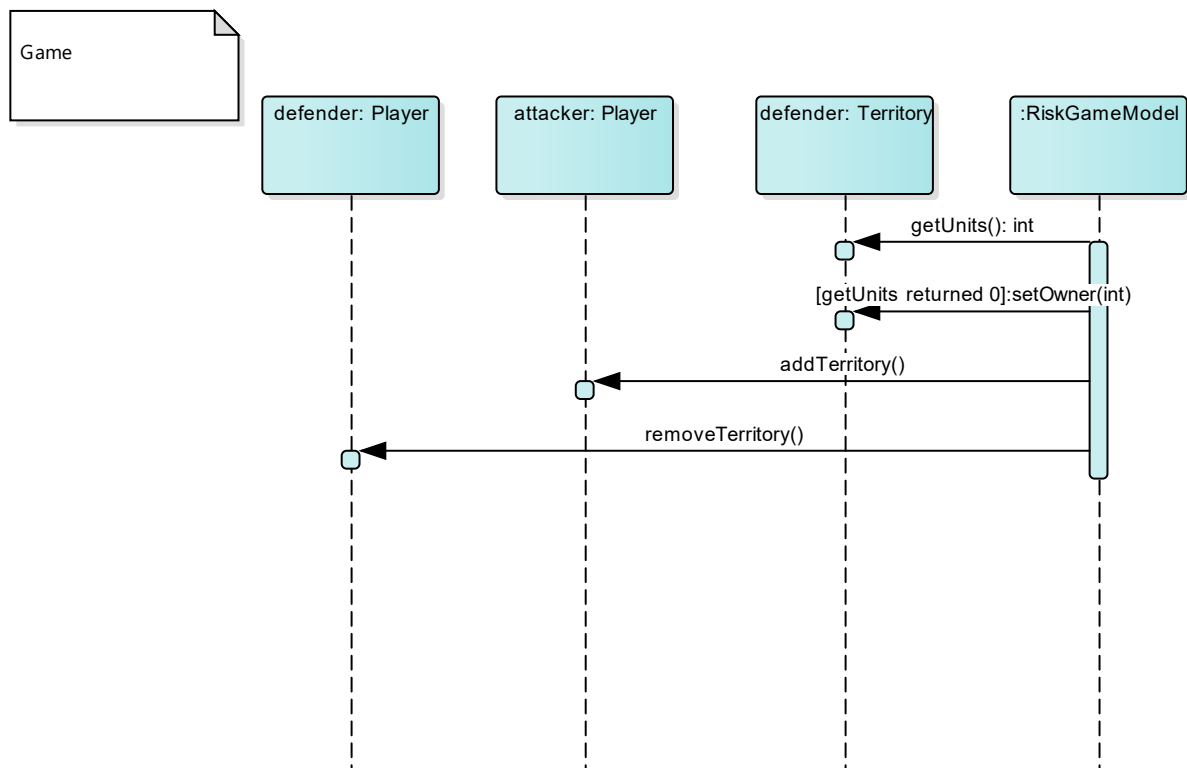
Figure 9: Check If Captured And Conquer

Exchange Cards diagram

Interaction diagram in package 'Game Stage'

<div align="right">

Exchange Cards
Version 1.0
Szabi created on 2017. 03. 29.. Last modified 2017. 09. 21.

</div>

3 Card
Exchange/Using/TradeIn

(from Actors)
Player 1

JFXMainView
(from View)

RiskGameController
(from Controller)

RiskGameModel
(from Model)

Deck
(from Model)

Player1: Player

//end turn()

actionPerformed(ActionEvent)

endTurn()

Draw(): Card

putcard(Card)

exchangeCardsIfPossible(): byte

addReinforcements(byte)

update(Observable, Object)

nextTurn()
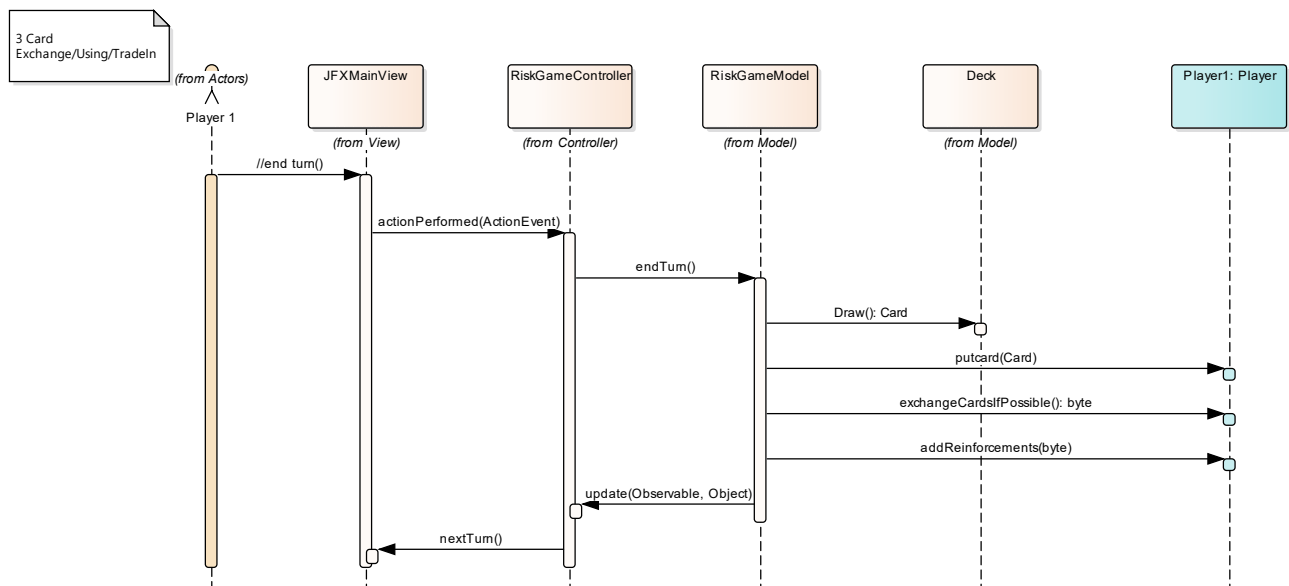
Figure 10: Exchange Cards

Exit Game diagram

Interaction diagram in package 'Game Stage'

