HW3 – Reliable Inter-process Communication

Desing Issues

This documents contains some diagrams of the implemented protocols. In following pages, a component, sequence and class diagram are drawn respectively.

Component Diagram
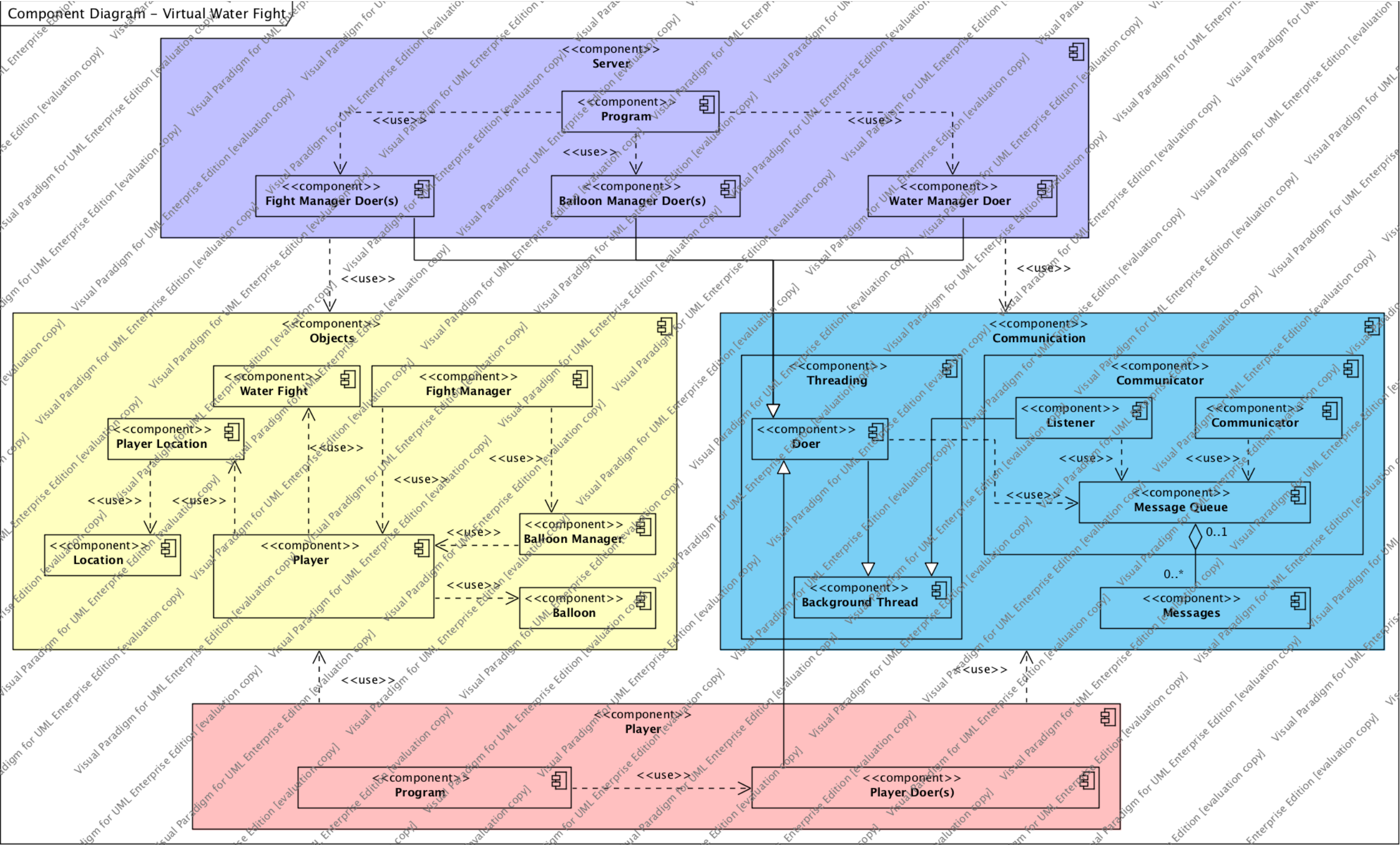This diagram shows all components and relationships between them in the game.
