

Memoria del proyecto Polis

Ingeniería del Software de Gestión II - Grupo 10 - Iteración 6

Samuel Navas Portillo
Juan Jesús Pérez Luna
Manuel de los Santos Campos
María José Sancha Maya
Ángel Martínez Olivares
José Antonio Jiménez Carmona

11 de enero de 2011

Índice general

1. Vídeo explicativo del juego	5
2. Planificación Temporal	7
2.1. Reuniones ordinarias	7
2.2. Entregas de las iteraciones	7
2.3. Reuniones extraordinarias	8
3. Memorando técnico	9
4. Seguimiento	11
4.1. Actividades	11
4.2. Resumen de tiempo invertido y puntuaciones	15
5. Diagrama UML de Análisis	17
6. Asignación de Responsabilidades	21
7. Diagrama UML de Diseño	43

Capítulo 1

Vídeo explicativo del juego

El siguiente vídeo online explica brevemente cómo se juega al juego de mesa Polis.

<http://www.youtube.com/watch?v=gTCdMBfJiHo>

Capítulo 2

Planificación Temporal

En una primera conversación del grupo realizada el 29 de Septiembre, se llega al siguiente acuerdo sobre la planificación de las reuniones regulares del grupo y de las distintas entregas del proyecto:

2.1. Reuniones ordinarias

Iteración 1

Cada martes de 9:30 a 11:30 y/o viernes de 12:30 a 14:30 de cada semana hasta que tenga lugar la última entrega del proyecto, el grupo debe reunirse para analizar el estado del proyecto y del grupo y, en su caso, discutir y tomar las decisiones que e tomen oportunas.

2.2. Entregas de las iteraciones

Iteración 1

La primera entrega del proyecto será el día 7 de Octubre de 2010. Las siguientes entregas serán, respectivamente, los días 20 de Octubre, 29 de Octubre, 5 de Noviembre, 12 de Noviembre, 19 de Noviembre, 26 de Noviembre, 3 de Diciembre, 10 de Diciembre, 17 de Diciembre y, por último, 14 de Enero.

Iteración 2

La primera entrega del proyecto será el día 7 de Octubre de 2010. Las siguientes entregas (2ª y 3ª Iteración) serán, respectivamente, los días 14 de Octubre y 21 de Octubre respectivamente.

Iteración 3

La primera entrega del proyecto será el día 7 de Octubre de 2010. Las siguientes entregas (2ª y 3ª Iteración) serán, respectivamente, los días 14 de Octubre y 11 de noviembre respectivamente.

Iteración 4

La entrega de la 4ª iteración será el día 3 de Diciembre.

Iteración 5

La entrega de la 5ª iteración será el día 18 de Diciembre.

Iteración 6

La entrega de la 6ª iteración será el día 11 de Enero.

2.3. Reuniones extraordinarias**Iteración 1**

El martes 5 de Octubre (de 17:30 a 20:30) y el miércoles 6 de Octubre (de 9:00 a 11:30, de 15:30 a 17:30 y de 19:30 a 21:30) tuvimos 2 reuniones extra respectivamente.

Capítulo 3

Memorando técnico

Este capítulo presenta un breve resumen de las actividades realizadas en cada iteración o etapa.

Iteración 1

En esta primera etapa se ha estudiado las reglas del juego, construido el juego de mesa y producido un vídeo explicativo del juego.

Iteración 2

Las actividades principales fueron el repaso del temario impartido por la asignatura Ingeniería del Software 1 y el diseño del diagrama UML de análisis del proyecto.

Iteración 3

En este tiempo se diseñó el diagrama UML de diseño del proyecto, se modificó el diagrama UML de análisis del proyecto, se realizó el diagrama de asignación de responsabilidades y se preparó el entorno técnico (Eclipse y Subversion) para el desarrollo del proyecto.

Iteración 4

En esta iteración se modificó el diagrama de asignación de responsabilidades y se implementó la arquitectura de código del proyecto y parte de sus clases y funciones.

Iteración 5

En esta etapa se ha completado el código del proyecto hasta terminarlo. Para la próxima etapa, queda pendiente la refactorización de algunas partes del código

Iteración 6

Esta fase nos ha llevado a refactorizar el código del proyecto en una gran parte (incluida la interfaz de usuario de consola), para hacer de éste un código más legible, más modular, con menos acoplamiento y más reutilizable. Esta iteración ha supuesto un esfuerzo bastante grande para el grupo ya que ha sido necesario un análisis de todo el proyecto en profundidad, ya que no era trivial determinar qué partes del código había que cambiar. También, en fase parte del desarrollo, hemos realizado las primeras pruebas de ejecución completa del programa, es decir, hemos usado el programa tomando el papel de usuarios para comprobar que funcionaba correctamente.

Capítulo 4

Seguimiento

Este capítulo presenta las actividades concretas realizadas por cada miembro del grupo así como su tiempo invertido.

4.1. Actividades

Iteración 1

- **Samuel Navas Portillo**

1. Búsqueda de materiales: **1h 30min**
2. Fabricación de las fichas y cartas del juego: **2h**
3. Lectura de las Normas: **1h**
4. Impresión de las Normas del juego: **1h 30min**
5. Simulación de partidas: **2h**
6. Guión del vídeo (1ª parte): **2h**
7. Guión del vídeo (2ª parte): **2h**

- **Juan Jesús Pérez Luna**

1. Análisis y síntesis de las Normas del juego: **3h**
2. Impresión del juego: **30min**
3. Fabricación de las fichas y cartas del juego: **2h**
4. Lectura de las Normas: **1h**
5. Simulación de partidas: **2h**
6. Guión del vídeo y actor (mano) (1ª parte): **2h**
7. Guión del vídeo (2ª parte): **1h**

- **Manuel de los Santos Campos**

1. Lectura de las Normas: **2h**
2. Simulación de partidas: **2h**
3. Guión del vídeo (1ª parte): **2h**

4. Guión del vídeo y actor (mano) (2ª parte): **2h**

■ **María José Sancha Maya**

1. Búsqueda y preparación del equipo de grabación del video: **30min**
2. Fabricación tablero: **2h**
3. Lectura de las Normas: **1h**
4. Simulación de partidas: **2h**
5. Guión del vídeo (1ª parte): **2h**
6. Grabación del vídeo (cámara): **2h**
7. Edición del vídeo (postproducción): **5h**

■ **Ángel Martínez Olivares**

1. Búsqueda de materiales: **1h 30min**
2. Fabricación de las fichas y cartas del juego: **2h**
3. Lectura de las Normas: **1h**
4. Simulación de partidas: **2h**
5. Guión del vídeo y actor (voz) (1ª parte): **2h**
6. Guión del vídeo y actor (voz) (2ª parte): **3h**

■ **José Antonio Jiménez Carmona**

1. Impresión del juego: **30min**
2. Fabricación de las fichas y cartas del juego: **2h**
3. Lectura de las Normas: **1h**
4. Simulación de partidas: **1h 30min**
5. Guión del video (1ª parte): **2h**
6. Aprendizaje básico de "Latexz producción de la Memoria (versión 1.0): **4h**

Iteración 2

■ **Samuel Navas Portillo**

1. Estudio/repaso del tema de análisis de Ingeniería del Software de Gestión 1 y propuesta de diseño UML: **10h**

■ **Juan Jesús Pérez Luna**

1. Estudio/repaso del tema de análisis de Ingeniería del Software de Gestión 1 y propuesta de diseño UML: **10h**

■ **Manuel de los Santos Campos**

1. Estudio/repaso del tema de análisis de Ingeniería del Software de Gestión 1 y propuesta de diseño UML: **10h**

■ **María José Sancha Maya**

1. Estudio/repaso del tema de análisis de Ingeniería del Software de Gestión 1 y propuesta de diseño UML: **10h**

■ **Ángel Martínez Olivares**

1. Estudio/repaso del tema de análisis de Ingeniería del Software de Gestión 1 y propuesta de diseño UML: **10h**

■ **José Antonio Jiménez Carmona**

1. Estudio/repaso del tema de análisis de Ingeniería del Software de Gestión 1 y realización de Memoria: **10h**

Iteración 3

■ **Samuel Navas Portillo**

1. Diseño del diagrama UML de análisis, diseño del diagrama UML de diseño, implementación de parte del proyecto (código), asignación de responsabilidades e implantación de Eclipse y Subversion: **8h**

■ **Juan Jesús Pérez Luna**

1. Diseño del diagrama UML de análisis, diseño del diagrama UML de diseño, diseño de la estructura del código fuente del proyecto, implementación de parte del proyecto (código), asignación de responsabilidades e implantación de Eclipse y Subversion: **10h**

