MVC Baserad Spel Applikation

Objektorienterade Applikationer

DAT055

Grupp 12

Filip Törnqvist *940813-7496* *torfi@student.chalmers.se*

Inlämningdeadline: Måndagen den 5 mars 2018

Sammanfattning

Innehåll

Sammanfattning **2**
Innehåll 2
Resultat 2

Spelidé **3**

Användarmanual **4**
GUI 4
Spelfaser 4
Uppstart 5

Programstruktur **6**
Klass Specifikation 6
Package: Controller 7
Package: Event 7
Package: General 8
Package: Init 8
Package: Model 8
Package: Net 8
Package: Util 8
Package: View 8

UML Klassdiagram **9**
Model Package 10
View Package 11
Controller Package 12
Net Package 13
Event Package 14
General Package 15

Resultat

Programmet är till största del en spelmotor för MVC baserade Java spel, implementeringen av själva Risk spelet är påbörjad men många delar av spelet är inte färdigutvecklade. För närvarande fungerar multiplayer och singleplayer versionen av spelet till synes på samma sätt. Det finns inget AI eller liknande utan alla motståndare måste vara andra spel instanser. Spelare har möjlighet att gå ihop och spela spelet tillsammans med hjälp av det interna Server / Client system som applikationen har implementerat. Spelet har problem med en del buggar relaterat till avstängning av multiplayer spel samt finns det problem med delar av spel logiken.

Spelidé

Idén med detta spelet är att utforska möjligheterna samt problemen som kan uppkomma med en strikt MVC implementation, och den önskade effekten med den strikta MVC implementationen skulle inte vara en snabbare implementering utan istället var poängen att programmet skulle bli ett extremt skalbart. MVC systemet skulle använda sig utav ett specialgjort event system som sköter all kommunikation mellan de olika MVC delarna. Samt var poängen att testa möjligheterna och problemen med ett multiplayer system som skickar dessa event till de olika spel instanserna. Spelet skulle fungera liknande ett typiskt risk spel där spelare tränar soldater och attackerar i ett tur och ordnings system. Vinnaren var den skulle vara den som tog över hela kartan.

Användarmanual

Detta Risk spelet kan spelas av en eller flera spelare via en nätverksuppkoppling. Spelet går ut på att vinna alla zoner genom krig, detta sker genom att en spelare tränar upp en arme i en av deras zoner och därefter attackerar en zon som samma spelar inte äger. Vinnaren av slaget är baserat till del på tur dock är antalet soldater också medräknat. En stor arme är starkare än flera små armeer, även om de små armeerna tillsammans har fler soldater än den stora armen. Spelare spelar inte samtidigt utan en spelare spelar sitt drag och därefter spelar nästa spelare sitt drag.

GUI

Vid uppstart av applikationen finns en meny upp till vänster där man kan välja olika alternativ. Efter ett spel är startat kommer kartan upp i mitten av fönstret och under spelkartan finns en yta med text som ger information om spelet. När det är den lokala spelarens tur så omringas text ytan av en grön färg samt sätts texten i högra hörnet på denna ytan till “Your turn!”. Till vänster i denna text ytan finns den lokala spelarens namn, höger om detta finns den totala produktionen för spelaren, till höger om detta finns den totala armee styrkan för spelaren, ytterligare till höger finns en text yta som indikerar vilken fas spelet är i samt en kort instruktion vad som ska / kan göras.

Spelfaser

Spelet börjar med att alla spelare i tur och ordning kan välja vilka zoner de ska börja med och detta sker genom att varje spelare placerar en soldat i de zonerna de vill ha, totalt placerar varje spelare 5 soldater i valfria zoner vid starten av spelet.

Efter start fasen börjar träningsfasen och där kan spelare träna soldater och antalet soldater de kan träna är baserat på hur många zoner de äger.

Efter träningsfasen är över börjar attack / flytt fasen, där varje spelare får chansen att attackera med eller flytta en av deras armeer. Detta sker genom att spelaren klickar ner på vänster musknapp på den zon soldaterna ursprungligen finns och därefter synliggörs markeringar som visar vart du kan flytta soldaterna. Soldaterna flyttas genom att du flyttar musen till den zon de ska flyttas till och därefter släpper vänster musknapp.