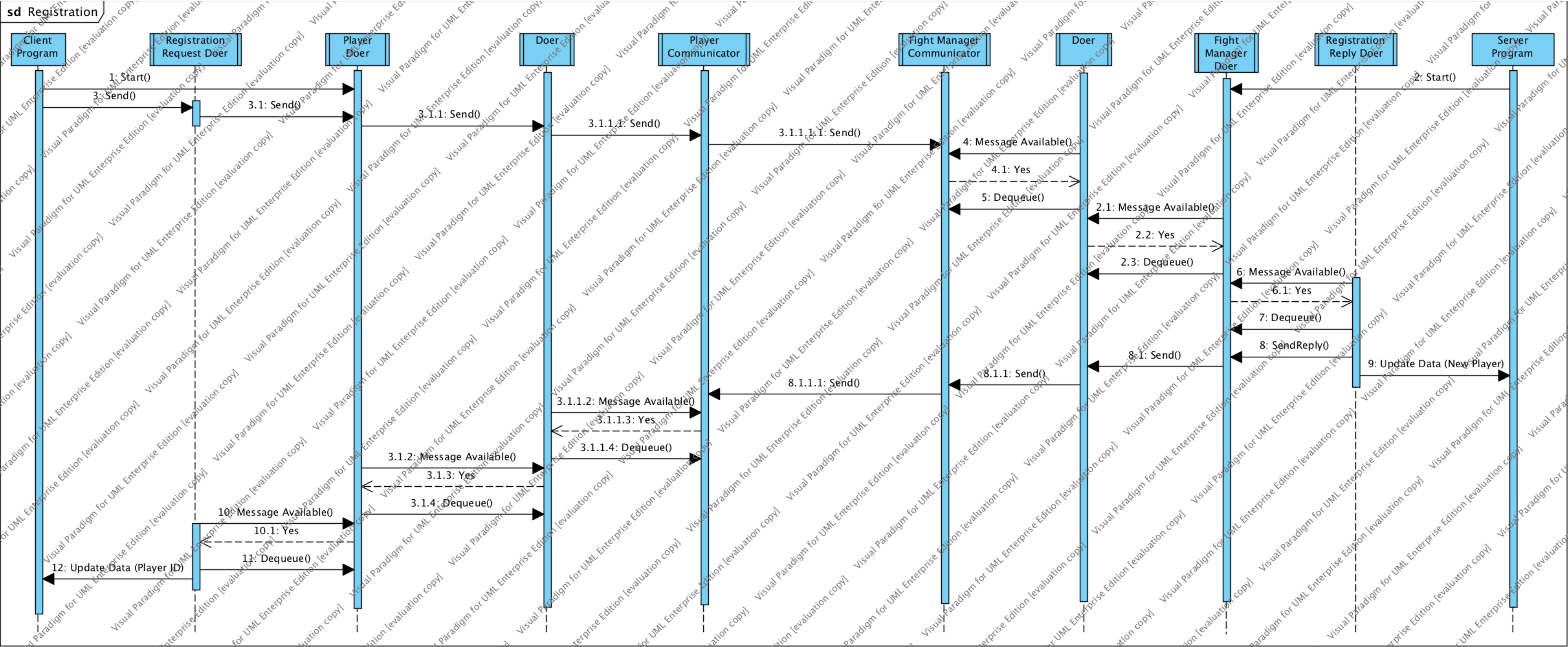
Sequence Diagram
In this one, only the process of registration is illustrated. Since all protocls follow a request-reply pattern for communication, this figure are presented as a sample.
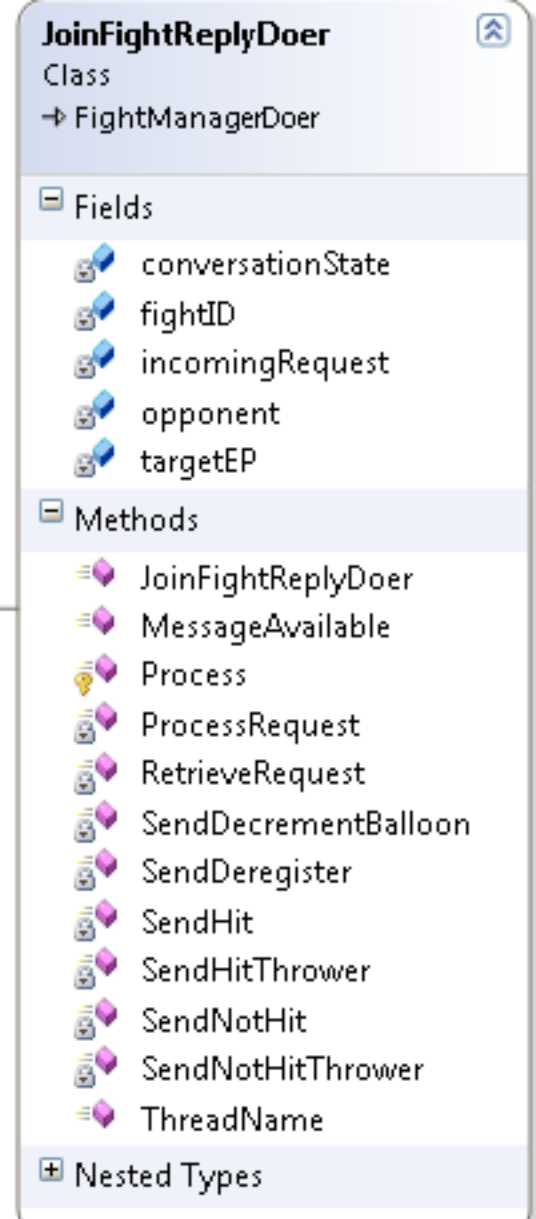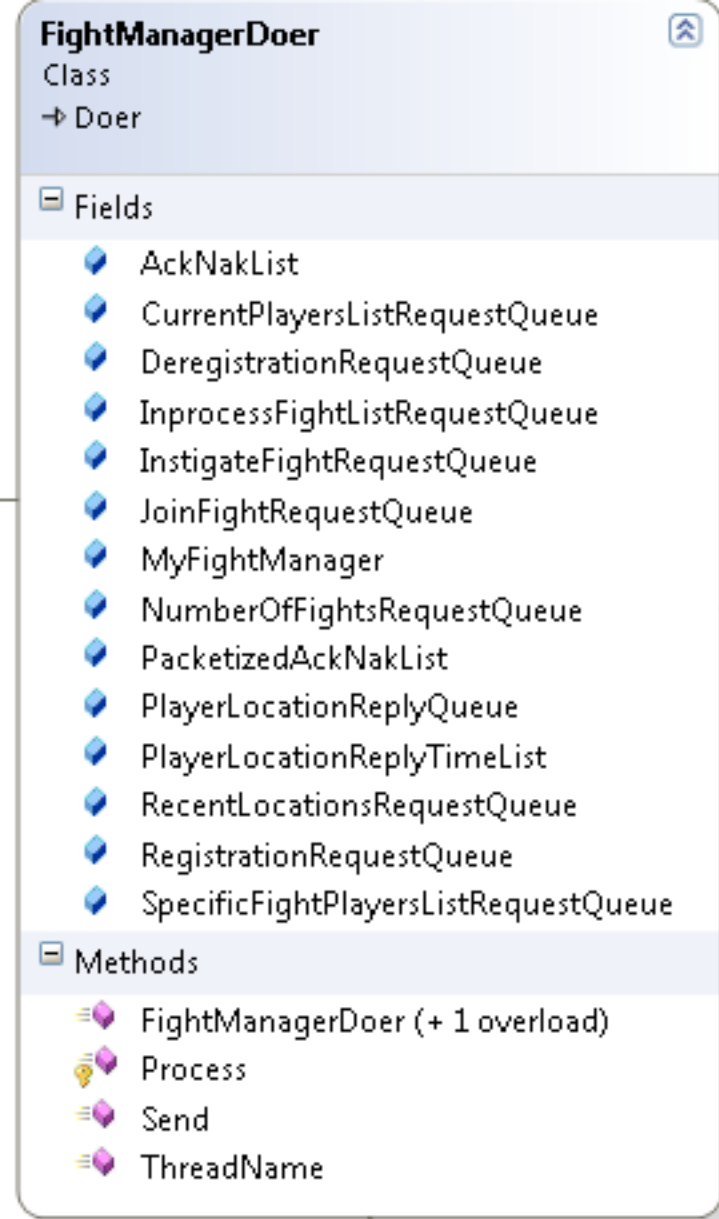
Class Diagram
Simillar to the previous diagram, some classes are drawn as samples. There are at least two specialized doer classes for each protocol, so some of them are selected to show the detail of implementation. Comprehensive class diagrams for derived classes from PlayerDoer, FightManagerDoer and BalloonManagerDoer are created in the related projects of my Visual C# project.