■ **Manuel de los Santos Campos**

1. Implantación de Eclipse y Subversion y asignación de responsabilidades: **5h**

■ **María José Sancha Maya**

1. Implantación de Eclipse y Subversion y traducciones: **4h**

■ **Ángel Martínez Olivares**

1. Implantación de Eclipse y Subversion y asignación de responsabilidades: **5h**

■ **José Antonio Jiménez Carmona**

1. Diseño del diagrama UML de diseño, implantación de Eclipse y Subversion y ampliación de la Memoria: **6h**

Iteración 4

■ **Samuel Navas Portillo**

1. Implementación de parte del código: **30**

■ **Juan Jesús Pérez Luna**

1. Implementación de parte del código: **30**

■ **Manuel de los Santos Campos**

1. Diagrama de responsabilidades: **15**

■ **María José Sancha Maya**

1. Pruebas unitarias: **9**

■ **Ángel Martínez Olivares**

1. Diagrama de responsabilidades: **12**

■ **José Antonio Jiménez Carmona**

1. Implementación de parte del código, Memoria: **15**

Iteración 5

■ **Samuel Navas Portillo**

1. Implementación de parte del código: **30**

■ **Juan Jesús Pérez Luna**

1. Implementación de parte del código y de la interfaz de usuario: **30**

■ **Manuel de los Santos Campos**

1. Pruebas unitarias: **30**

■ **María José Sancha Maya**

1. Pruebas unitarias: **10**

■ **Ángel Martínez Olivares**

1. Implementación de parte del código: **20**

■ **José Antonio Jiménez Carmona**

1. Implementación de parte del código, Memoria: **10**

Iteración 6

■ **Samuel Navas Portillo**

1. Refactorización del código: **?**

■ **Juan Jesús Pérez Luna**

1. Refactorización del código: **?**

■ **Manuel de los Santos Campos**

1. Pruebas: **?**

■ **María José Sancha Maya**

1. Pruebas: **?**

- **Ángel Martínez Olivares**

1. Pruebas: ?

- **José Antonio Jiménez Carmona**

1. Refactorización del código, Memoria: ?

4.2. Resumen de tiempo invertido y puntuaciones

Se presenta el total de horas de trabajo invertidas por cada miembro del grupo y la puntuación de esta iteración para cada miembro del grupo resultante de su esfuerzo individual (puntos repartidos entre los 6 miembros de 30 a repartir).

Iteración 1

Nombre	Total de tiempo	Puntuación
Samuel Navas Portillo	12h	5
Juan Jesús Pérez Luna	11h 30min	5
Manuel de los Santos Campos	8h	5
María José Sancha Maya	14h 30min	5
Ángel Martínez Olivares	11h 30min	5
José Antonio Jiménez Carmona	11h	5

Iteración 2

Nombre	Total de tiempo	Puntuación
Samuel Navas Portillo	22h	5
Juan Jesús Pérez Luna	21h 30min	5
Manuel de los Santos Campos	18h	5
María José Sancha Maya	24h 30min	5
Ángel Martínez Olivares	21h 30min	5
José Antonio Jiménez Carmona	21h	5

Iteración 3

Nombre	Total de tiempo	Puntuación
Samuel Navas Portillo	30h	6
Juan Jesús Pérez Luna	31h 30min	8
Manuel de los Santos Campos	23h	4
María José Sancha Maya	28h 30min	4
Ángel Martínez Olivares	26h 30min	4
José Antonio Jiménez Carmona	27h	4

Iteración 4

Nombre	Total de tiempo	Puntuación
Samuel Navas Portillo	60h	6
Juan Jesús Pérez Luna	61h 30min	6
Manuel de los Santos Campos	38	5
María José Sancha Maya	37h 30min	3
Ángel Martínez Olivares	38h 30min	4
José Antonio Jiménez Carmona	42	6

Iteración 5

Nombre	Total de tiempo	Puntuación
Samuel Navas Portillo	90h	6
Juan Jesús Pérez Luna	91h 30min	6
Manuel de los Santos Campos	68	6
María José Sancha Maya	47h 30min	1
Ángel Martínez Olivares	58h 30min	5
José Antonio Jiménez Carmona	52	6

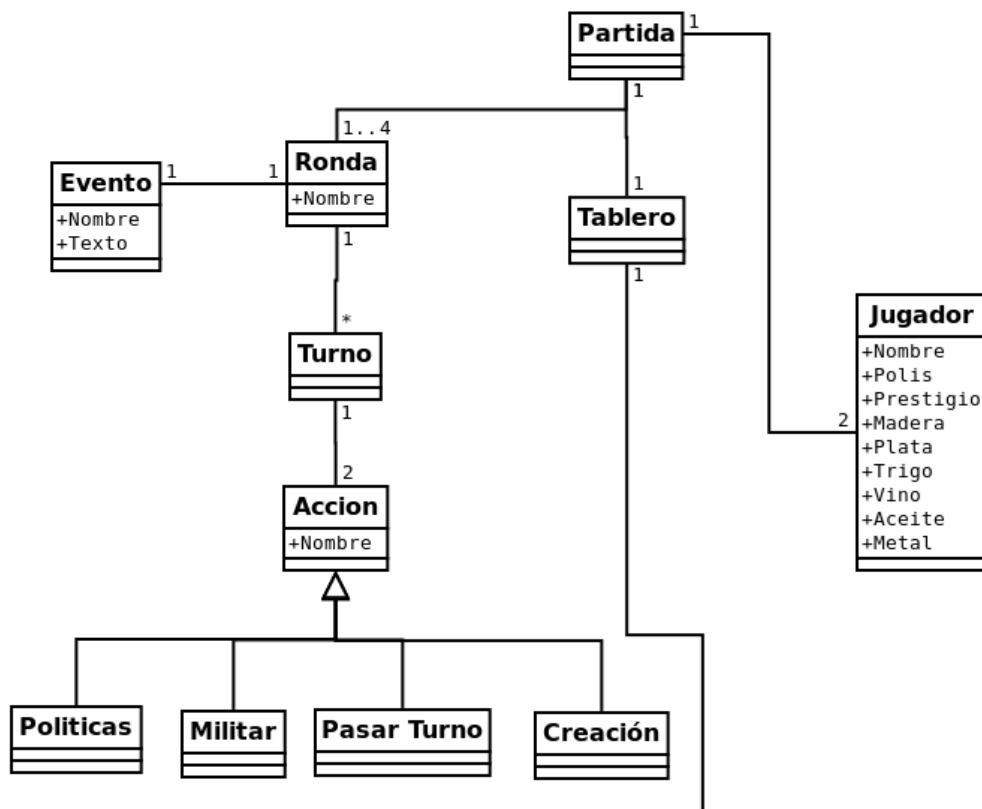
Iteración 6

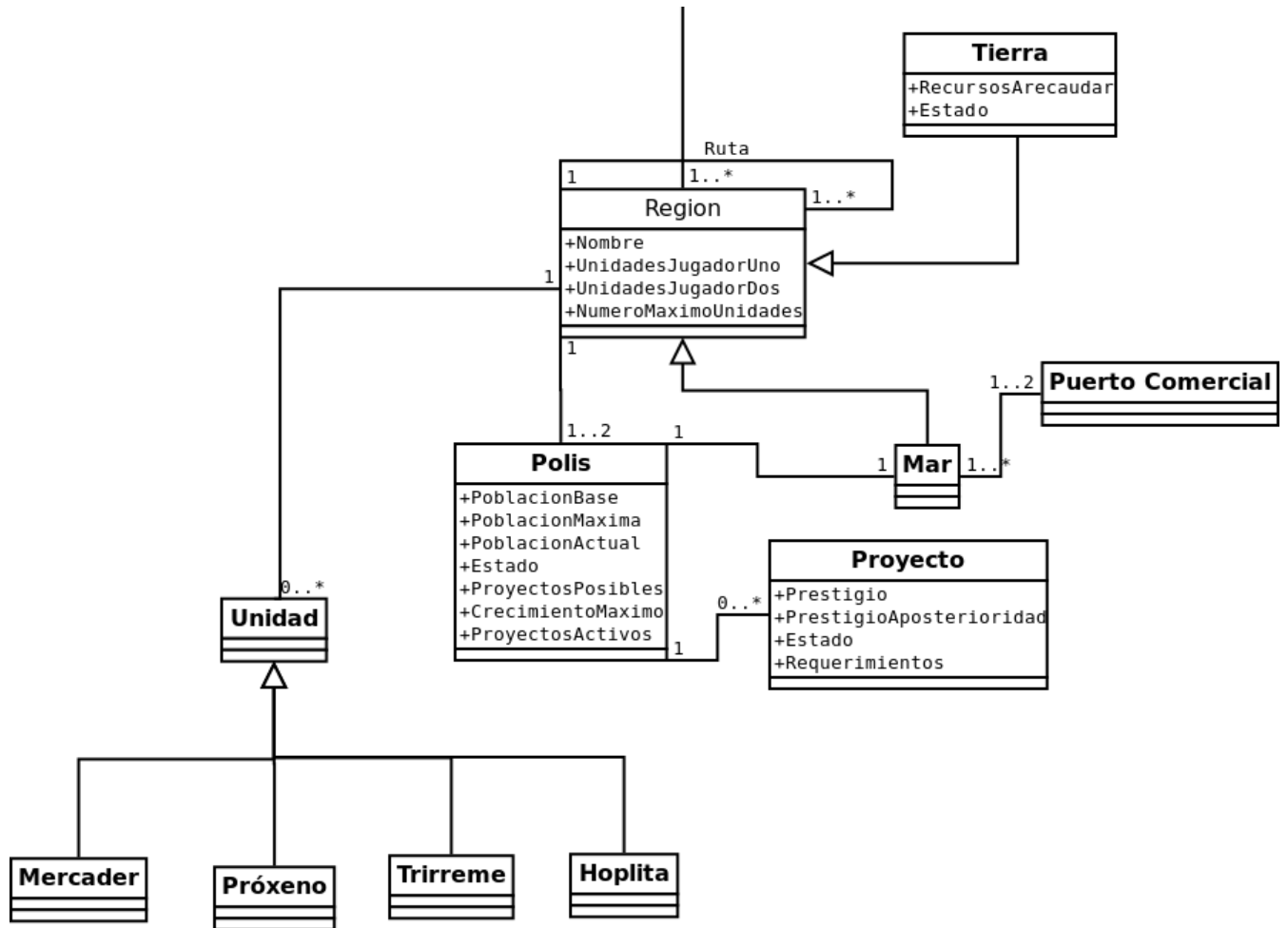
Nombre	Total de tiempo	Puntuación
Samuel Navas Portillo	?	?
Juan Jesús Pérez Luna	?	?
Manuel de los Santos Campos	?	?
María José Sancha Maya	?	?
Ángel Martínez Olivares	?	?
José Antonio Jiménez Carmona	?	?