Efter attack / flytt fasen återgår spelet till träningsfasen.

När en spelare äger alla zoner vinner den och spelet pausas, ner till höger syns namnet på spelaren som vann.

Uppstart

Spelet är byggt för Java 1.8 på Windows men bör fungera likande på Linux.

1. Spelapplikationen startas genom att användaren dubbelklickar Risk.jar
    - För start av spelet krävs en karta att spela på, Default.map är standard och bör ligga i samma mapp som Risk.jar
    - Alternativt kan Default.map generas genom att användaren väljer “File->Create Map” i applikationen, detta kräver dock att bilden sistariskcolored.png finns i samma map som Risk.jar

2. Spel kan spelas lokalt själv eller i grupp via nätverket
    - “File->New Game” startar ett nytt lokalt spel och i detta alternativet kan du testa kontroller och hur spelet fungerar, för tillfället finns ingen motståndare i detta spelläget
    - “File->Host Game” startar ett nytt spel via nätverket och det kommer upp en lista på vilka spelare som försöker gå med i spelet, när host användaren väljer “Start Multiplayer Game” startar spelet, detta alternativ är enbart möjligt om det finns andra spelare i spelet
    - “File->Join Game” går med i ett spel via nätverket och detta kräver att ett spel är skapat på den adressen användaren skrev in samt att användares namn är unikt i spelet.

Programstruktur

Programmets kommunikation mellan de olika MVC delarna sker via events, av dessa event finns olika typer, de har prefix i namnet som indikerar var de kommer användas samt vad de kommer användas till, prefixen och förklaring är:

Lcl	Local Event	Lcl hittas i alla system, dessa kan användas till allt lokalt, detta inkluderar även lokalt på servrar.
Rpc	Remote Procedure Call	Alla Rpc event skickas automatiskt till servern, dessa bör innehålla information om vem som skickade dem samt så bör de inte lyssnas på utanför servern då de är mer av ett önskemål som skickas till servern.
Svr	Server Event	Alla Svr event skickas automatiskt ut från servern till alla klienter, dessa bör alltid lyssnas på då det är dessa event som bestämmer vad som händer i spelet.

Programmet har följande huvudklasser aktiva i olika spel uppsättningar:

- Alla spel uppsättningar

InstanceModel	Hanterar uppstarts logik och alla Model klasser
InstanceView	Hanterar uppstarts grafik och alla View klasser
InstanceController	Hanterar uppstarts input och alla Controller klasser

- Singleplayer

LocalPlayerController	Hanterar input från den lokala användaren
ServerGameModel	Hanterar själva spelets logik
LocalGameView	Hanterar spelets grafik

- Multiplayer Host

LocalPlayerController	Hanterar input från den lokala användaren
ServerGameModel	Hanterar själva spelets logik
LocalGameView	Hanterar spelets grafik
RemotePlayerControllers	Tar emot all input från andra uppkopplade spelare
RemotePlayerViews	Skickar Svr events till andra uppkopplade spelare

- Multiplayer Client

LocalPlayerController	Hanterar input från den lokala användaren
Client	Hanterar input och output från och till servern
LocalGameView	Hanterar spelets grafik

Klass Specifikation

Package: Controller

InstanceController - Hanterar alla controllers.
IPlayerController - Förtydligar att Local- och RemotePlayerController fyller likande syfte.
IResponse - Tillåter View-delen av programmet att skicka meddelanden controllern utan att bryta på MVC strukturen.
LocalPlayerController - Hanterar spel-input från den lokala användaren.
RemotPlayerControllers - Förtydligar programstrukturen genom skicka ServerClient klassernas input via en Controller som representerar de externa spelarna.