Laleh Rostami Hosoori
A01772483

Component Diagram – Virtual Water Fight

sd Registration

Client Program	Registration Request Doer	Player Doer	Doer	Player Communicator	Fight Manager Communicator	Doer	Fight Manager Doer	Registration Reply Doer	Server Program

1: Start()

2: Start()

3: Send()

3.1: Send()

3.1.1: Send()

3.1.1.1: Send()

3.1.1.1.1: Send()

4: Message Available()

4.1: Yes

5: Dequeue()

2.1: Message Available()

2.2: Yes

2.3: Dequeue()

6: Message Available()

6.1: Yes

7: Dequeue()

8: SendReply()

8.1: Send()

9: Update Data (New Player)

8.1.1: Send()

8.1.1.1: Send()

3.1.1.2: Message Available()

3.1.1.3: Yes

3.1.1.4: Dequeue()

3.1.2: Message Available()

3.1.3: Yes

3.1.4: Dequeue()

10: Message Available()

10.1: Yes

11: Dequeue()

12: Update Data (Player ID)

BackgroundThread
Abstract Class
▱

Fields
- 🔑 keepGoing
- 🔑 myThread
- 🔑 suspended

Properties
- 📋 Status
- 📋 Suspended

Methods
- 🟣 BackgroundThr...
- 🔑 Process
- 🟣 Resume
- 🟣 Start
- 🟣 Stop
- 🟣 Suspend
- 🟣 ThreadName

Doer
Class
➔ BackgroundThread
▱

Fields
- 🔷 myCommunicator
- 🔷 settings

Methods
- 🟣 Doer (+ 1 overload)
- 🟣 getOutOfQueue
- 🟣 getReplyType
- 🟣 getRequestType
- 🟣 IsRequest
- 🟣 MessageAvailable
- 🔑 Process
- 🟣 Send
- 🟣 ThreadName

FightManagerDoer
Class
➔ Doer

Fields
- 🔵 AckNakList
- 🔵 CurrentPlayersListRequestQueue
- 🔵 DeregistrationRequestQueue
- 🔵 InprocessFightListRequestQueue
- 🔵 InstigateFightRequestQueue
- 🔵 JoinFightRequestQueue
- 🔵 MyFightManager
- 🔵 NumberOfFightsRequestQueue
- 🔵 PacketizedAckNakList
- 🔵 PlayerLocationReplyQueue
- 🔵 PlayerLocationReplyTimeList
- 🔵 RecentLocationsRequestQueue
- 🔵 RegistrationRequestQueue
- 🔵 SpecificFightPlayersListRequestQueue

Methods
- 🟣 FightManagerDoer (+ 1 overload)
- 🔑 Process
- 🟣 Send
- 🟣 ThreadName

JoinFightReplyDoer
Class
➔ FightManagerDoer

Fields
- 🔷 conversationState
- 🔷 fightID
- 🔷 incomingRequest
- 🔷 opponent
- 🔷 targetEP

