CopsAndRobbers: dall'architettura al codice

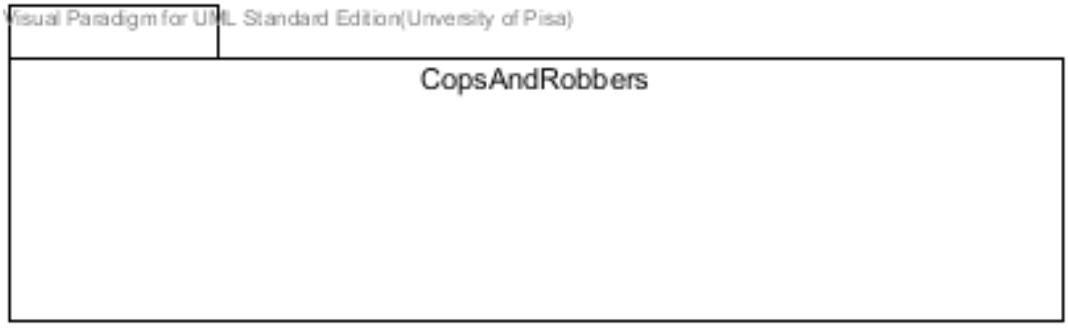
Dipartimento di Informatica - Università di Pisa

C. Montangero

Table of Figures

Class Diagram

Introduzione



Summary

Name	Documentation
Package - CopsAndRobbers	Il sistema CopsAndRobbers gestisce partite al gioco di ruolo descritto in Testo del problema (http://didawiki.cli.di.unipi.it/doku.php/informatica/is-a/start)

Documentation

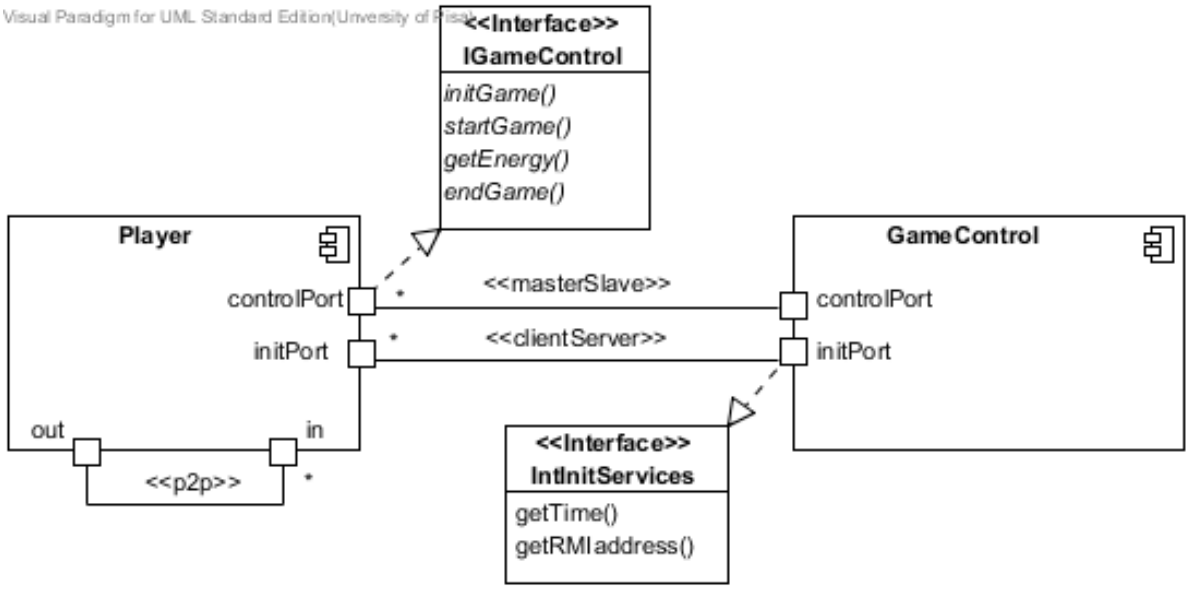
Questo documento presenta alcune viste sull'architettura del sistema software CopsAndRobbers, la progettazione di dettaglio (parziale) della componente Player, e il codice relativo.

Senza pretese di completezza, si vuol mostrare in cosa consista una documentazione di qualità dello sviluppo di un sistema software.