Package: Event

AEvent - Implementering av gemensam event kod.
AEventSystem - Implementering av typisk användnings kod av den globala EventManagern, syftet är att minska kod duplikation.
ANetEvent - Orginal syftet var att implementera Serializable men nu finns den kvar enbart för att förtydliga att klasser som använder den faktiskt ska skickas över nätet.
EventManager - Hanterar event kommunikation.
EventType - Syftet är att ge varje event en unik id som kan användas för att lätt identifiera dem.
IEventManager - Bas interface för en EventManager, fyller inget direkt syfte utöver utökning av skalbarhet.
IEventSystem - Bas interface som klasser bör implementera om de skall använda lyssnare.
LclAttackFromEvent - Används för att skapa visuella indikeringar om vart du kan attackera.
LclEndGameEvent - Används för att avsluta ett spel.
LclGenerateMap - Används för att generera en karta.
LclHostGameEvent - Används för att påbörja lyssnande efter potentiella klienter.
LclPreStartGameEvent - Används för att meddela delsystem om att ett spel kommer starta.
LclServerHostStartGameEvent - Används för att meddela servern att den kommer starta spelet.
LclStartGameEvent - Används för att påbörja ett spel.
LclStartGameHostEvent - Används för att avslytta lyssning efter nya klienter.
LclStartGameSentEvent - Används för att indikera att spelet kommer börja.
LclStopAttackFromEvent - Används för att ta bort visuella indikeringar om vart spelaren kan attakera.
RpcAttackZoneEvent - Används för att meddela servern att en spelare vill attackera eller flytta soldater.
RpcConnectEvent - Används för att meddela servern att klienten vill gå med.
RpcDisconnectEvent - Används temporärt inte, men syftet är att den ska skickas vid avstängen av servern / klienten.
RpcUpdateZoneEvent - Används för att försöka uppdatera en zon.
SvrAttackZoneEvent - Används för att försöka flytta till / attackera en zon.
SvrNextTurnEvent - Används för att uppdatera vems tur det är.
SvrStartGameEvent - Används för att starta spelet.
SvrUpdateZoneEvent - Används för att flytta uppdatera en zon.

Package: General

Image - Används för att spara en serializable bild.
Map - Används för att samla zoner och kombinera detta med en bakgrundsbild.
Phase - Används för att visa vilken fas spelet är i.
Zone - Används för att spara information om hur en specifik zon innehåller, samt vem som äger den.

Package: Init

Main - Används enbart i statup.

Package: Model

IGameModel - Används för att tillåta lätt skalbarhet vid vidareutveckling.
InstanceModel - Används för att hantera alla model klasser.
ServerGameModel - Används för att hantera en spelserver samt all spellogik.

Package: Net

Client - Används för att koppla upp till en server samt skicka events till och från den.
NetPlayer - Representerar en spelare samt dess roll i ett multiplayer spel.
Server - Hanterar lyssnade på klienter som vill gå med i spelet.
ServerClient - Hanterar en klient som är uppkopplad till servern, används på servern.

Package: Util

Delegate - Används för att tillåta en pointer till en member function, används i event systemet.
ErrorHandler - Används för att tillåta användarvänlig assert som ger tydlig information och kraschar programmet vid problem.
IDestroyable - Används för visa att en klass kan bli direkt förstörd genom att kalla destroy funktionen.
IResetable - Används för att visa att en klass kan bli helt återställd genom att kalla reset funktionen.
Pair - Används i Delegaten.
TimedEvent - Används för att kunna skicka ett event efter en viss tid eller tidigare om ett förbestämt event hittas.
Utility - Används för en fusk funktion som minskar sannolikheten för programmeringsfel.