Exit Game
Version 1.0
fodorad created on 2017. 03. 22.. Last modified 2017. 09. 21.
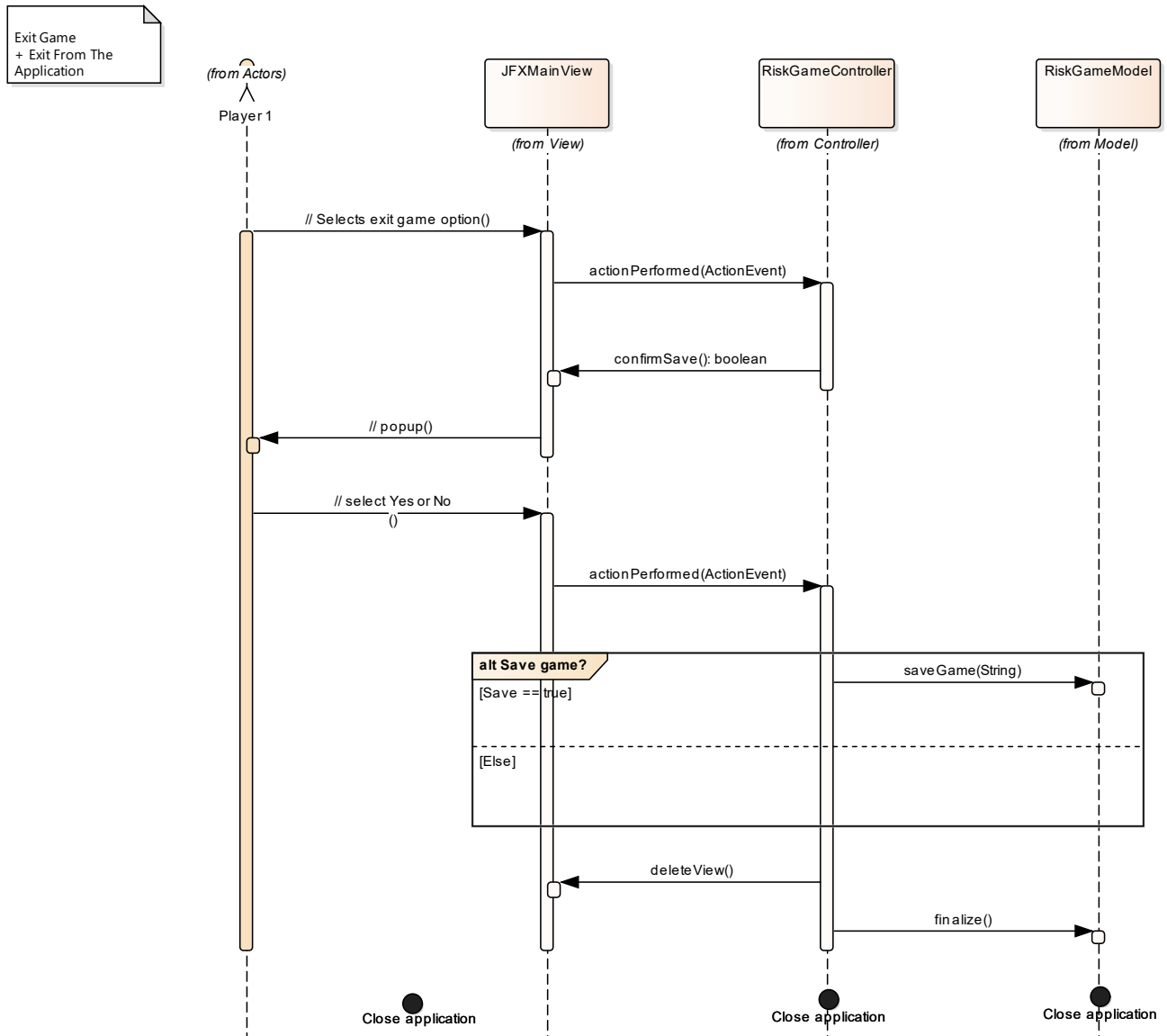
Figure 11: Exit Game

Main Battle Sequence diagram

Interaction diagram in package 'Game Stage'

Main Battle Sequence
Version 1.0
Szabi created on 2017. 03. 07.. Last modified 2017. 09. 21.
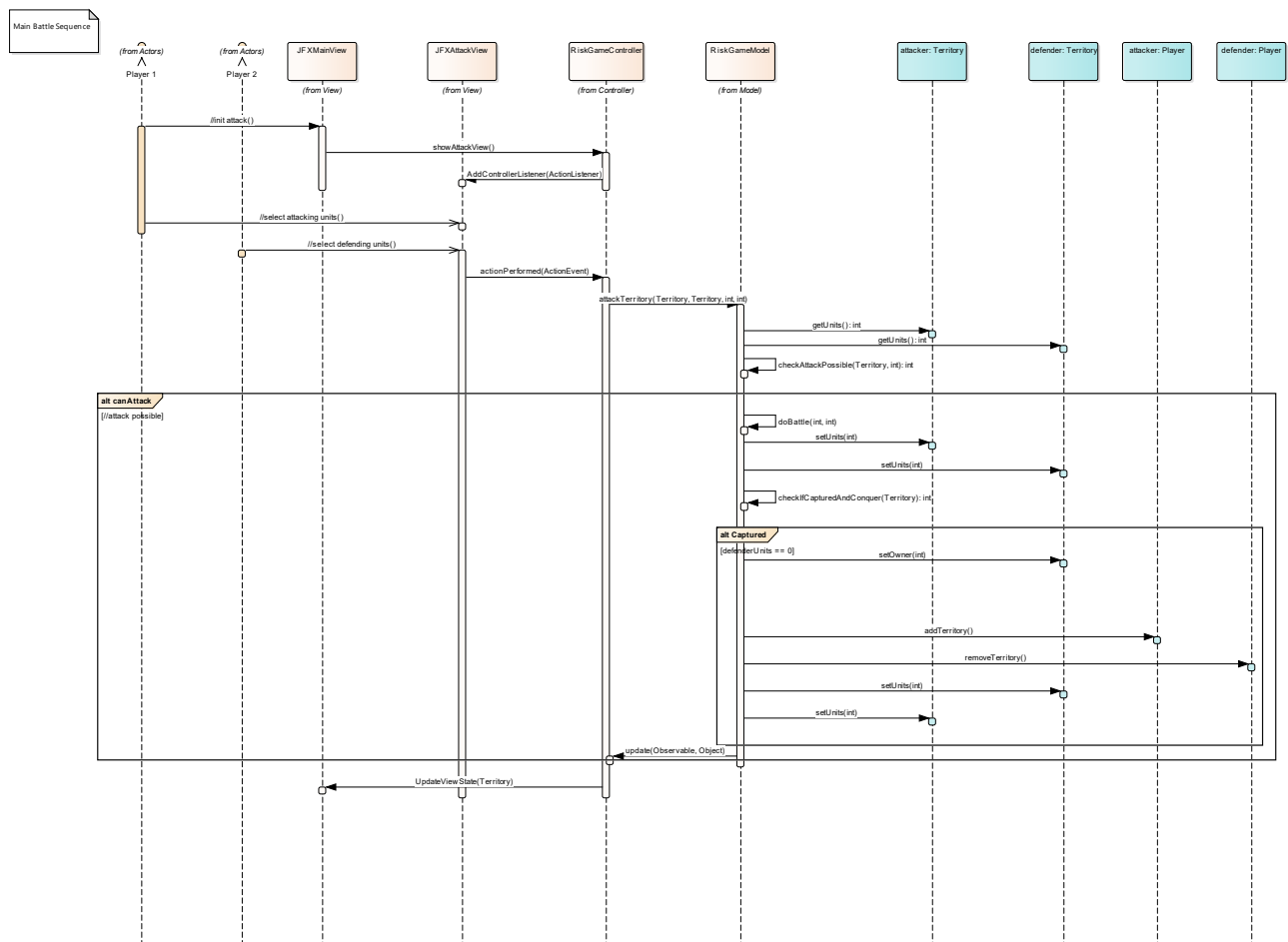
Figure 12: Main Battle Sequence