Methods
- 🟣 JoinFightReplyDoer
- 🟣 MessageAvailable
- 🔑 Process
- 🔷 ProcessRequest
- 🔷 RetrieveRequest
- 🔷 SendDecrementBalloon
- 🔷 SendDeregister
- 🔷 SendHit
- 🔷 SendHitThrower
- 🔷 SendNotHit
- 🔷 SendNotHitThrower
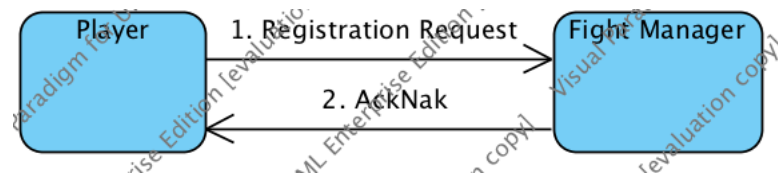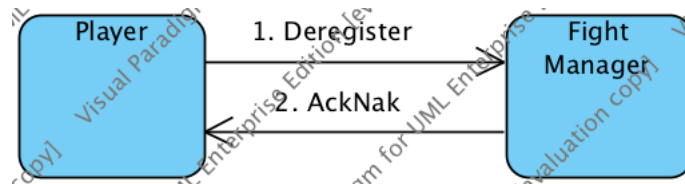- 🟣 ThreadName

⊞ Nested Types

Protocols

Following figures illustrate how defined protocols work. Each label is the name of a message class desinged for the specific protocol.
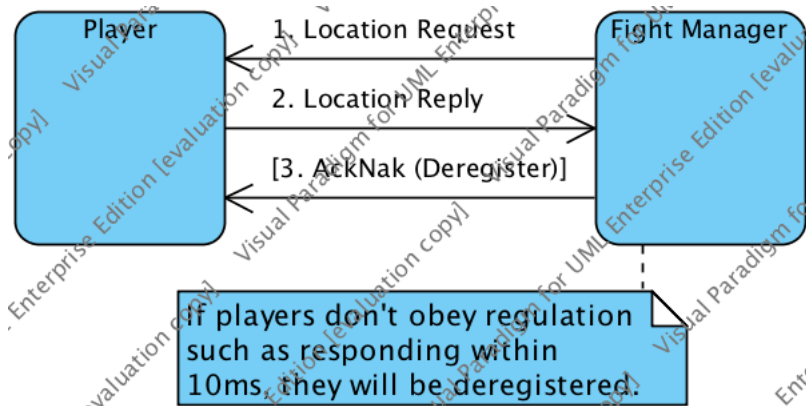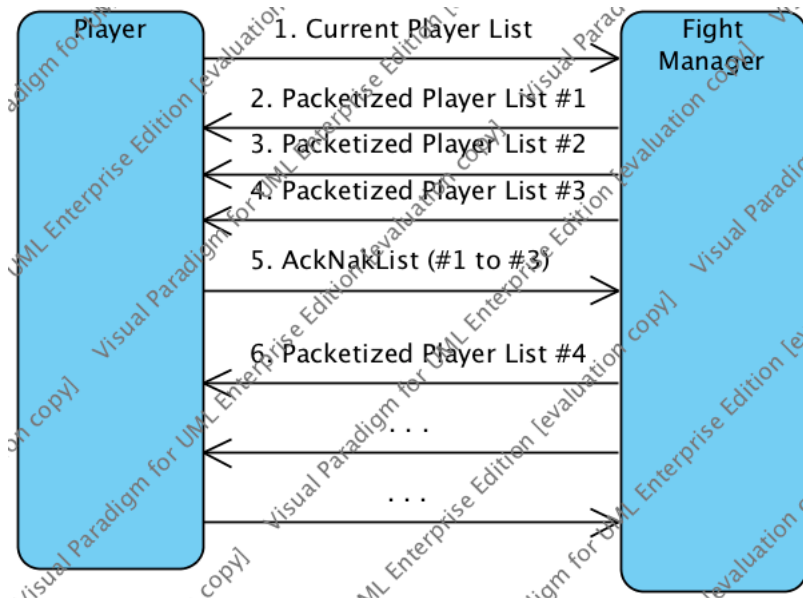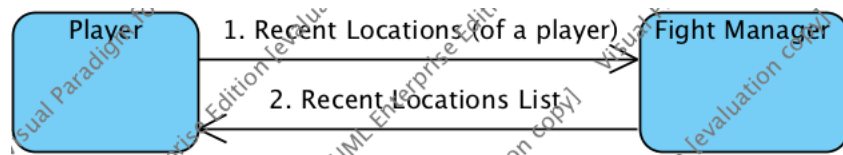
1. Registration