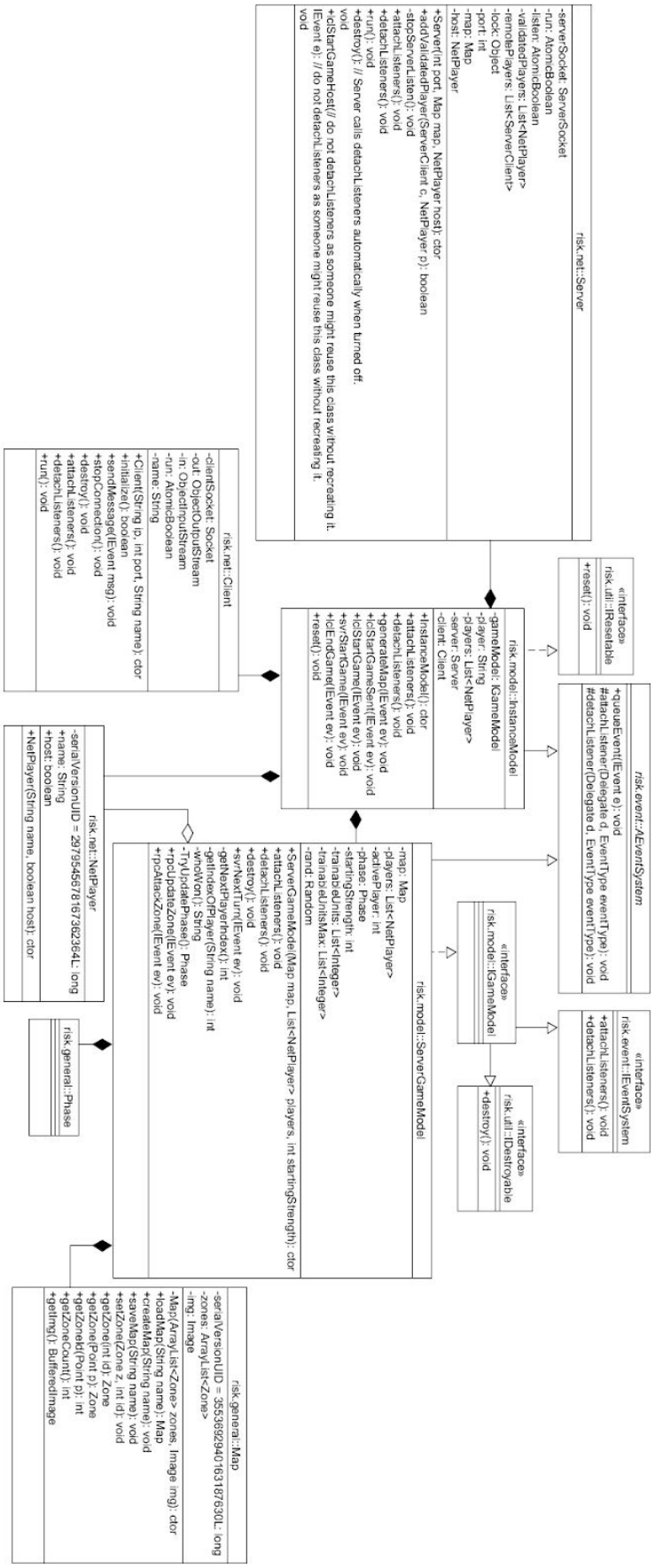
Package: View

HostPanelView - Visuell representering för hosten om vilka som är med när den väntar på andra spelare.
IGameView - Bas interface vars syfte är att visa att Local- och RemoteGameViews fyller sama syfte.
InstanceGameView - Används för att hantera alla view klasser samt skapa ett fönster med en meny.
LocalGameView - Används för att visa vad som händer i spelet för den lokala spelaren.
RemoteGameViews - Används för att förtydliga när det är andra externa spelare som också läser vissa events.
UIPanel - Används för att ge text indikationer på vad som händer i spelet.