Reinforcement diagram

Interaction diagram in package 'Game Stage'
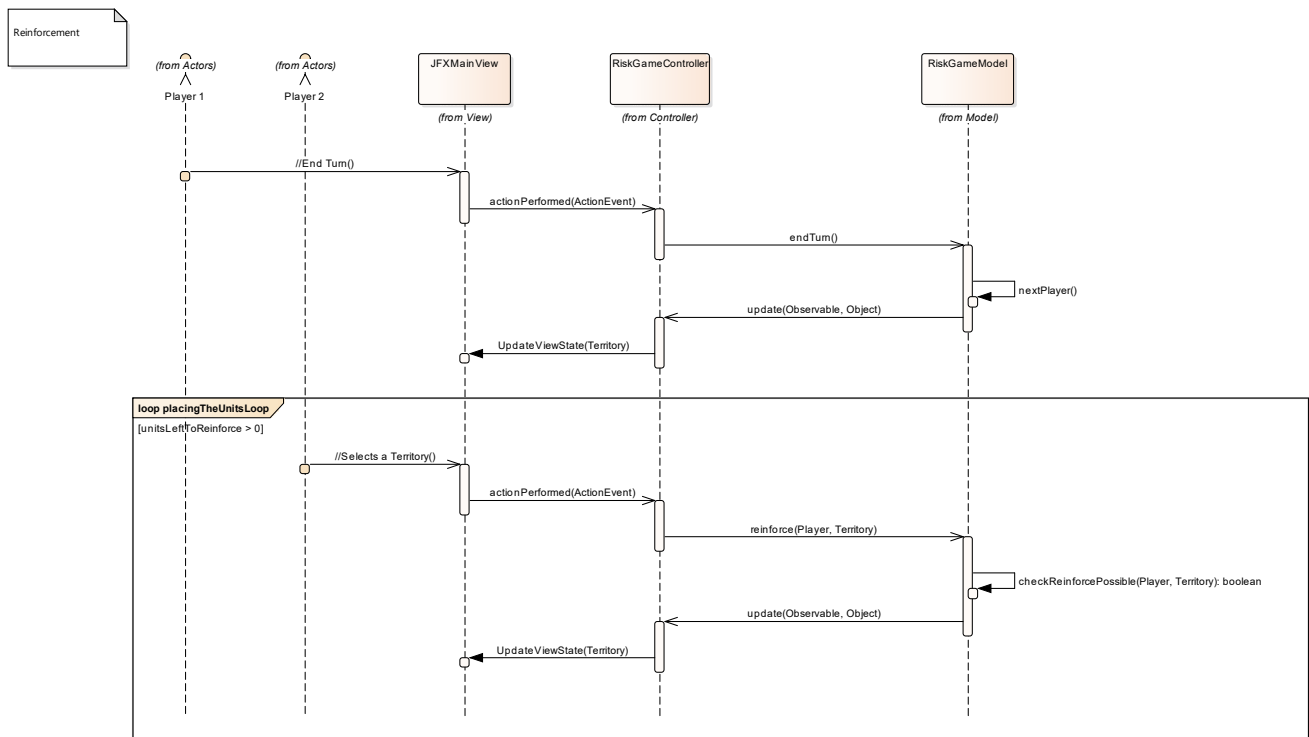
Figure 13: Reinforcement

Save Game diagram

Interaction diagram in package 'Game Stage'

Save Game
Version 1.0
Szabó Dávid created on 2017. 03. 22.. Last modified 2017. 09. 21.
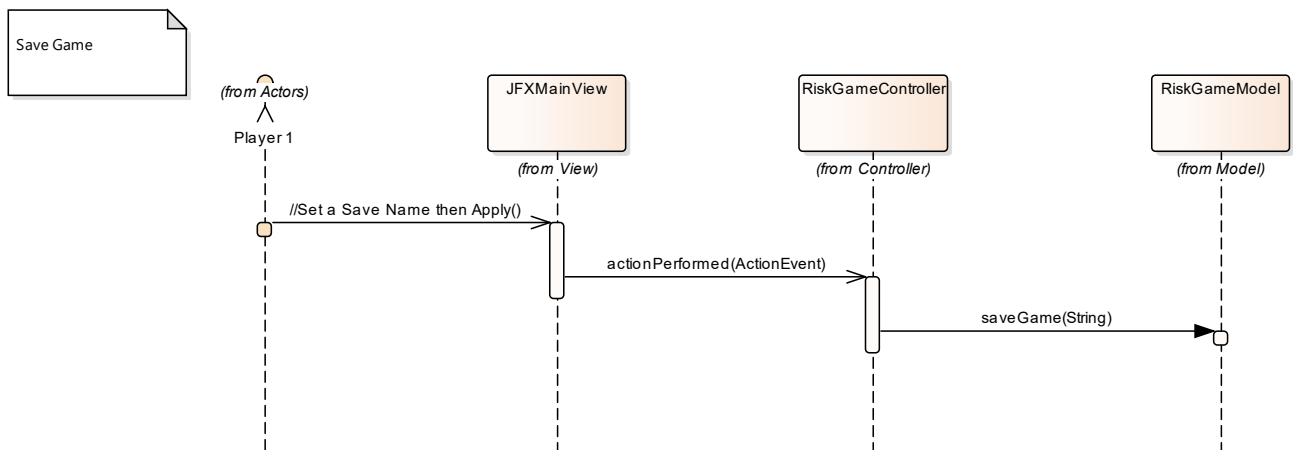
Figure 14: Save Game

Transfer Units diagram

Interaction diagram in package 'Game Stage'

Transfer Units
Version 1.0
Szabó Dávid created on 2017. 03. 22.. Last modified 2017. 09. 21.