I requisiti del sistema sono disponibili sulla wiki del corso: SR di Cops and robbers (http://didawiki.cli.di.unipi.it/doku.php/informatica/is-a/start).

Component Diagram

ArchitetturaVistaComportamentale



Summary

Name	Documentation
Class - IGameControl	Interfaccia per il controllo del gioco.
Component - Player	Componente che gestisce il gioco da parte di un giocatore.
Component - GameControl	Componente che controlla il gioco: accetta i giocatori, distribuisce mappa e guardie, ecc.
Class - IntInitServices	Definisce i servizi offerti dal controllore per inizializzare i Player. Nella versione più raffinata, offre anche l'operazione per la disconnessione.

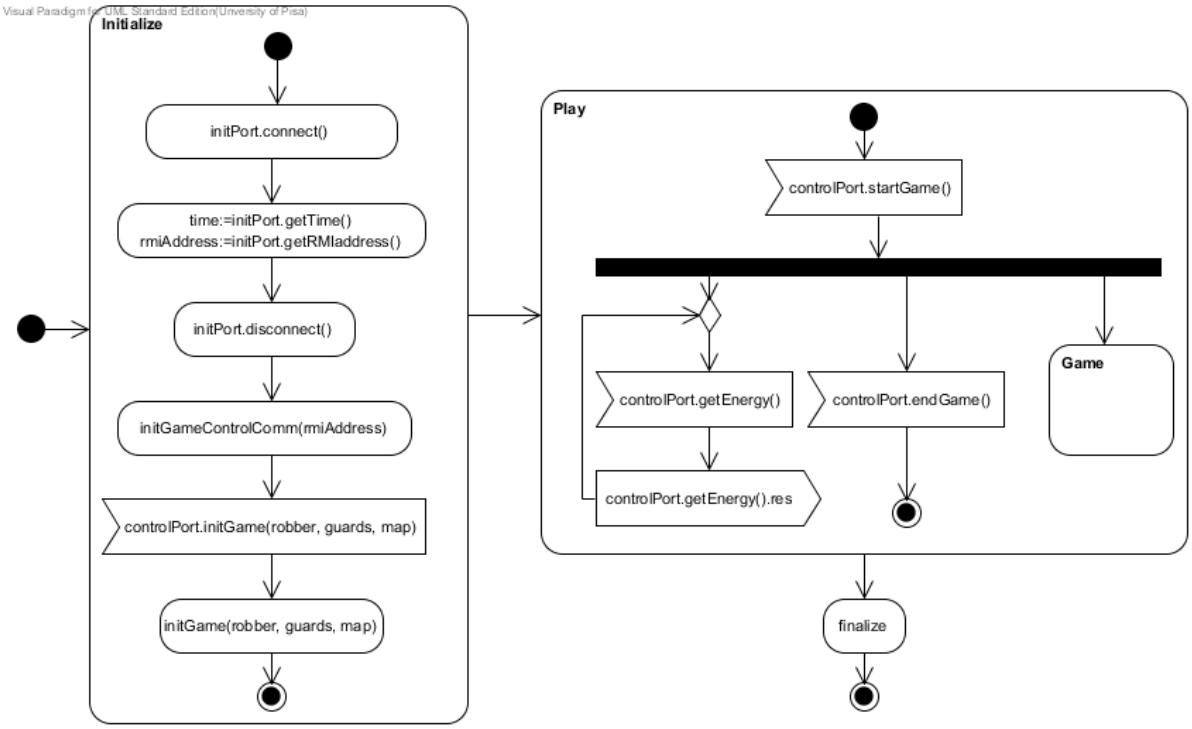
Documentation

Questa vista mostra le due diverse componenti di CopsAndRobbers e i diversi connettori tra di esse. Il connettore <<clientserver>> è usato nella fase iniziale da ogni Player, per ottenere i dati iniziali per sincronizzare gli orologi e inizializzare il connettore <<masterSlave>>, con cui GameControl gestisce il gioco.

Il connettore <<p2p>> non è considerato in questa versione.