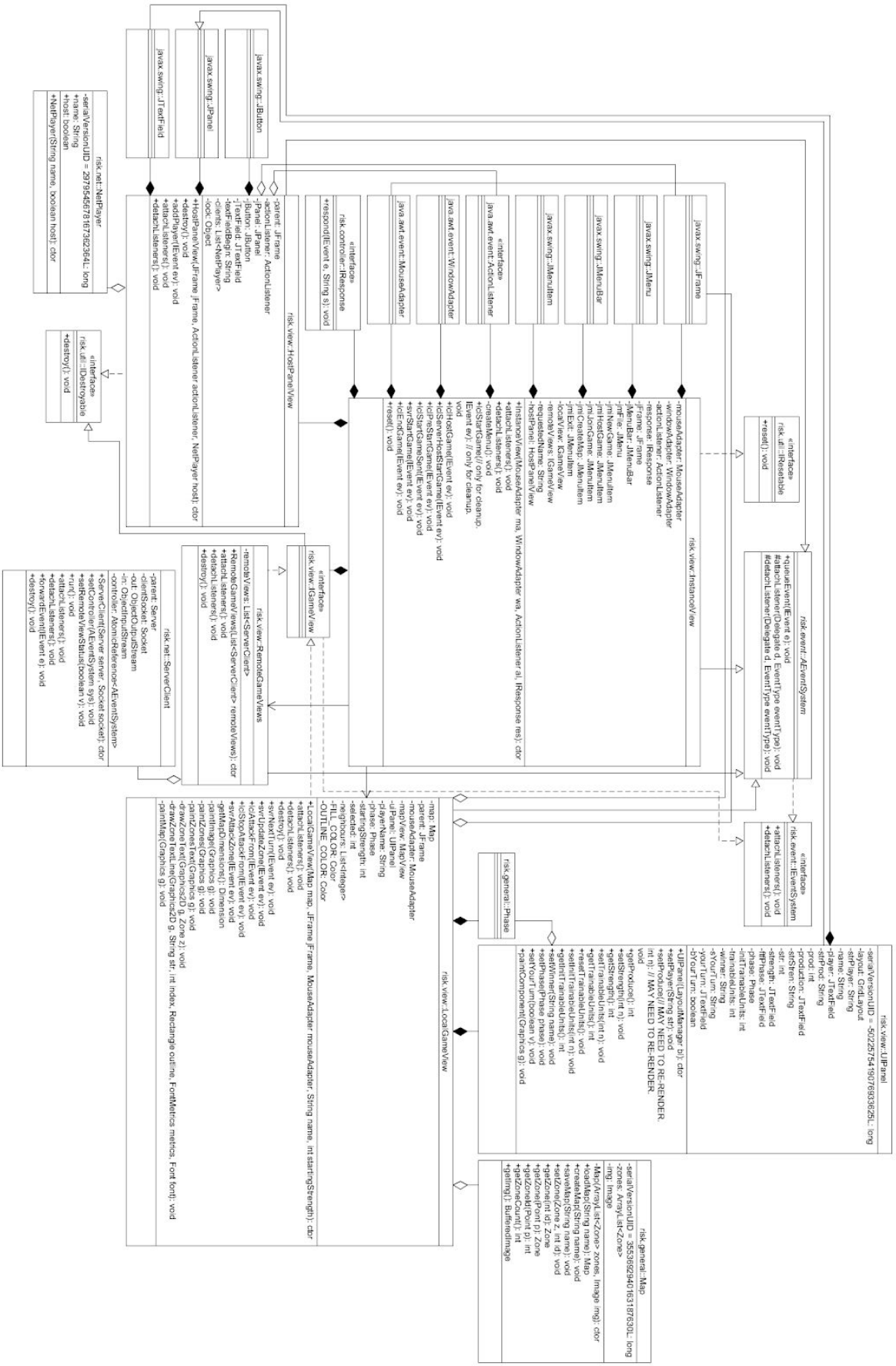
UML Klassdiagram

I UML diagrammet har många delar blivit borttagna samt så visas enbart ett package i taget, och de klasser som inte finns i det specifika package får inte en komplett klasshierarki eller sina relationer till associerade klasser uppvisade. Alla event har blivit ihopsatta till de tre olika grupper de tillhör.