Transfer Units

(from Actors)
Player 1

JFXMainView
(from View)

RiskGameController
(from Controller)

RiskGameModel
(from Model)

//Selects 2 Territories()

actionPerformed(ActionEvent)

checkTransferPossible(Territory, Territory): boolean

[Transfer Possible]:
showTransferView()

[Transfer Possible]:
AddControllerListener
(ActionListener)

JFXTransferView
(from View)

[Transfer Possible]:
UpdateViewState
(Territory, Territory)

//Selects Units then Apply()

actionPerformed(ActionEvent)

checkTransferPossible(Territory, Territory, int): boolean

[Transfer Possible]:transfer
(Territory, Territory, int)

update(Observable, Object)

UpdateViewState(Territory)

Figure 15: Transfer Units

Preparation Stage

Package in package 'Dynamic View'

<div align="right">

Preparation Stage
Version 1.0 Phase 1.0 Proposed
Oliver created on 2017. 03. 22.. Last modified 2017. 03. 28.

</div>

Define the starting player diagram

Interaction diagram in package 'Preparation Stage'

<div align="right">

Define the starting player
Version 1.0
Oliver created on 2017. 03. 22.. Last modified 2017. 04. 19.

</div>
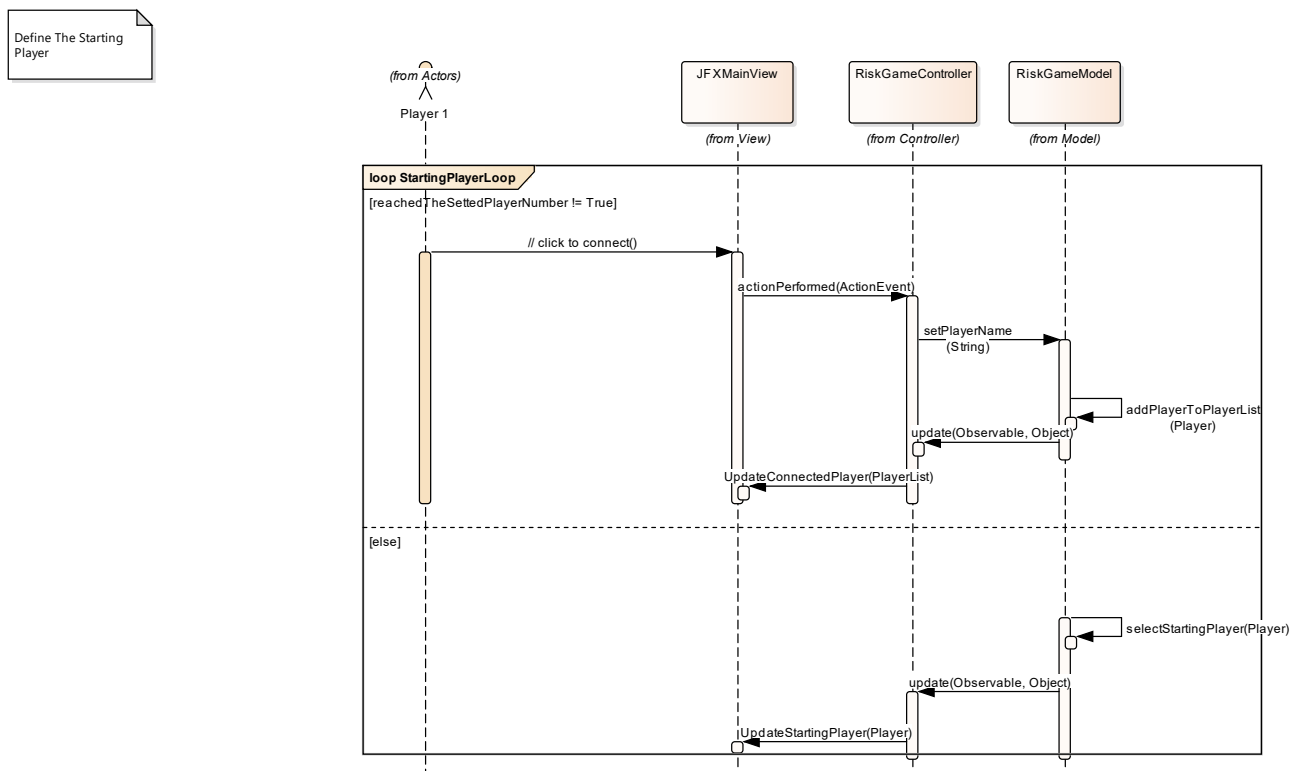
Figure 16: Define the starting player

Preparation stage diagram

Interaction diagram in package 'Preparation Stage'

<div align="right">

Preparation stage
Version 1.0
fodorad created on 2017. 03. 22.. Last modified 2017. 04. 19.