Capítulo 5

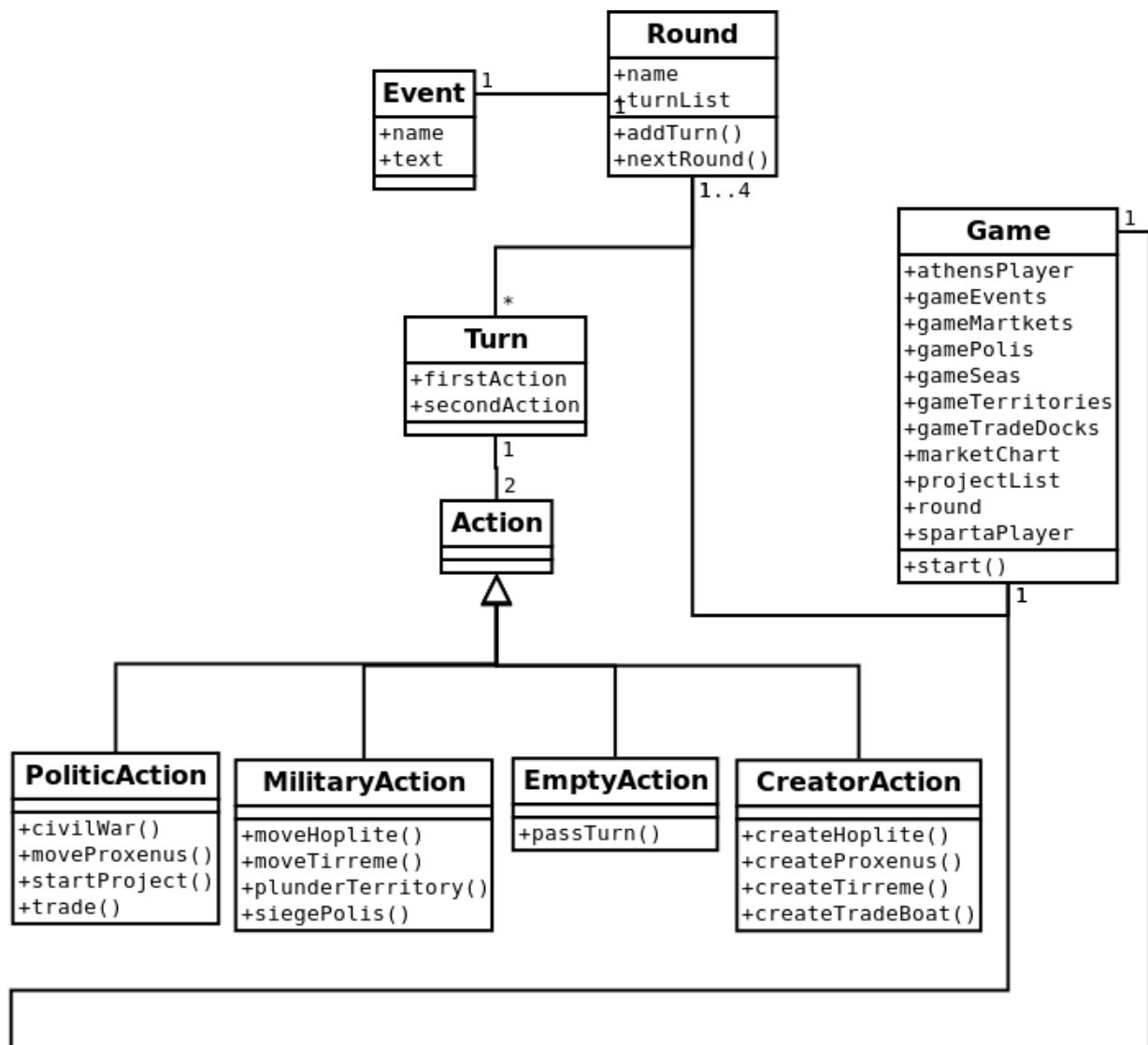
Diagrama UML de Análisis

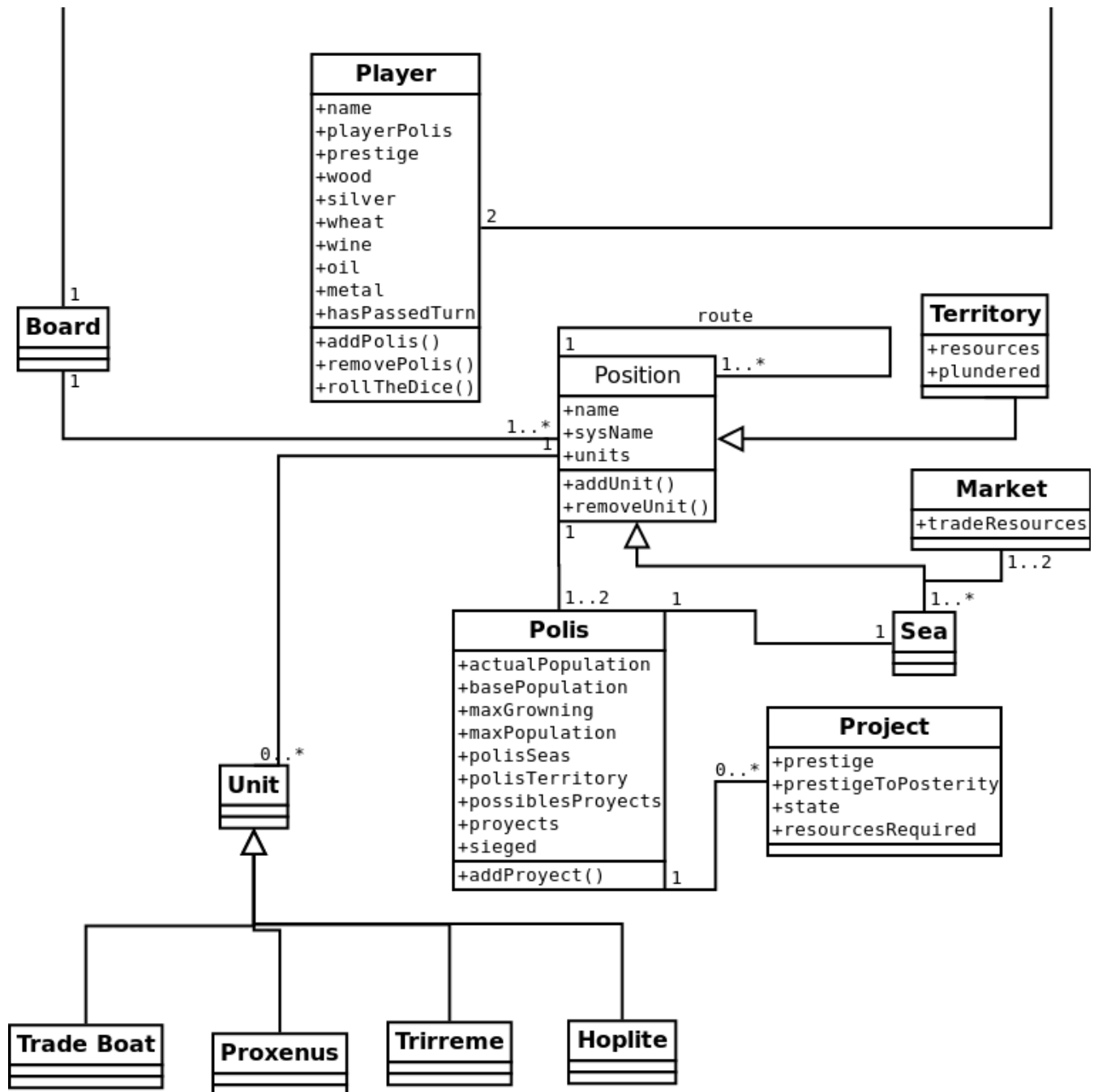
Iteración 2





Iteración 3





Capítulo 6

Asignación de Responsabilidades

Iteración 3

Nota: Hemos considerado como IU cuando no sólo llama directamente a un método de interfaz para comunicarse con el usuario, sino también cuando es el método que en su interior llamará al método concreto de interfaz de usuario.