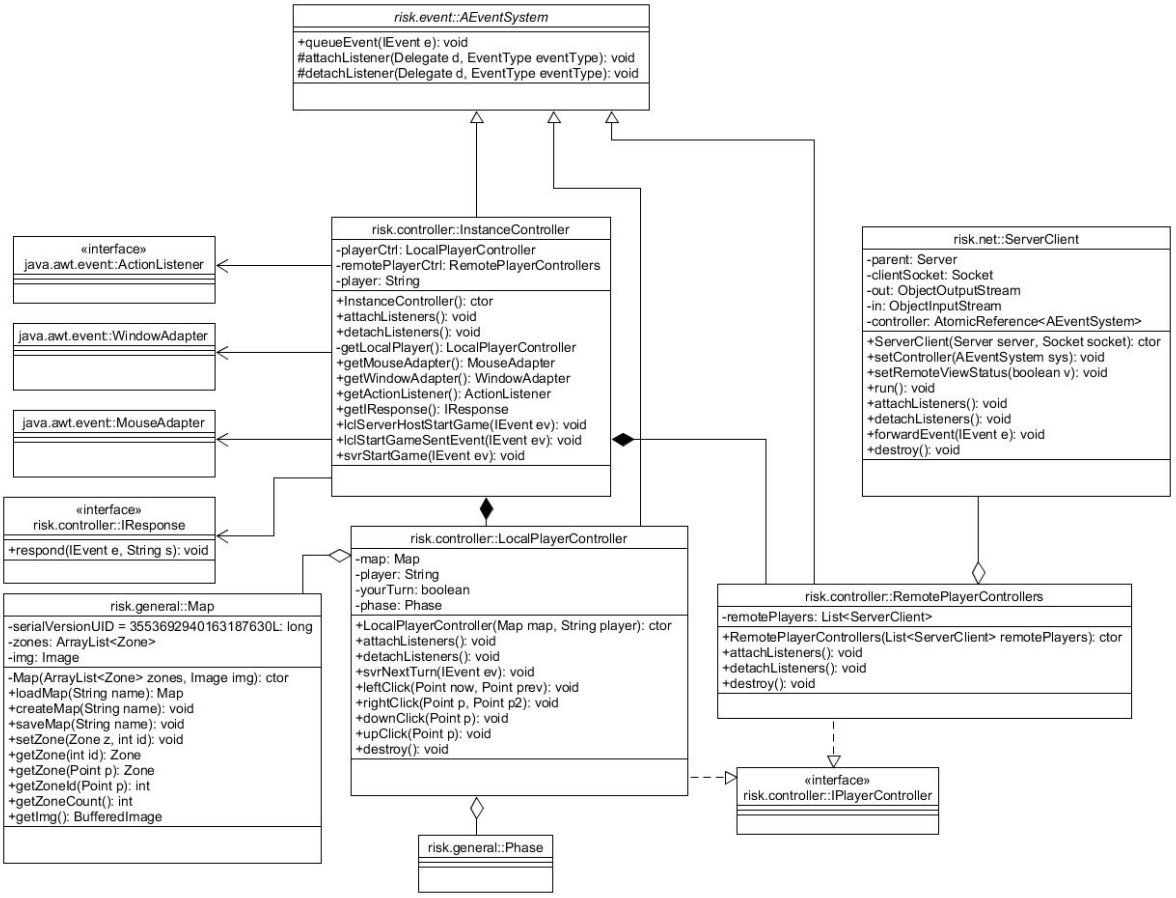
Model Package



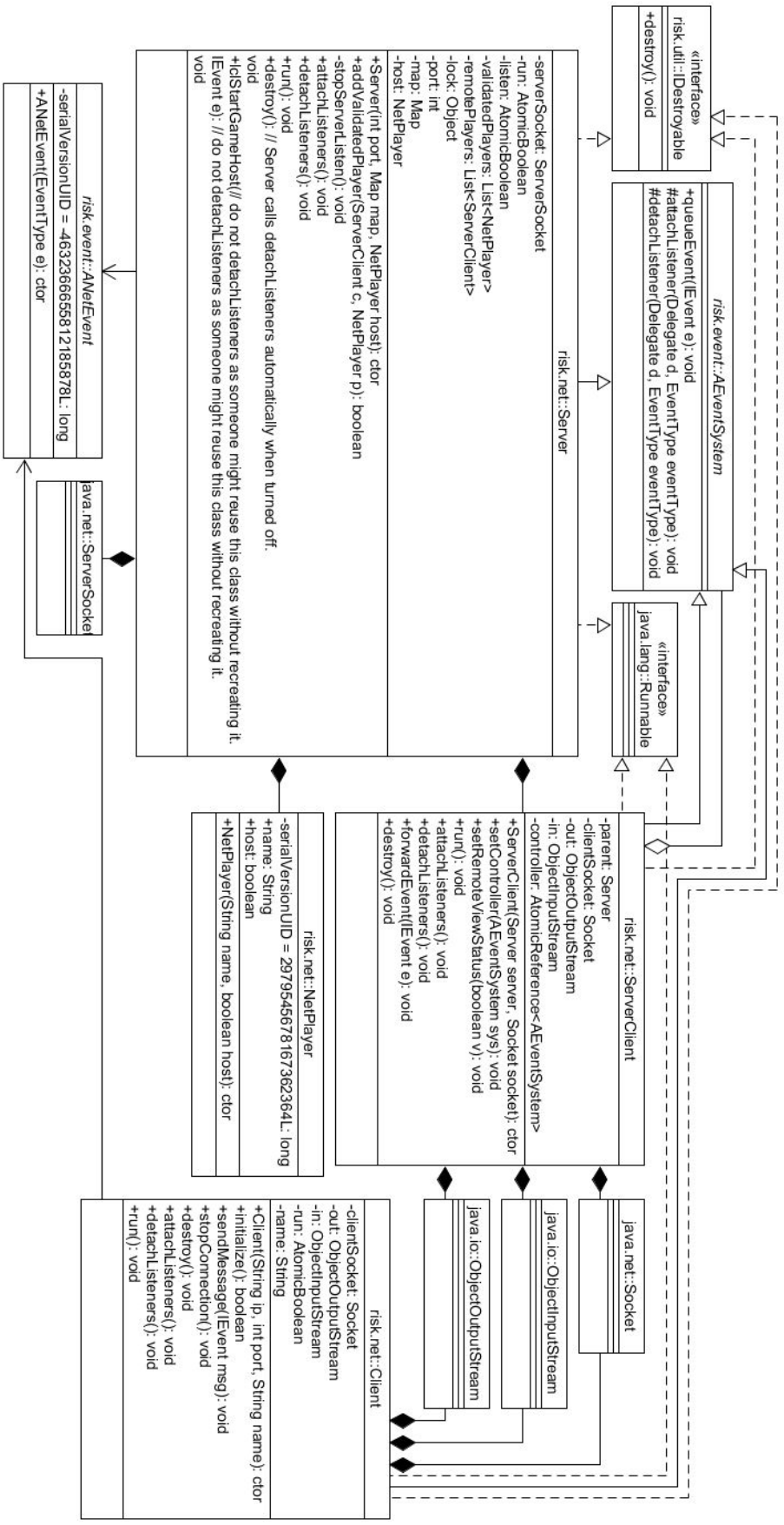
View Package



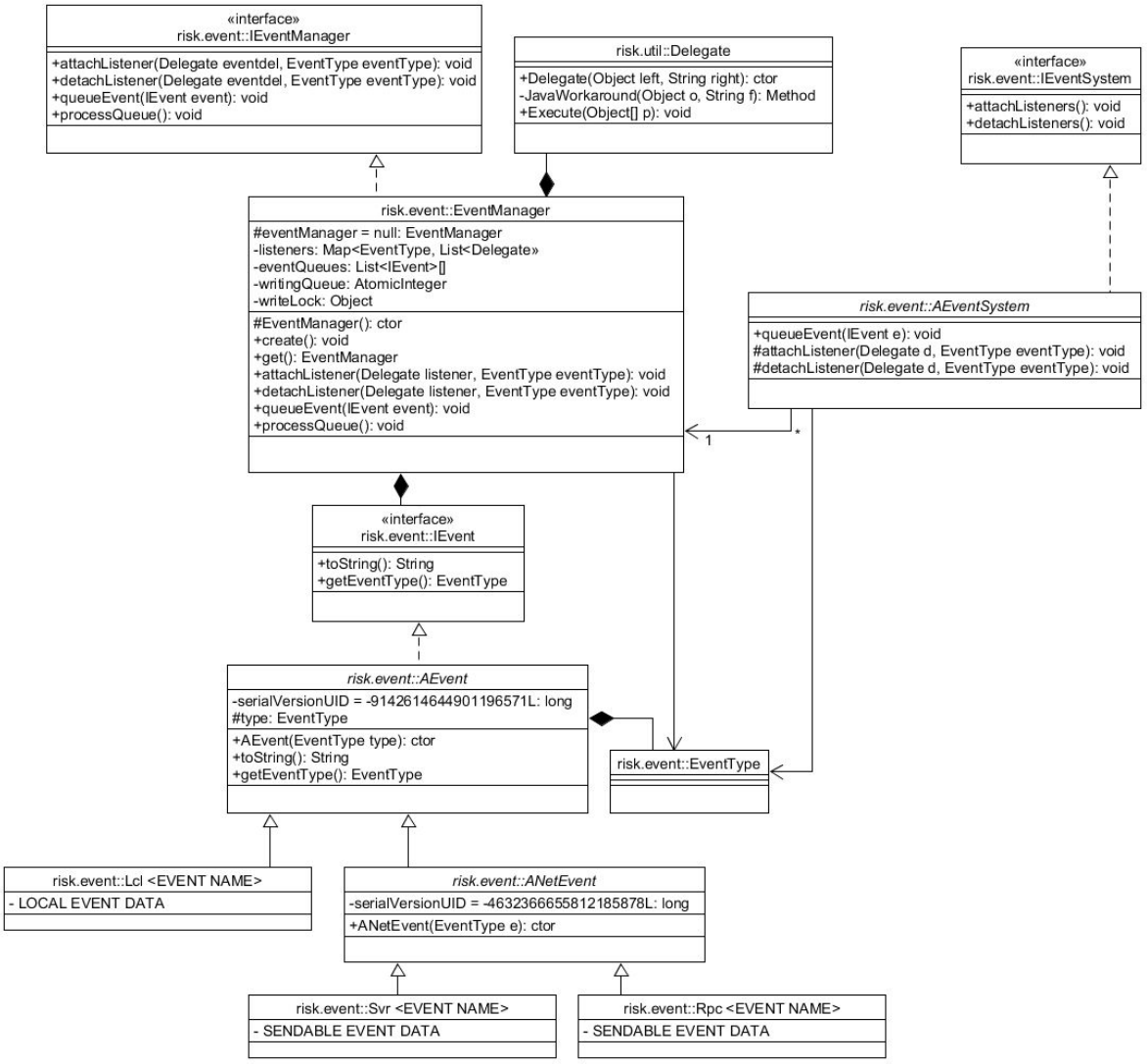
Controller Package



Net Package


«interface»
risk.util::IDestroyable
+destroy(): void
risk.event::AEventSystem
+queueEvent(IEvent e): void
#attachListener(Delegate d, EventType eventType): void
#detachListener(Delegate d, EventType eventType): void
«interface»
java.lang::Runnable
risk.net::Server
-serverSocket: ServerSocket
-run: AtomicBoolean
-listen: AtomicBoolean
-validatedPlayers: List<NetPlayer>
-remotePlayers: List<ServerClient>
-lock: Object
-port: int
-map: Map
-host: NetPlayer
+Server(int port, Map map, NetPlayer host): ctor
+addValidatedPlayer(ServerClient c, NetPlayer p): boolean