</div>
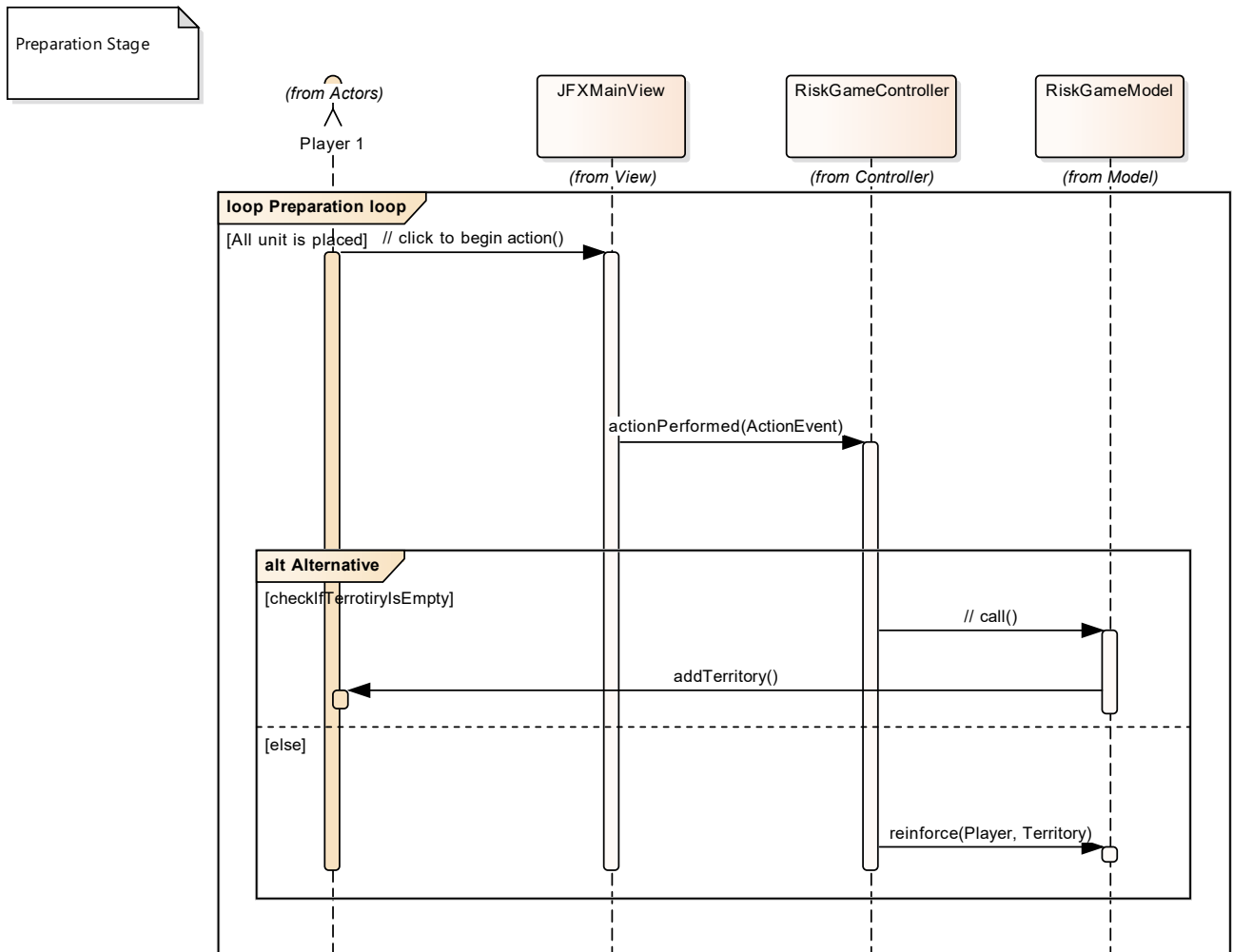
Preparation Stage

Figure 17: Preparation stage

Class Model

Package in package 'Model'

<div align="right">

Class Model
Version   Phase 1.0 Proposed
Szabi created on 2017. 02. 21.. Last modified 2017. 02. 21.

</div>

Class Model diagram

Class diagram in package 'Class Model'

<div align="right">

Class Model
Version 1.0
Oliver created on 2017. 02. 21.. Last modified 2017. 03. 28.

</div>

The Class Model is a logical model of the software system under construction. Classes generally have a direct relationship to source code or other software artifacts that can be grouped together into executable components.

The System package contains the classes and artifacts which are being built or designed as part of the current model. The Frameworks package generally contains classes and components that have been designed and built earlier and are being reused as part of the current project.

View
- + CountryPath
- ~ ImageViewPane
- + JFXAddPlayerView
- + JFXAttackView
- + JFXMainView
- + JFXRiskCardView
- + JFXTransferView
- + *JFXViewBase*
- + WorldBuilder
- + Country

Controller
- + RiskGameController

Model
- + AttackResult
- + Card
- + Deck
- + Map
- + Player
- + ResourceManager
- + RiskGameModel
- + SaveData
- + Territory
- + Color
- + Phase
- + Unit
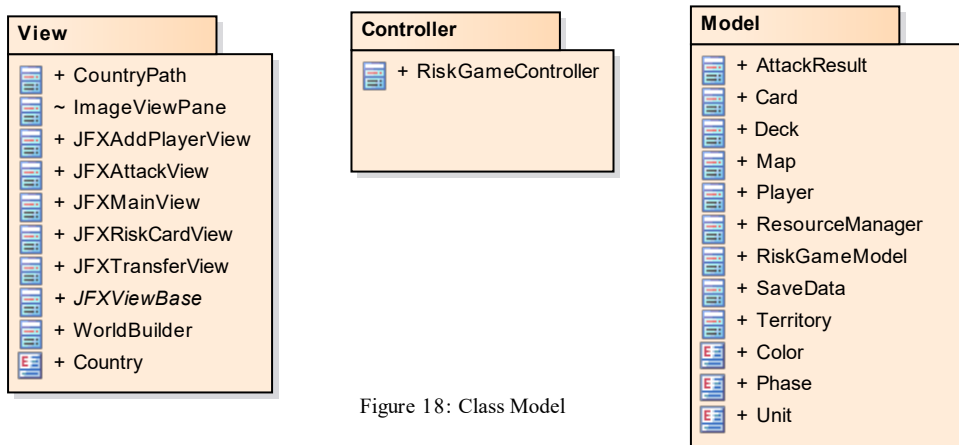
Figure 18: Class Model

Application

Package in package 'Class Model'

<div align="right">

Application
Version 1.0 Phase 1.0 Proposed
Oliver created on 2017. 05. 04.. Last modified 2017. 05. 04.

</div>

Application diagram

Class diagram in package 'Application'

<div align="right">

Application
Version 1.0
Oliver created on 2017. 05. 04.. Last modified 2017. 09. 21.

</div>

```
            Application
               App
    ─────────────────────────
    +    main(String[]): void
    +    start(Stage): void
```

Figure 19: Application

Controller

Package in package 'Class Model'

Controller
Version 1.0 Phase 1.0 Proposed
Szabó Dávid created on 2017. 03. 01.. Last modified 2017. 03. 01.

Controller diagram

Class diagram in package 'Controller'

Controller
Version 1.0
Szabó Dávid created on 2017. 03. 01.. Last modified 2017. 11. 22.

Controller

Package in package 'Class Model'

Model::RiskGameModel

-model

View::JFXTransferView

- btnAccept: Button
- fromID: int
- lblFromCountryName: Label
- lblFromQuantity: Label
- lblPlayerName: Label
- lblSelectedQuantity: Label
- lblToCountryName: Label
- lblToQuantity: Label
- sldSoilderQuantity: Slider
- toID: int

+ finalize(): void
+ initialize(URL, ResourceBundle): void
+ JFXTransferView()
+ UpdateViewState(String, Territory, Territory): void