Activity Diagram

PlayerActivity



Summary

Name	Documentation
Activity - Initialize	Descrive le azioni necessarie per inizializzare le connessioni tra un Player e il GameControl: lettura dei parametri iniziali, e instaurazione della connessione RMI per il controllo. Da completare con l'inizializzazione delle connessioni p2p tra i Player.
Action - initPort.connect()	La connessione al server avviene sfruttando informazioni ben note, passate al Player all'attivazione.
Action - time:=initPort.getTime() rmiAddress:=initPort.getRMIaddress()	Memorizzazione delle informazioni di inizializzazione, ottenute via initPort.
Action - initPort.disconnect()	La connessione con il server non è più necessaria.
Action - initGameControlComm(rmiAddress)	Con questa operazione interna, Player inizializza la connessione con il controllore del gioco.
Accept Event Action - controlPort.initGame(robber, guards, map)	Dopo l'inizializzazione, Player si mette in attesa dell'ordine di inizializzare la partita, inviato dal controllore del gioco
Action - initGame(robber, guards, map)	map è la mappa del gioco, robber il ladro attribuito a questa istanza di Player, e guards la lista di guardie da controllare.

Activity - Play	Una volta iniziato il gioco, la gestione vera e propria della partita (attività Game, da raffinare), è condotta in parallelo alla gestione degli ordini del controllore (invio del livello di energia di Player e fine partita).
Action - finalize	Chiusura ordinata delle connessioni e delle componenti.

Documentation

Visione semplificata del comportamento del Player: si tratta solo parte dell'inizializzazione e il controllo del gioco.

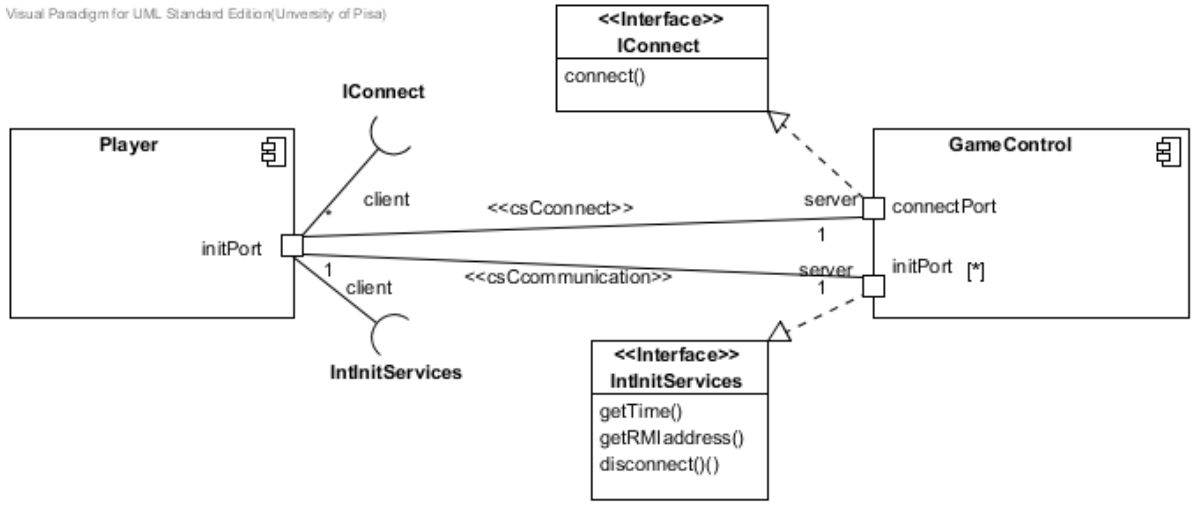
L'attività Initialize stabilisce la connessione con il server che fornisce i dati di inizializzazione, li richiede e si disconnette. Con il secondo si inizializza la connessione RMI su controlPort e si resta in attesa della richiesta di inizializzare il gioco, cosa che si fa (robber è il ladro asseganto al player, guards le guardie, e map la mappa iniziale del gioco).

initGameControlComm e initGame sono operazioni interne di Player.

L'attività Play richiede che il Player serva le richieste di comunicare l'energia al controllore, e di terminare il gioco, in parallelo alla gestione del gioco vero e proprio (non elaborata).