Identificador	Descripción de la acción de alto nivel			
Polis-0001	Inicializa el juego con todos los elementos iniciales y jugadores de forma estándar.			
Pasos (usar pseudocódigo o similar)				
1. Se carga elementos del juego. 2. Se pide nombre de jugadores. 3. Se prepara posición inicial estándar.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Téc.	IU
1	ElementsInitializer	InitializeGameElements()		NO
2	ElementsInitializer	InitializeGameElements()		SI
3	StandarStartInitializer	void standardStart(Game theGame)		NO
Método de alto nivel				
void StandardStartInitializer()				
Diagrama de Colaboración (Opcional)				

Identificador		Descripción de la acción de alto nivel		
Polis-0002		Inicia la ronda del juego cargando y aplicando los proyectos.		
Pasos (usar pseudocódigo o similar)				
1. Se reparten 3 cartas de proyecto y 1 carta de evento.				
2. Se aplica evento.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	Round	startRound()		NO
2	Round	startRound()		NO
Método de alto nivel				
// TODO				
Diagrama de Colaboración (Opcional)				

Identificador		Descripción de la acción de alto nivel		
Polis-0003		Turnos, donde jugadores realizan sus dos acciones respectivamente.		
Pasos (usar pseudocódigo o similar)				
1. Jugador uno realiza sus acciones.				
2. Jugador dos realiza sus acciones.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	Action		0006	SI
2	Action		0007	SI
Método de alto nivel				
//TODO				
Diagrama de Colaboración (Opcional)				

Identificador	Descripción de la acción de alto nivel			
Polis-0004	Finaliza la ronda, resolviendo cada uno de los ajustes para empezar la siguiente ronda.			
Pasos (usar pseudocódigo o similar)				
1. Resolver Asedio. 2. Resolver Proyecto. 3. Alimentación. 4. Crecimiento. 5. Megalópolis. 6. Ajuste de bienes. 7. Phoros. 8. Preparar siguiente ronda.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	EndRoundManager	void checkSieges()		NO
2	EndRoundManager	void checkProjects()		NO
3	EndRoundManager	void checkFedding()		NO
4	EndRoundManager	void checkGrowth ()		NO
5	EndRoundManager	void checkMegalopolis()		NO
6	EndRoundManager	void checkGoodsAdjust()		NO
7	EndRoundManager	void checkPhoros()		NO
8	EndRoundManager	void initializeNextRound()		NO
Método de alto nivel				
// TODO				
Diagrama de Colaboración (Opcional)				

Identificador	Descripción de la acción de alto nivel			
Polis-0005	Finalizamos el juego resolviendo los parámetros necesarios para designar al ganador del juego.			
Pasos (usar pseudocódigo o similar)				
1. Comprobación de capitales. 2. Comprobación de prestigio. 3. Comprobación general. 4. Se declara el ganador del juego. 5. Finaliza juego.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	EndGameManager	void checkCapitals()		NO
2	EndGameManager	void checkNoPrestige()		NO
3	EndGameManager	void checkStandarEndGame()		NO
4	EndGameManager	Player getWinner()		NO
5	EndGameManager	void endTheGame()		NO
Método de alto nivel				
void EndGameManager()				
Diagrama de Colaboración (Opcional)				

Identificador	Descripción de la acción de alto nivel			
Polis-0006	Turnos, donde jugadores realizan sus dos acciones respectivamente			
Pasos (usar pseudocódigo o similar)				
3.1.a Jugador hace una acción creación.				
3.1.b Jugador hace acción militar.				
3.1.c Jugador hace acción política.				
3.1.d Jugador hace acción pasa turno.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Téc.	IU
1	CreateAction		0008	SI
2	MillitaryAction		0009	SI
3	PoliticAction		0010	SI
4	EmptyAction	void passTurn()		SI
Método de alto nivel				
//TODO				
Diagrama de Colaboración (Opcional)				

Identificador	Descripción de la acción de alto nivel			
Polis-0007	Turnos, donde jugadores realizan sus dos acciones respectivamente.			
Pasos (usar pseudocódigo o similar)				
3.2.a Si jugador ha realizado acción creación realiza 3.1.b,3.1.c .				
3.2.b Si jugador ha realizado acción militar realiza 3.1.a,3.1.c .				
3.2.c Si jugador ha realizado acción política realiza 3.1.a,3.1.b.				
3.2.d Jugador pasa turno.				
3.2.e Si jugador ha pasado turno no realiza acción.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Téc.	IU
1	Action		0011	SI
2	Action		0012	SI
3	Action		0013	SI
4	EmptyAction	void passTurn()		SI
5	Player	void getHasPassedTurn()		NO
Método de alto nivel				
Diagrama de Colaboración (Opcional)				

Identificador	Descripción de la acción de alto nivel			
Polis-0008	Acciones de creación, Jugador decide cual realizar.			
Pasos (usar pseudocódigo o similar)				
8.1.a Crear Hoplita				
8.1.b Crear Trirreme				
8.1.c Crear Próximo				
8.1.d Crear Barco de Comercio				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	CreateAction	void createHoplite()		SI
2	CreateAction	void createTrirreme()		SI
3	CreateAction	void createProxenus()		SI
4	CreateAction	void createTradeBoat()		SI
Método de alto nivel				
Diagrama de Colaboración (Opcional)				

Identificador	Descripción de la acción de alto nivel			
Polis-0009	Acciones militares, el Jugador decide cual realizar.			
Pasos (usar pseudocódigo o similar)				
9.1. a Mover Hoplita.				
9.1. b Mover Trirreme.				
9.1. c Recaudar.				
9.1. d Asediar .				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Téc.	IU
1	MilitaryAction	void moveHoplite()		SI
2	MilitaryAction	void moveTrirreme()		SI
3	MilitaryAction	void plunderTerritory()		SI
4	MilitaryAction	void siegePolis()		SI
Método de alto nivel				
Diagrama de Colaboración (Opcional)				

Identificador		Descripción de la acción de alto nivel		
Polis-0010		Acciones políticas, Jugador decide cual realizar		
Pasos (usar pseudocódigo o similar)				
10.1. a Guerra Civil. 10.1. b Mover Proxeno. 10.1. c Comenzar proyecto. 10.1. d Comerciar.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Téc.	IU
1	PoliticAction	void civilWar()		SI
2	PoliticAction	void moveProxenus ()		SI
3	PoliticAction	void startProyect()		SI
4	PoliticAction	void trade()		SI
Método de alto nivel				
Diagrama de Colaboración (Opcional)				

Identificador		Descripción de la acción de alto nivel		
Polis-0011		Jugador elige entre 2 acciones		
Pasos (usar pseudocódigo o similar)				
11.a Jugador hace una acción militar 11.b Jugador hace acción política				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Téc.	IU
1	MillitaryAction		0009	SI
2	PoliticAction		0010	SI
Método de alto nivel				
Diagrama de Colaboración (Opcional)				

Identificador		Descripción de la acción de alto nivel		
Polis-0012		Jugador elige entre acción de creación o política.		
Pasos (usar pseudocódigo o similar)				
11. a Jugador hace una acción creación.				
11. b Jugador hace acción política.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	CreateAction		0008	SI
2	PoliticAction		0010	SI
Método de alto nivel				
Diagrama de Colaboración (Opcional)				

Identificador		Descripción de la acción de alto nivel		
Polis-0013		Jugador elige entre acción de creación o militar.		
Pasos (usar pseudocódigo o similar)				
11. a Jugador hace una acción creación.				
11. b Jugador hace acción militar.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	CreateAction		0008	SI
2	MilitaryAction		0009	SI
Método de alto nivel				
Diagrama de Colaboración (Opcional)				