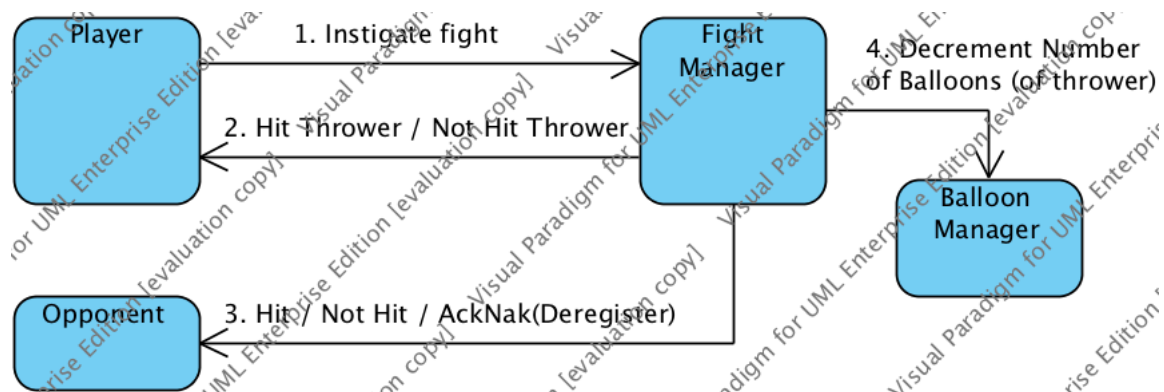## 2. Deregistration
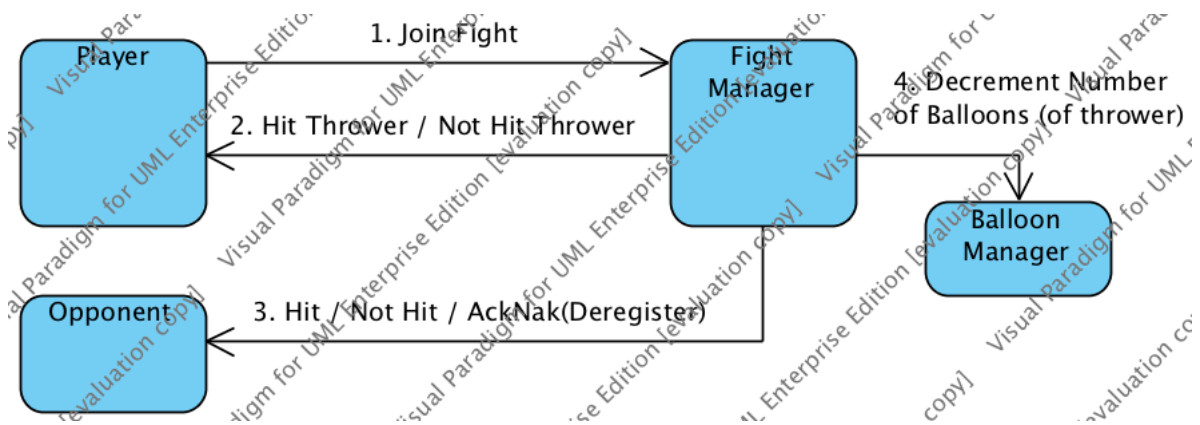