View::JFXAddPlayerView

- btnAddPlayer: Button
- btnDeletePlayer4: Button
- btnDeletePlayer5: Button
- btnDeletePlayer6: Button
- btnReady: Button
- lblPlayer4: Label
- lblPlayer5: Label
- lblPlayer6: Label
- ~ stage: Stage
- tfList: ArrayList<TextField>
- tfPlayer1: TextField
- tfPlayer2: TextField
- tfPlayer3: TextField
- tfPlayer4: TextField
- tfPlayer5: TextField
- tfPlayer6: TextField

addPlayerAction(ActionEvent): void
finalize(): void
+ initialize(URL, ResourceBundle): void
+ JFXAddPlayerView(Stage)
# readyOnAction(ActionEvent): void
# removePlayer4Action(ActionEvent): void
# removePlayer5Action(ActionEvent): void
# removePlayer6Action(ActionEvent): void
+ UpdateViewState(String): void

java.util.Observable
RiskGameController

- addPlayerView: JFXAddPlayerView
- attackResultView: JFXAttackResultView
- attackView: JFXAttackView
- cardGroup: Group
- cardScene: Scene
- cardView: JFXCardView
- lastPhaseUpdate: Phase
- lastPlayerUpdate: int
- mainView: JFXMainView
- model: RiskGameModel
- popupStage: Stage
- preStage: Stage {readOnly}
- previouslySelectedTerritory: int = -1
- primaryStage: Stage
- titledPaneStage: Stage
- transferView: JFXTransferView
- victoryStage: Stage
- victoryView: JFXVictoryView

+ addLog(String): void
+ attackAccepted(int, int, int, int): void
+ attackRetreat(): void
+ attackShowed(int, int): void
- closePopupWindow(): void
- convertToCountry(int): Country
- convertToTerritoryID(Country): int
+ countrySelected(Country): void
+ getPhase(): Phase
+ getPreviouslySelectedTerritory(): int
+ getTerritoryData(): List<Territory>
- isVictory(): boolean
+ loadGame(File): void
+ moveCardView(): void
+ nextPlayerSelected(): void
- reinforce(int, String): void
+ RiskGameController(Stage)
+ saveGame(File): void
+ setModel(RiskGameModel): void
+ showAddPlayerView(): void
- showAttackResultView(): void
- showAttackView(int, int): void
+ showCardView(): void
+ showMainView(): void
- showTransferView(int, int): void
- showVictoryView(): void
+ switchToMain(Map<Color, String>): void
+ transferAccepted(int, int, int): void
+ transferPhaseSelected(): void
- UpdateAllTerritoriesOnMap(): void
- UpdateCurrentDeck(): void
- UpdateCurrentPhase(): void
- UpdateCurrentPlayer(): void
- UpdateTerritoryOnMap(int): void

View::JFXRiskCardView

+ finalize(): void
+ JFXRiskCardView()

Initializable
View::JFXViewBase

controller: RiskGameController
root: Parent

+ AddControllerListener(RiskGameController): void
+ getRoot(): Parent
+ initialize(URL, ResourceBundle): void
+ JFXViewBase()
# LoadFXML(String): void

#controller

-transferView

-addPlayerView

View::JFXAttackView

- attackerID: int
- attackerUnits: int
- barUnits: ProgressBar
- btnAttack: Button
- btnRetreat: Button
- defenderID: int
- defenderUnits: int
- lblAttackerAlive: Label
- lblAttackerName: Label
- lblCountryAttacker: Label
- lblCountryDefender: Label
- lblDefenderAlive: Label
- lblDefenderName: Label
- spinneratt: Spinner
- spinnerdef: Spinner

+ finalize(): void
+ initialize(URL, ResourceBundle): void
+ JFXAttackView()
+ UpdateViewState(AttackResult): void
+ UpdateViewState(String, String, Territory, Territory): void

-attackView

View::JFXMainView

- BASE_HEIGHT: double = 600 {readOnly}
- BASE_WIDTH: double = 815 {readOnly}
- borderPane: BorderPane
- btnBattlePhase: Button
- btnControlList: ArrayList<Button>
- btnLoad: Button
- btnLog: Button
- btnNextPlayer: Button
- btnReinforcePhase: Button
- btnSave: Button
- btnSettings: Button
- btnTransferPhase: Button
- centerStackPane: StackPane
- ~ dimLog: Dimension
- imageView: ImageView
- ~ isLogOpened: boolean
- ~ isSLOpened: boolean
- lblPlayerName1: Label
- lblPlayerName2: Label
- lblPlayerName3: Label
- lblPlayerName4: Label
- lblPlayerName5: Label
- lblPlayerName6: Label
- lblPlayerNameList: ArrayList<Label>
- pain: Pane
- panePlayerColor1: Pane
- panePlayerColor2: Pane
- panePlayerColor3: Pane
- panePlayerColor4: Pane
- panePlayerColor5: Pane
- panePlayerColor6: Pane
- panePlayerColorList: ArrayList<Pane>
- stage: Stage
- taLog: TextArea
- world: World

+ AddControllerListener(RiskGameController): void
+ appendLog(String): void
+ closeLog(): void
+ finalize(): void
+ fit(double, double): void
+ getRoot(): Parent
+ getWorld(): World
+ handleLogButton(ActionEvent): void
- hangleLoadButton(ActionEvent): void
- hangleSaveButton(ActionEvent): void
- hangleSettingsButton(ActionEvent): void
+ initialize(URL, ResourceBundle): void
+ isLogOpened(): boolean
+ isSLOpened(): boolean
+ JFXMainView(Stage)
+ openLog(): void
+ UpdateConnectedPlayers(List<String>): void
+ UpdateCurrentDeck(List<Integer>): void
+ UpdateCurrentPhase(Phase): void
+ UpdateCurrentPlayer(int): void
+ updateIcons(int, int): void
+ UpdateViewState(Territory): void
+ UpdateViewState(List<Territory>): void

-mainView

Figure 20: Controller

View

Package in package 'Class Model'

<div align="right">

View
Version 1.0 Phase 1.0 Proposed
Szabó Dávid created on 2017. 03. 01.. Last modified 2017. 03. 01.

</div>

View diagram

Class diagram in package 'View'