Iteración 4

Identificador	Descripción de la acción de alto nivel			
Polis-001	Inicializa el juego con todos los elementos iniciales			
Pasos (usar pseudocódigo o similar)				
1.Inicializa el juego a partir de un fichero 2.Inicializa los territorios 3.Inicializa los mares 4.Inicializa los puertos comerciales 5.Inicializa los mercados 6.Inicializa los proyectos 7.Inicializa las polis 8.Inicializa los eventos 9.Inicializa la ronda 10.Inicializa las tablas de comercio 11.Inicialización de los jugadores 12.Inicializa el juego con todos los elementos				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Téc.	IU
1	PolReader	[] polReader()		NO
2	PolReader	[Map] readTerritoris()		NO
3	PolReader	[Map] readSeas()		NO
4	PolReader	[Map] readTradeDocks()		NO
5	PolReader	[Map] readMarkets()		NO
6	PolReader	[Map] readProjects()		NO
7	PolReader	[Map] readPolis()		NO
8	PolReader	[Map]readGameEvents()		NO
9	Round	[] Round()		NO
10	MarketChart	[MarkedChart]MarketChart()		NO
11	TextModeUI	[List]requestPlayersGame()		SI
12	Game	[]Game()		NO
Método de alto nivel				
[void] ElementsInitializer ()				
Diagrama de Colaboración (Opcional)				

Identificador	Descripción de la acción de alto nivel			
Polis-0002	Comprueba el final de la ronda			
Pasos (usar pseudocódigo o similar)				
1.Comprueba las polis asediadas 2.Comprueba los proyectos y quien recibe el prestigio 3.Comprueba alimentación de la población 4.Comprueba el crecimiento de la población 5.Comprueba Megalópolis 6.Reajuste de recursos perecederos 7.Comprueba Phoros 8.Inicializa siguiente ronda				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	EndRoundManager	[void]checkSieges(Game: game,Player: player)		NO
2	EndRoundManager	[void]checkProjects(Player: player)		NO
3	EndRoundManager	[void]checkFeeding(Player :player)		NO
4	EndRoundManager	[void]checkGround(Player:player)		SI
5	EndRoundManager	[void]checkMegalopolis(Player :player)		NO
6	EndRoundManager	[void]checkGoodsAjust(Player: player)		NO
7	EndRoundManager	[void]checkPhoros(Player: player)		SI
8	EndRoundManager	[void]initializeNextRound()	0003	NO
Método de alto nivel				
[void] EndRoundManager ()				
Diagrama de Colaboración (Opcional)				

Identificador		Descripción de la acción de alto nivel		
Polis-0003		Inicia la ronda del juego cargando y aplicando los proyectos.		
Pasos (usar pseudocódigo o similar)				
1. Se reparten 3 cartas de proyecto y se elige un evento.				
2. Se aplica evento.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	Round	[void]startRound()	0004	NO
2	Round	[void] startRound()		NO
Método de alto nivel				
void startRound()				
Diagrama de Colaboración (Opcional)				

Identificador	Descripción de la acción de alto nivel			
Polis-0004	Obtener una lista de elementos al azar.			
Pasos (usar pseudocódigo o similar)				
1. Se selecciona una lista de elementos al azar.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	RandomCollection	[List<T>]getRandomSubList (List<T> list, Integer elementCount)		NO
Método de alto nivel				
void RandomCollection()				
Diagrama de Colaboración (Opcional)				

Identificador	Descripción de la acción de alto nivel			
Polis-0005	Inicia la ronda del juego cargando y aplicando los proyectos.			
Pasos (usar pseudocódigo o similar)				
1. Se reparten 3 cartas de proyecto y se elige un evento.				
2. Se aplica evento.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	Round	[void]startRound()		NO
2	Round	[void] startRound()		NO
Método de alto nivel				
void startRound()				
Diagrama de Colaboración (Opcional)				

Identificador	Descripción de la acción de alto nivel			
Polis-0006	Finalizamos el juego resolviendo los parámetros necesarios para designar al ganador del juego.			
Pasos (usar pseudocódigo o similar)				
1. Comprueba si el jugador ha perdido la capital.				
2. Comprueba el prestigio.				
3. Comprobación del modo estándar de ganar.				
4. Se declara el ganador del juego.				
5. Finaliza juego.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Téc.	IU
1	EndGameManager	[void] checkCapitals(Player:player)		NO
2	EndGameManager	[void] checkNoPrestige(Player:player)		NO
3	EndGameManager	[void] checkStandarEndGame(Player player1, Player player2)	0007	NO
4	EndGameManager	[Player] getWinner()		NO
5	EndGameManager	[void] endTheGame()		NO
Método de alto nivel				
void EndGameManager()				
Diagrama de Colaboración (Opcional)				

Identificador	Descripción de la acción de alto nivel			
Polis-0007	Comprobación del modo estándar para ganar , primero comprobando prestigio y luego recursos			
Pasos (usar pseudocódigo o similar)				
1. Obtención de prestigio 2. Obtencion de recursos				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Téc.	IU
1	EndGameManager	[int] getPlayerTotalPrestige(Player: player);		NO
2	EndGameManager	[int] getPlayerResourceCount(player)		NO
Método de alto nivel				
void EndGameManager()				
Diagrama de Colaboración (Opcional)				

Identificador	Descripción de la acción de alto nivel			
Polis-0008	Turnos, donde jugadores realizan sus dos acciones respectivamente.			
Pasos (usar pseudocódigo o similar)				
1. Jugador uno realiza sus acciones.				
2. Jugador dos realiza sus acciones.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	TextModeUI	[void]showAvailableActions(Game g, Player p)	0009	SI
2	TextModeUI	[void]showAvailableActions(Game g, Player p)	0009	SI
Método de alto nivel				
[void]TextModeUI ()				
Diagrama de Colaboración (Opcional)				

Identificador	Descripción de la acción de alto nivel			
Polis-0009	Turnos, donde jugadores realizan sus dos acciones respectivamente.			
Pasos (usar pseudocódigo o similar)				
1. Jugador hace una acción creación, la segunda acción no puede ser ésta. 2. Jugador hace una acción militar, la segunda acción no puede ser ésta. 3. Jugador hace una acción política, la segunda acción no puede ser ésta. 4. Jugador hace una acción pasa turno.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	creatorAction		0010	SI
2	MilitaryAction		0011	SI
3	politicAction		0012	SI
4	EmptyAction	[void] passTurn()		SI
Método de alto nivel				
[void]TextModeUI ()				
Diagrama de Colaboración (Opcional)				

Identificador		Descripción de la acción de alto nivel		
Polis-0010		Acciones de creación, Jugador decide cual realizar.		
Pasos (usar pseudocódigo o similar)				
1 Crear Hoplita 2 Crear Trirreme 3 Crear Próximo 4 Crear Barco de Comercio				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	CreateAction	[boolean] createHoplite(Player owner, Polis polis, Round round)		SI
2	CreateAction	[boolean] createTrirreme(Player owner, Polis polis, Round round)		SI
3	CreateAction	[boolean] createProxenus(Player owner, Polis polis, Round round)		SI
4	CreateAction	[boolean] createTradeBoat(Player owner, Polis polis, Round round)		SI
Método de alto nivel				
[void]creatorAction()				
Diagrama de Colaboración (Opcional)				

Identificador		Descripción de la acción de alto nivel		
Polis-0011		Acciones militares, el Jugador decide cual realizar.		
Pasos (usar pseudocódigo o similar)				
1 Mover Hoplita. 2 Mover Trirreme. 3 Recaudar. 4 Asediar .				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	MilitaryAction	[Boolean] moveHoplite(Player player, Round round,Territory initialPosition, Territory finalPosition, Integer numberOfUnits, Boolean multiMovement)		SI
2	MilitaryAction	[Boolean] moveTrirreme(Round round, Player player, Sea initialSea, Sea finalSea, Integer numberOfUnits, Boolean multiMovement)		SI
3	MilitaryAction	Boolean siegePolis(Player player,Position initialPosition, Polis siegedPolis)		SI
4	MilitaryAction	Boolean plunderTerritory(Player player)		SI
Método de alto nivel				
[void]MilitaryAction()				
Diagrama de Colaboración (Opcional)				