3. Player's Location
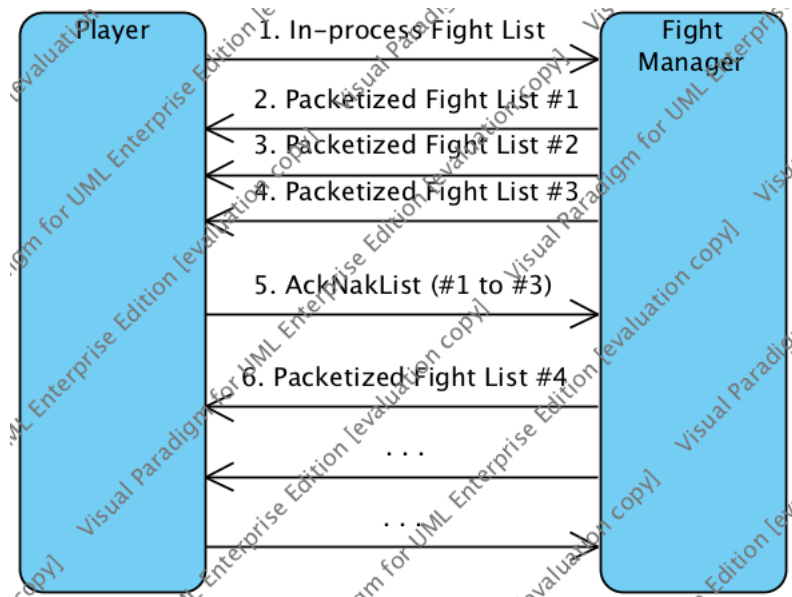
4. Current Player List

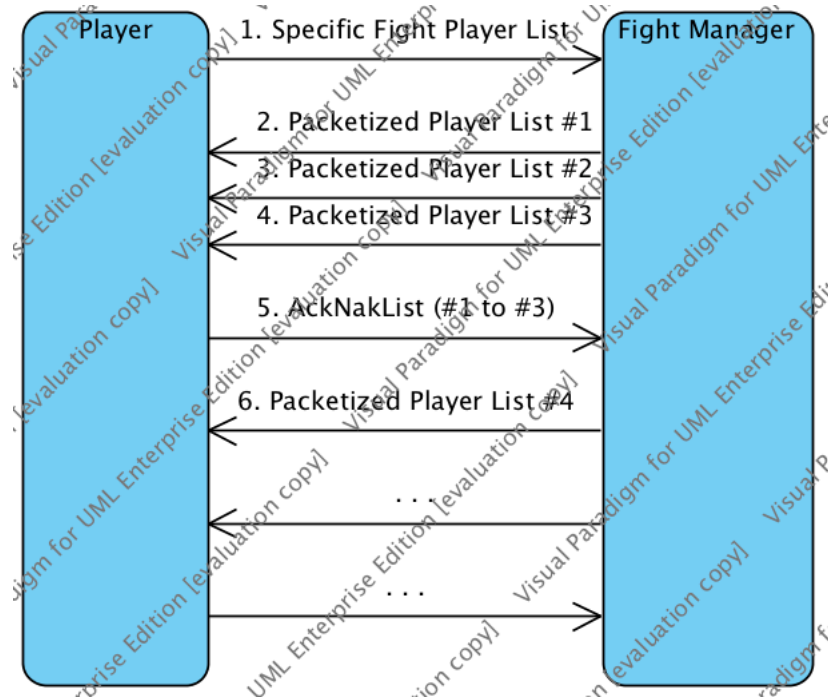

5. Player's Most Recent Location

6. Instigate a Fight

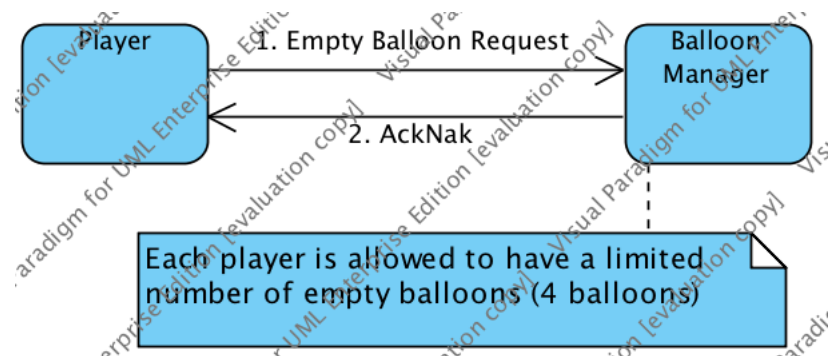Laleh Rostami Hosoori
A01772483

7. Join a Fight

8. In-process Fight List

Laleh Rostami Hosoori
A01772483

9. Players of a Specific Fight

10. An Empty Balloon Request

Laleh Rostami Hosoori
A01772483

11. Amount of Water Request