<div align="right">

View
Version 1.0
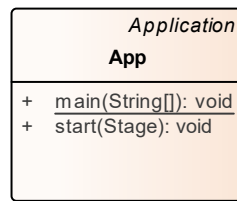Szabó Dávid created on 2017. 03. 01.. Last modified 2017. 09. 21.
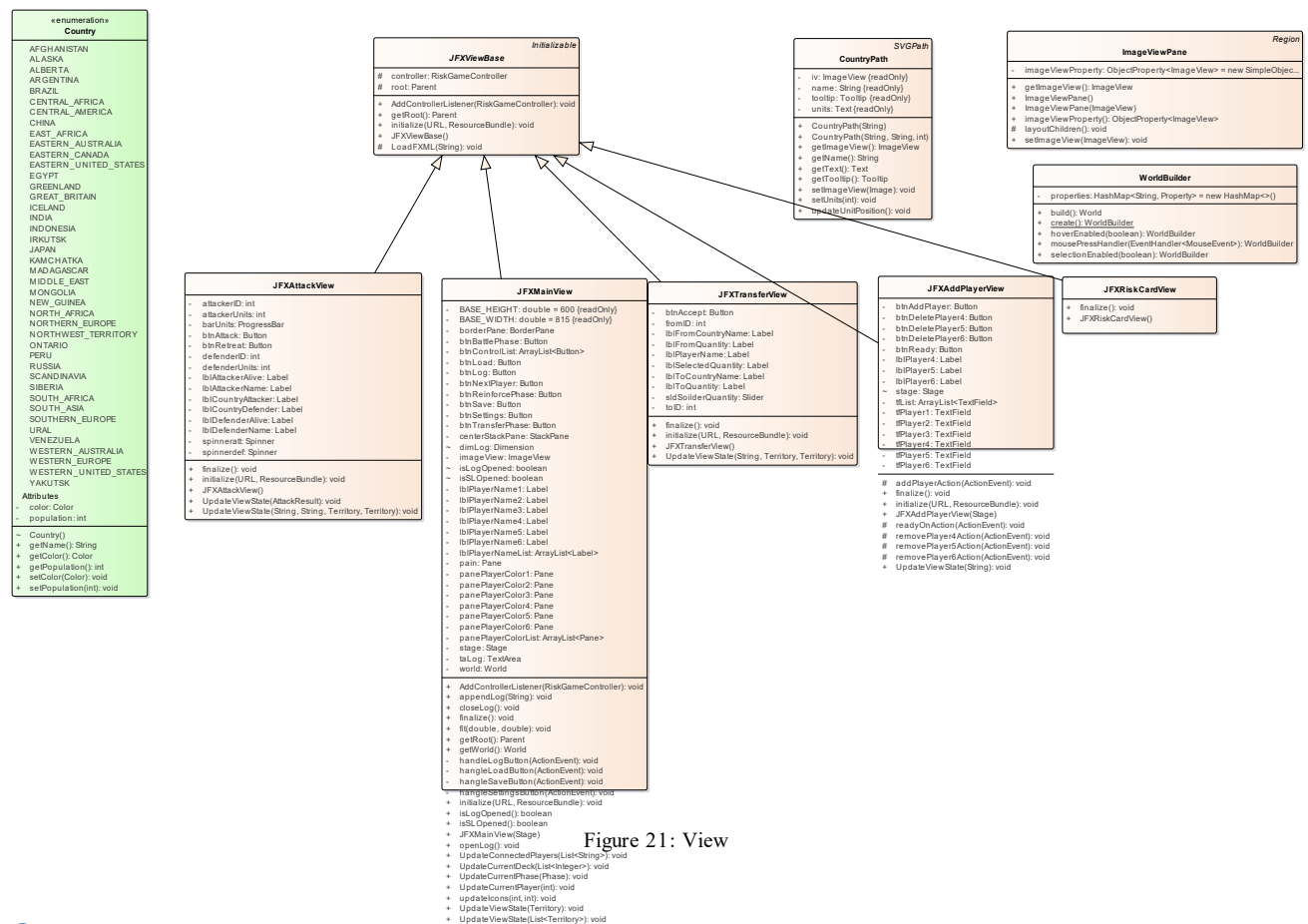
</div>

Figure 21: View

Country

Enumeration in package 'View'

Enumerator for territories' view objects

<div align="right">

Country
Version 1.0 Phase 1.0 Proposed
fodorad created on 2017. 09. 21.. Last modified 2017. 11. 22.

</div>

Model

Package in package 'Class Model'

<div align="right">

Model
Version 1.0 Phase 1.0 Proposed
Szabi created on 2017. 02. 21.. Last modified 2017. 03. 01.

</div>

Model diagram

Class diagram in package 'Model'

<div align="right">

Model
Version 1.0
Szabi created on 2017. 02. 21.. Last modified 2017. 09. 21.

</div>

Figure 22: Model

Color

Enumeration in package 'Model'

A szin adatokat adja meg.

<div align="right">

Color
Version 1.0 Phase 1.0 Proposed
Szabi created on 2017. 02. 28.. Last modified 2017. 11. 22.

</div>

Phase

Enumeration in package 'Model'

Phase
Version 1.0 Phase 1.0 Proposed
Oliver created on 2017. 09. 21.. Last modified 2017. 11. 22.

Unit

Enumeration in package 'Model'

Az egyseg adatait adja meg.

Unit
Version 1.0 Phase 1.0 Proposed
Szabi created on 2017. 02. 28.. Last modified 2017. 11. 22.

User Interface Model

Package in package 'Model'

User Interface Model diagram

Custom diagram in package 'User Interface Model'
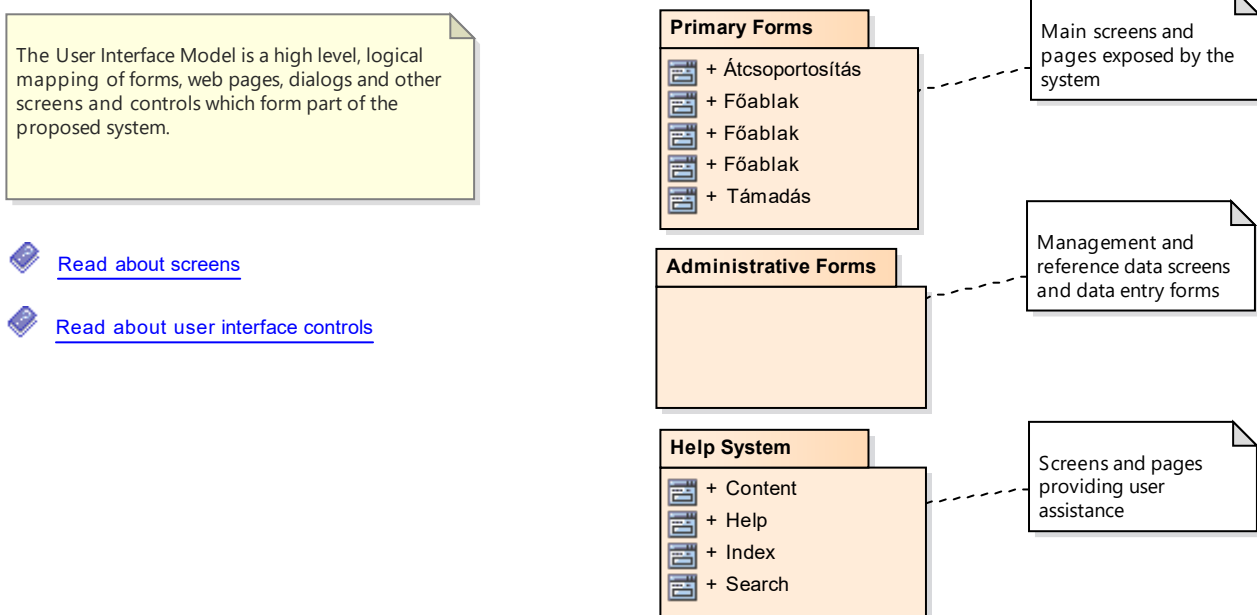
Figure 23: User Interface Model

Primary Forms

Package in package 'User Interface Model'

Primary Forms diagram

Custom diagram in package 'Primary Forms'

This package contains the primary
user interface screens. These are the
main points of interaction for system
users.