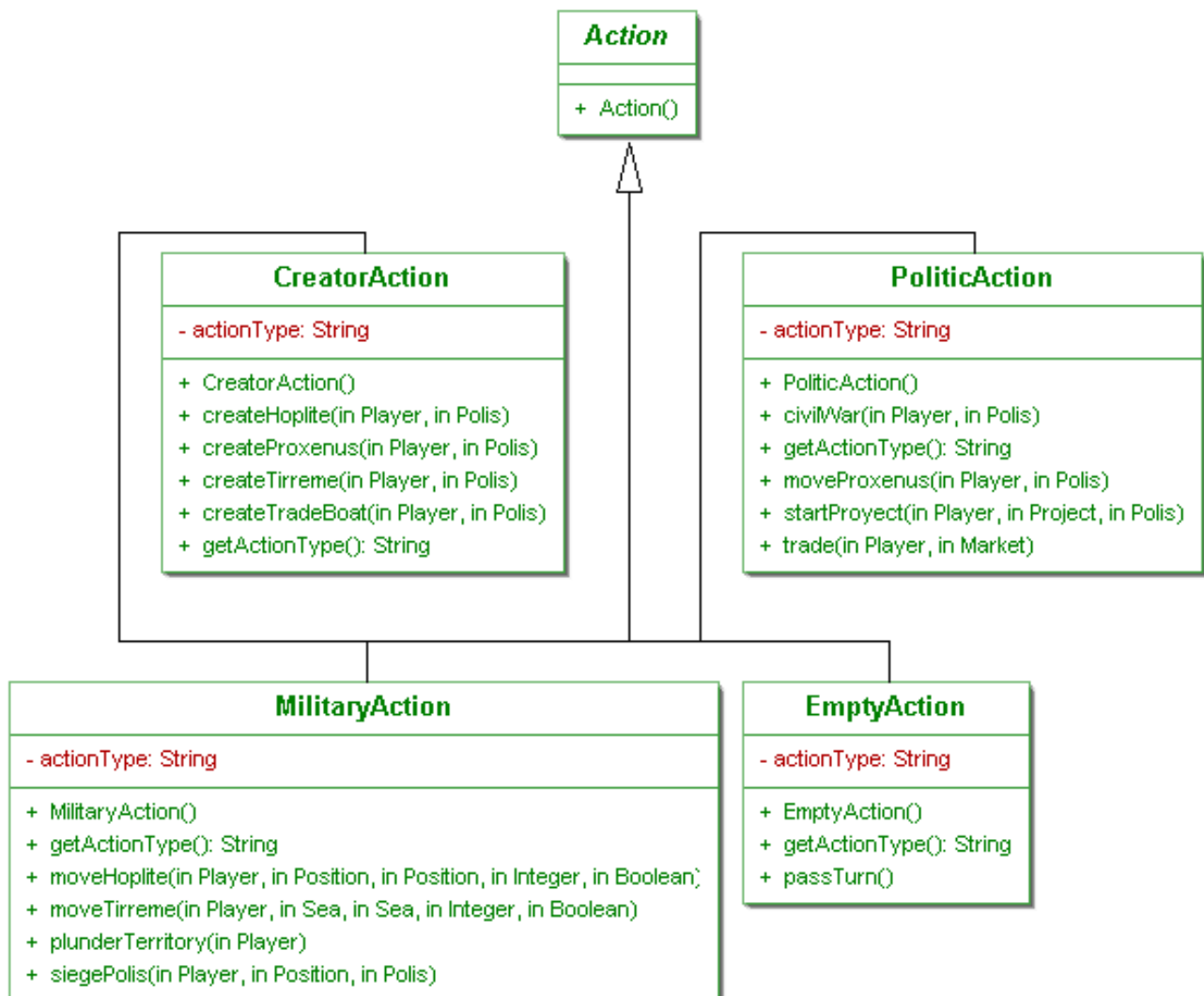
Identificador	Descripción de la acción de alto nivel			
Polis-0012	Acciones políticas, Jugador decide cual realizar			
Pasos (usar pseudocódigo o similar)				
1 Guerra Civil. 2 Mover Próximo. 3 Comenzar proyecto. 4 Comerciar.				
Diagrama de estados (Opcional)				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	PoliticAction	[Boolean] civilWar(Player player, Polis designatedPolis)		SI
2	PoliticAction	[Boolean] moveProxenus(Player player, Polis destination)		SI
3	PoliticAction	[Boolean]startProject(Player player, Project project, Polis polis)		SI
4	PoliticAction	[Boolean] trade(Player player, Round round, MarketChart marketChart, Market market, String resource1, String resource2)		SI
Método de alto nivel				
[void]PoliticAction()				
Diagrama de Colaboración (Opcional)				

Capítulo 7

Diagrama UML de Diseño


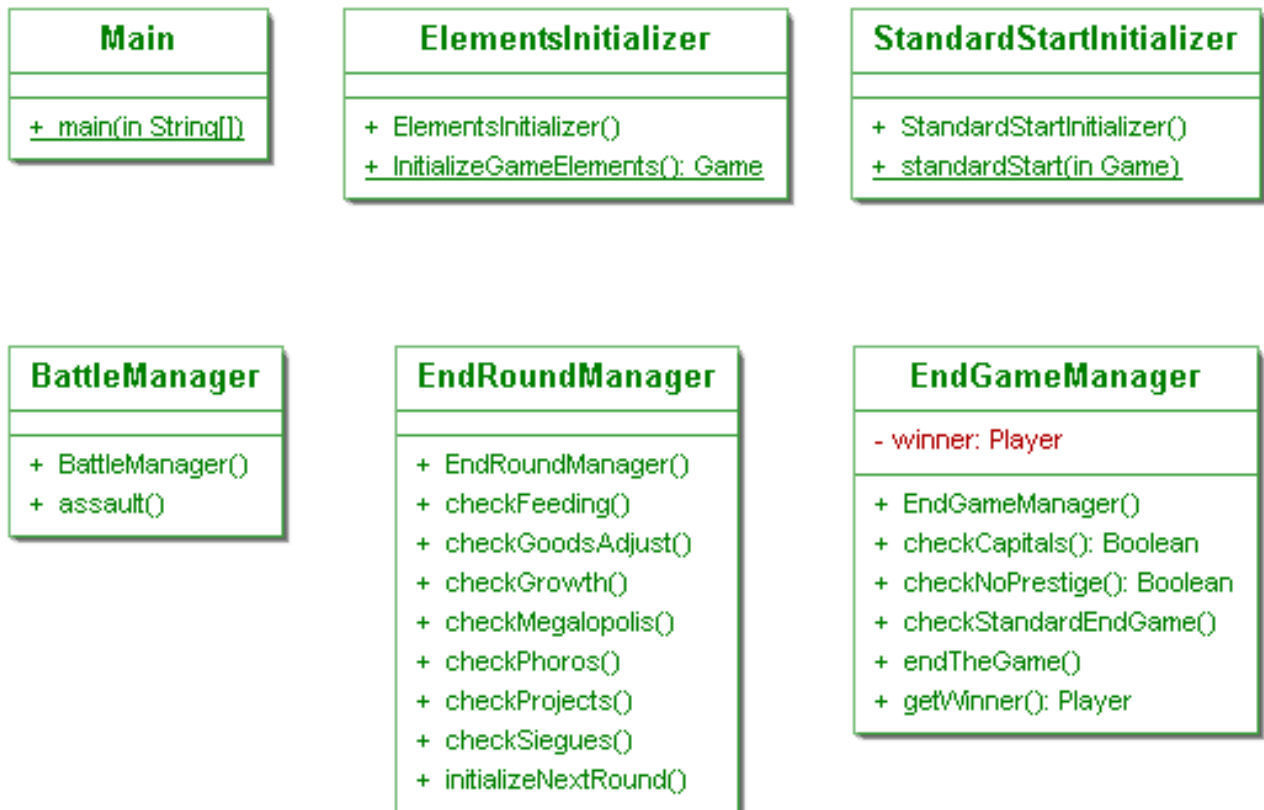
Iteración 3

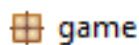
game



 game

Game
<ul style="list-style-type: none"> - athensPlayer: Player - gameEventsRound3: List<GameEvent> - gameEventsRound4: List<GameEvent> - gameEventsRound5a: List<GameEvent> - gameEventsRound5b: List<GameEvent> - gameMarkets: Map<String,Market> - gamePolis: Map<String,Polis> - gameSeas: Map<String,Sea> - gameTerritories: Map<String,Territory> - gameTradeDocks: Map<String,TradeDock> - marketChart: MarketChart - projectList: List<Project> - round: Round - spartaPlayer: Player
<ul style="list-style-type: none"> + Game(in Player, in Player, in Map<String,Territory>, in Map<String,Sea>, in Map<String,TradeDock>, in Map<String,Market>, in Map<String,Polis>, in List<Project>, in List<List<GameEvent>>, in Round, in MarketChart) + getAthensPlayer(): Player + getGameEventsRound3(): List<GameEvent> + getGameEventsRound4(): List<GameEvent> + getGameEventsRound5a(): List<GameEvent> + getGameEventsRound5b(): List<GameEvent> + getGameMarkets(): Map<String,Market> + getGamePolis(): Map<String,Polis> + getGameSeas(): Map<String,Sea> + getGameTerritories(): Map<String,Territory> + getGameTradeDocks(): Map<String,TradeDock> + getMarketChart(): MarketChart + getProjectList(): List<Project> + getRound(): Round + getSpartaPlayer(): Player