-stopServerListen(): void
+attachListeners(): void
+detachListeners(): void
+run(): void
+destroy(): // Server calls detachListeners automatically when turned off.
void
+lclStartGameHost(// do not detachListeners as someone might reuse this class without recreating it.
IEvent e): // do not detachListeners as someone might reuse this class without recreating it.
void
risk.net::ServerClient
-parent: Server
-clientSocket: Socket
-out: ObjectOutputStream
-in: ObjectInputStream
-controller: AtomicReference<AEventSystem>
+ServerClient(Server server, Socket socket): ctor
+setController(AEventSystem sys): void
+setRemoteViewStatus(boolean v): void
+run(): void
+attachListeners(): void
+detachListeners(): void
+forwardEvent(IEvent e): void
+destroy(): void
risk.net::NetPlayer
-serialVersionUID = 2979545678167362364L: long
+name: String
+host: boolean
+NetPlayer(String name, boolean host): ctor
risk.net::Client
-clientSocket: Socket
-out: ObjectOutputStream
-in: ObjectInputStream
-run: AtomicBoolean
-name: String
+Client(String ip, int port, String name): ctor
+initialize(): boolean
+sendMessage(IEvent msg): void
+stopConnection(): void
+destroy(): void
+attachListeners(): void
+detachListeners(): void
+run(): void
risk.event::ANetEvent
-serialVersionUID = -4632366655812185878L: long
+ANetEvent(EventType e): ctor
java.net::ServerSocket
java.net::Socket
java.io::ObjectInputStream
java.io::ObjectOutputStream

Event Package



General Package