Főablak

Soron következő játékosok

1. játékos   2. játékos   3. játékos

Adatlap

egységek száma

Következő erősítés

Le nem helyezett
egységek száma

Világtérkép

4. játékos   5. játékos

Főablak

Soron következő játékosok

1. játékos   2. játékos   3. játékos

Napló

4. játékos   5. játékos

«abstraction»
«instantiate»

Főablak

Soron következő játékosok

1. Játékos   2. játékos   3. játékos

Világtérkép

**Bónusz
egységek
számlálója**

Napló

Erősítés | Támadás | Átcsoportosítás

4. játékos   Információs panel   Számláló   5. játékos

«flow»

Átcsoportosítás

Játékos neve

Honnan   **kijelölés**   Hová

darab   darab

Elfogad   Visszavonás

«navigate»

«navigate»

Támadás

Támadó játékos   Védekező játékos

**Egységek
képei**   **Támadó
dobókockái**   **Védekező
dobókockái**   Egységek képei

Életben maradt
egységek száma   **Vizuális élmény**   Életben maradt
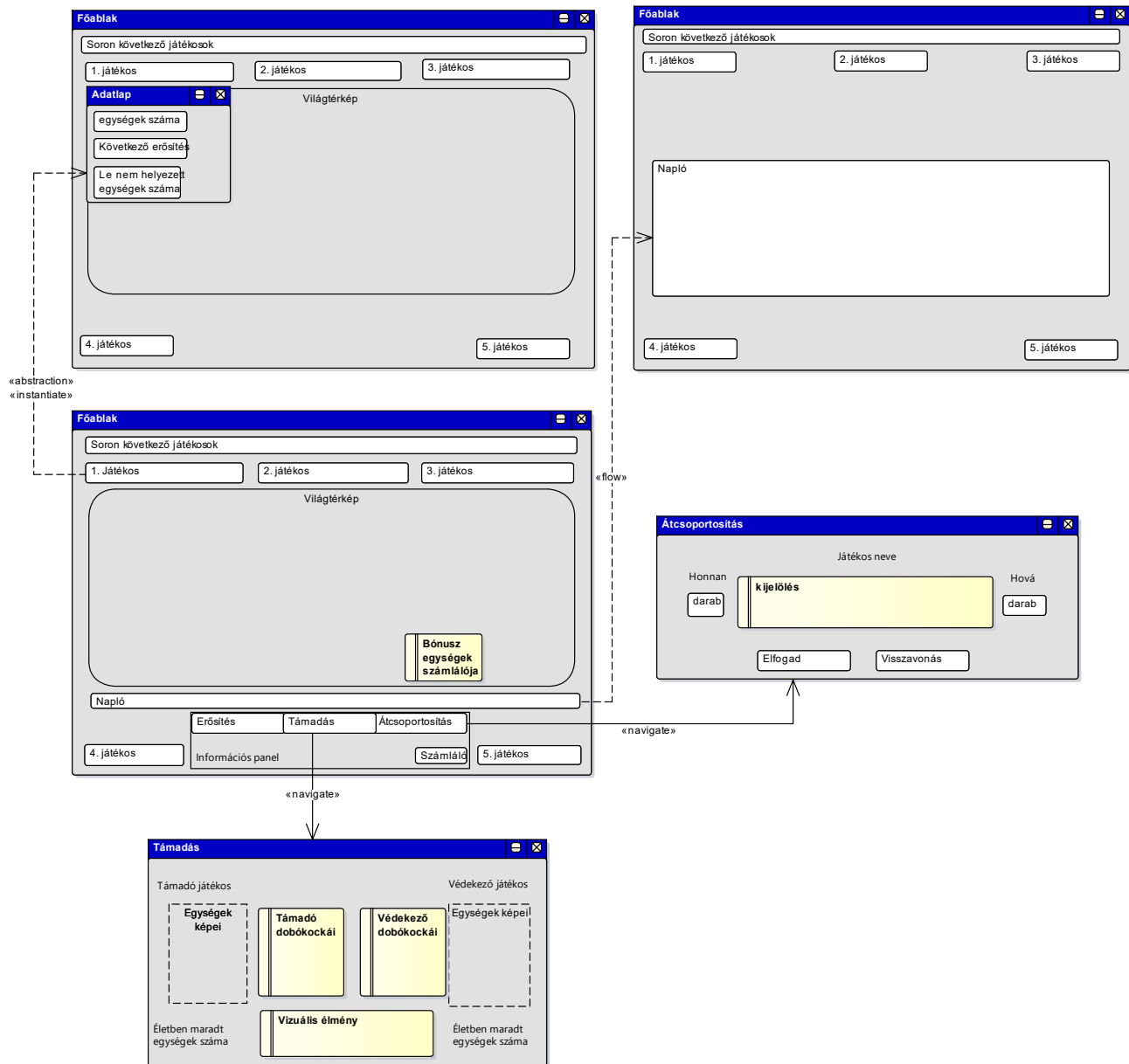egységek száma

Figure 24: Primary Forms
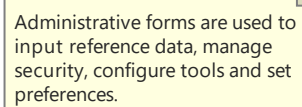
Administrative Forms

Package in package 'User Interface Model'

Administrative Forms
Version 1.0 Phase 1.0 Proposed
fodorad created on 2017. 03. 01.. Last modified 2017. 03. 01.

Administrative Forms diagram

Custom diagram in package 'Administrative Forms'

Administrative Forms
Version 1.0
Oliver created on 2017. 03. 01.. Last modified 2017. 03. 01.

Administrative forms are used to input reference data, manage security, configure tools and set preferences.

Figure 25: Administrative Forms

Help System

Package in package 'User Interface Model'

Help System
Version 1.0 Phase 1.0 Proposed
fodorad created on 2017. 03. 01.. Last modified 2017. 03. 01.

Help System diagram

Custom diagram in package 'Help System'

Help System
Version 1.0
Oliver created on 2017. 03. 01.. Last modified 2017. 03. 01.

Figure 26: Help System