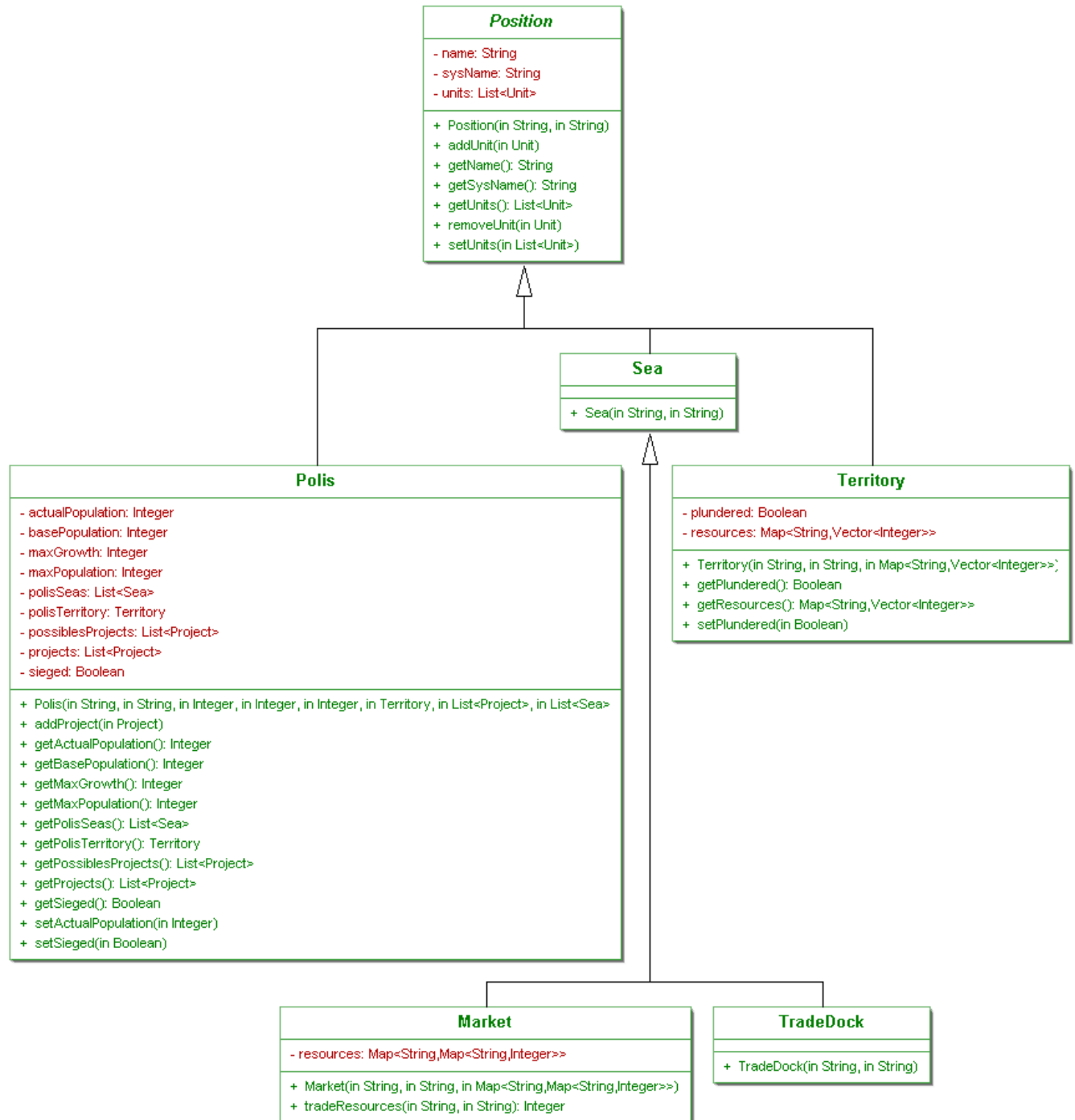
 game

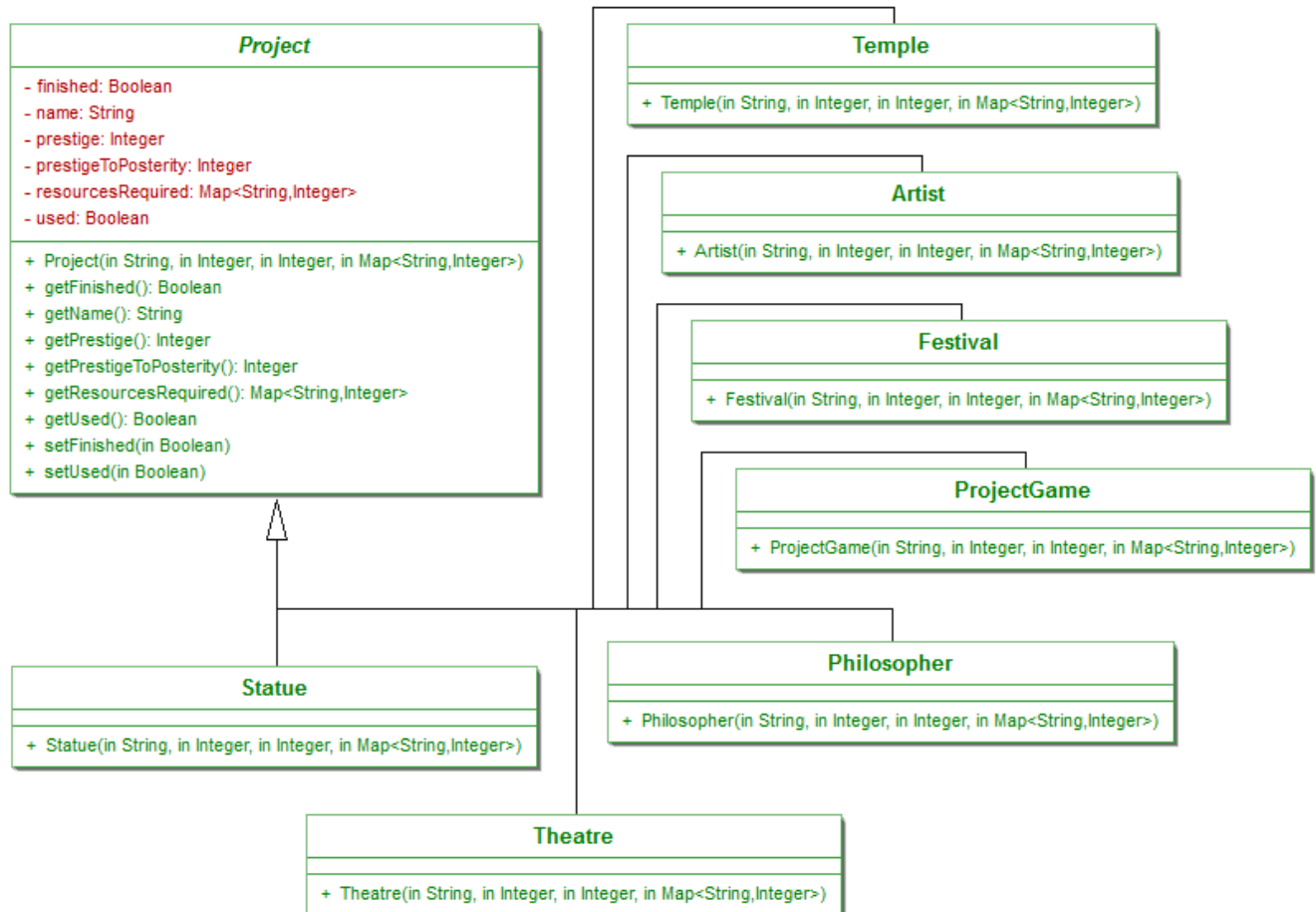



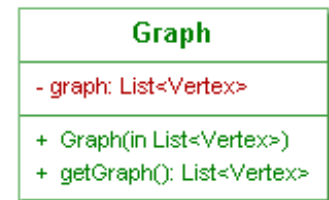
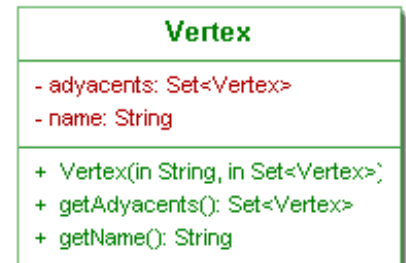
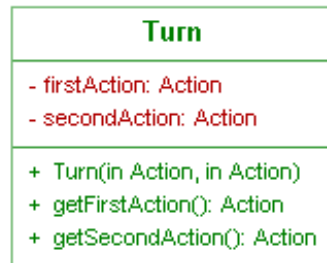
MarketChart
<ul style="list-style-type: none"> - metalPrice: Integer - metalPricePointer: Integer - oilPrice: Integer - oilPricePointer: Integer - round3_prices: Vector<Integer> - round4_prices: Vector<Integer> - round5_prices: Vector<Integer> - winePrice: Integer - winePricePointer: Integer - woodPrice: Integer - woodPricePointer: Integer
<ul style="list-style-type: none"> + MarketChart() + getMetalPrice(): Integer + getOilPrice(): Integer + getWinePrice(): Integer + getWoodPrice(): Integer + moveResourcePrice(in String, in String, in Integer)

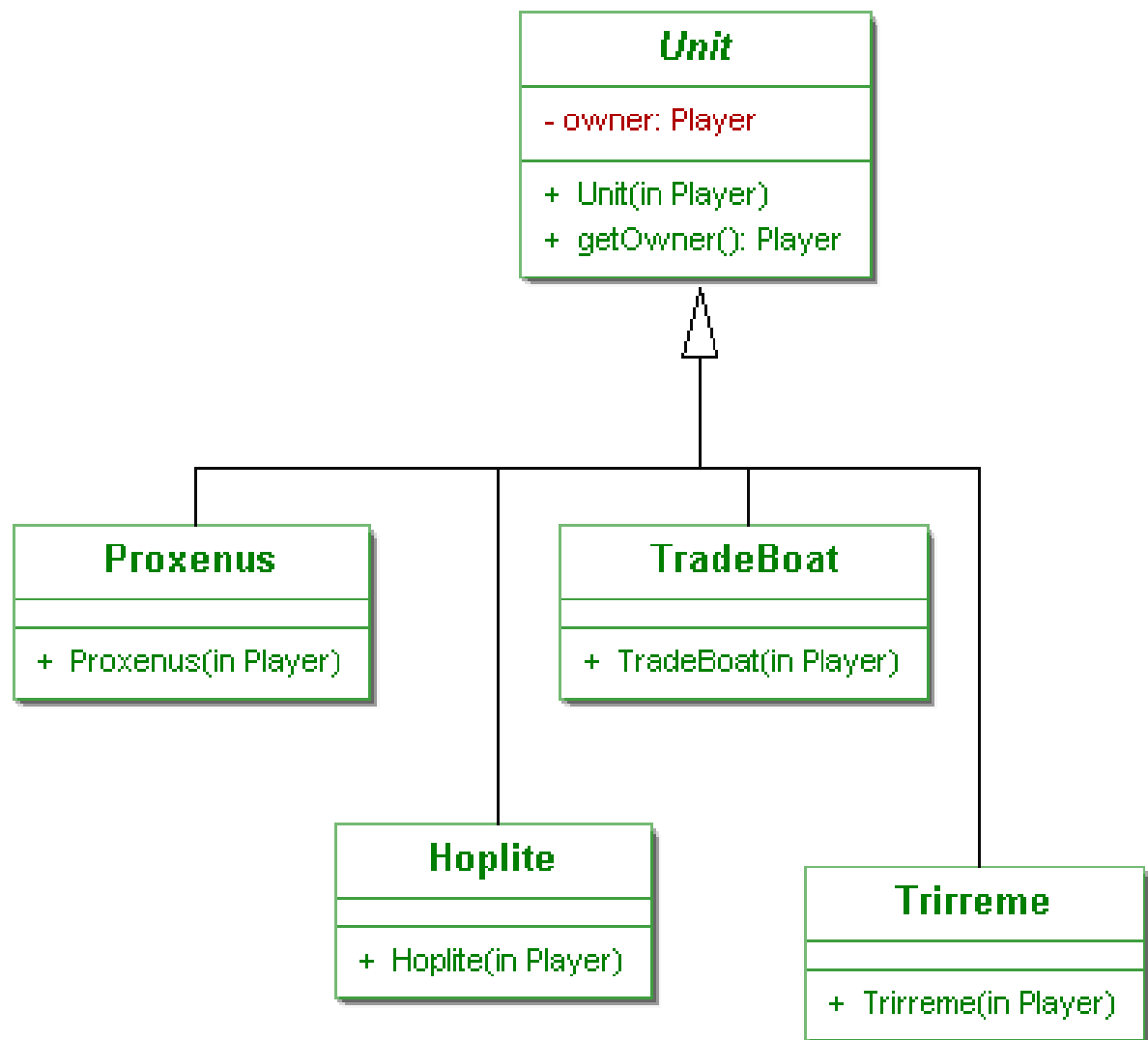
GameEvent
<ul style="list-style-type: none"> - name: String - round: String - sysName: String - text: String
<ul style="list-style-type: none"> + GameEvent(in String, in String, in String, in String) + <u>executeAction(in String)</u> + getName(): String + getRound(): String + getSysName(): String + getText(): String

Player
<ul style="list-style-type: none"> - hasPassedTurn: Boolean - metal: Integer - name: String - oil: Integer - playerPolis: List<Polis> - prestige: Integer - silver: Integer - wheat: Integer - wine: Integer - wood: Integer
<ul style="list-style-type: none"> + Player(in String) + addPolis(in Polis) + getHasPassedTurn(): Boolean + getMetal(): Integer + getName(): String + getOil(): Integer + getPlayerPolis(): List<Polis> + getPrestige(): Integer + getSilver(): Integer + getWheat(): Integer + getWine(): Integer + getWood(): Integer + removePolis(in Polis) + <u>rollTheDice(): Integer</u> + setHasPassedTurn(in Boolean) + setMetal(in Integer) + setName(in String) + setOil(in Integer) + setPrestige(in Integer) + setSilver(in Integer) + setWheat(in Integer) + setWine(in Integer) + setWood(in Integer)

 game


 game


 game

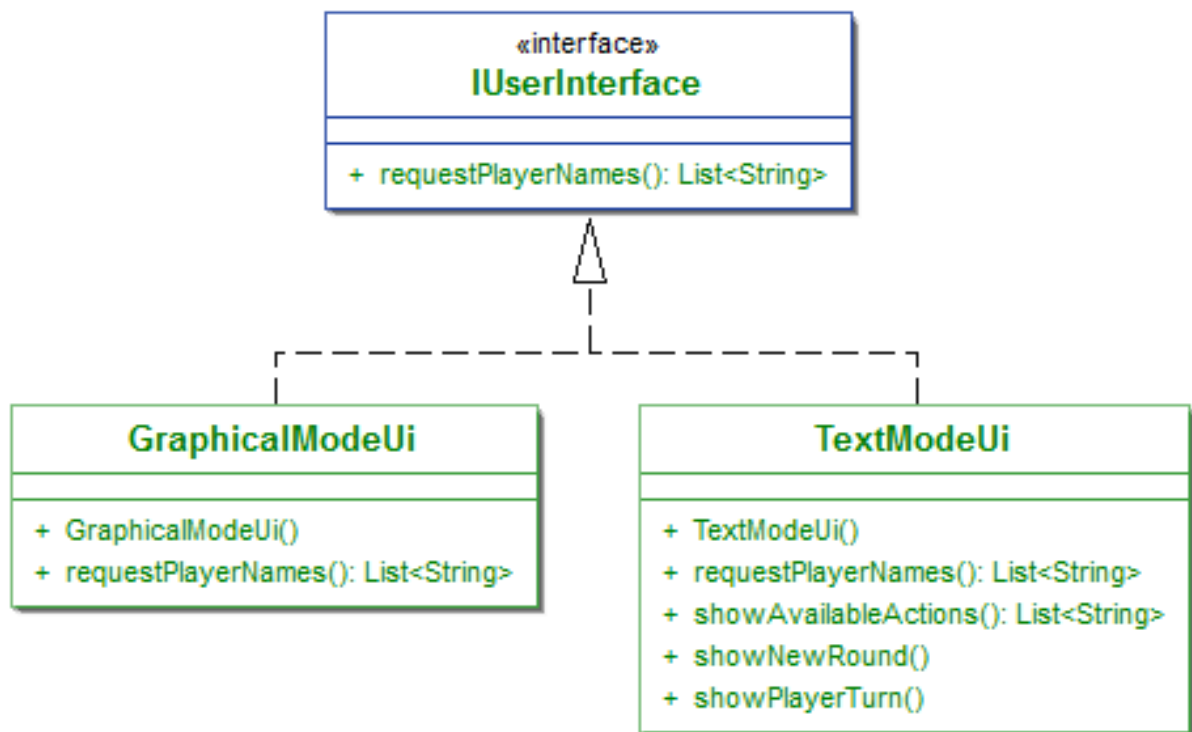
 game



GameConfigurations

- pathOfGameEvents: String
- pathOfMarkets: String
- pathOfPolis: String
- pathOfProjects: String
- pathOfSeas: String
- pathOfTerritories: String
- pathOfTradeDocks: String

- + GameConfigurations()
- + getPathOfGameEvents(): String
- + getPathOfMarkets(): String
- + getPathOfPolis(): String
- + getPathOfProjects(): String
- + getPathOfSeas(): String
- + getPathOfTerritories(): String
- + getPathOfTradeDocks(): String
- + setLanguageToEnglish()
- + setLanguageToSpanish()

 ui

 utils