Component Diagram

ArchitetturaVistaComportamentale2



Summary

Name	Documentation
Class - IConnect	Offre l'operazione per la connessione al server.
Component - IConnect	Offre l'operazione per la connessione al server.
Component - Player	Componente che gestisce il gioco da parte di un giocatore.
Component - GameControl	Componente che controlla il gioco: accetta i giocatori, distribuisce mappa e guardie, ecc.
StructuredInterface - IntInitServices	Definisce i servizi offerti dal controllore per inizializzare i Player. Nella versione più raffinata, offre anche l'operazione per la disconnessione.
Class - IntInitServices	Definisce i servizi offerti dal controllore per inizializzare i Player. Nella versione più raffinata, offre anche l'operazione per la disconnessione.

Documentation

Questa vista mostra un raffinamento della rappresentazione convenzionale del connettore <<clientServer>> usato nel diagramma ArchitetturaVistaCOmportamentale.

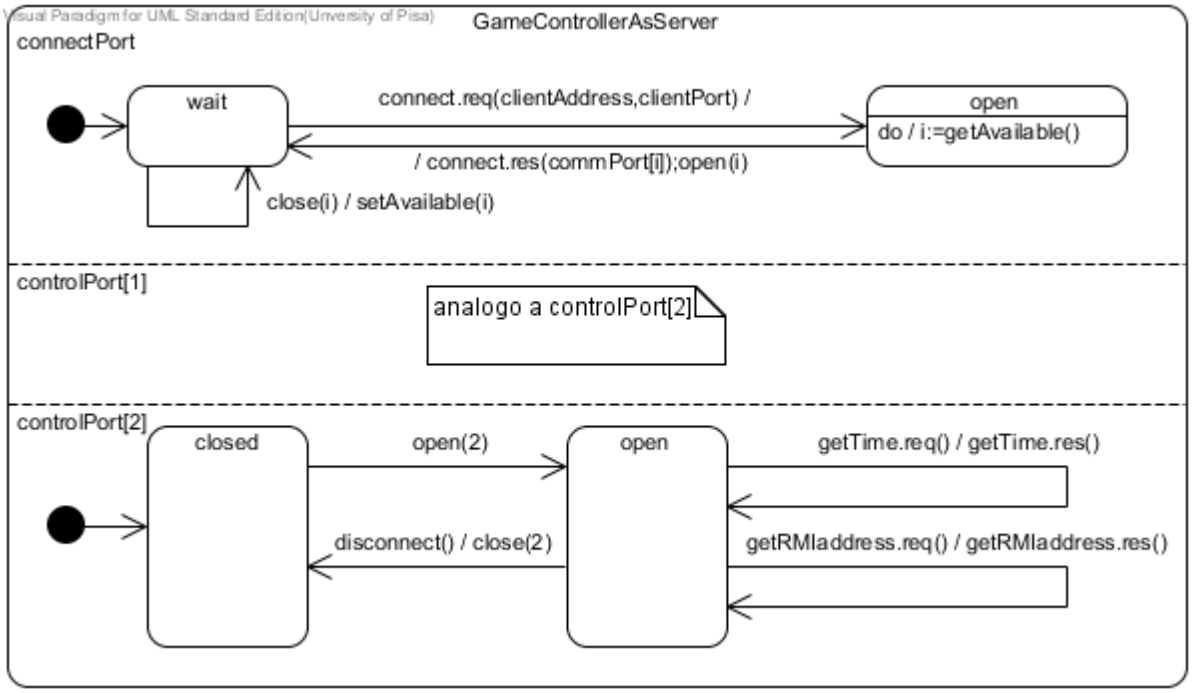
Si mostra come il protocollo abiia due passi: il primo descritto dal connettore <<csConnect>> fa riferimento a un porto "ben noto" del server (connectPort), a cui il cliente si rivolge per instaurare la connessione (operazione connect). Per semplicità, non sono wui indicati i parametri che portano al server l'informazione circa il cliente che permette di instaurare la connessione.

Il secondo passo del protocollo, retto dal connettore <<csCommunication>>, oltre alle operazioni relative ai servizi offerti da server, offer anche l'operazione disconnect, che il cliente usa per terminare la sessione.

Il diagramma GameControlBehaviourAsServer esemplifica l'uso di queste operazioni. Si riveda anche il diagramma PlayerActivity.

State Machine Diagram

GameControlBehaviourAsServer



Summary

Name	Documentation
Region - connectPort	Questa regione descrive il comportamento del porto corrispondente all'indirizzo ben noto, cui i clienti si rivolgono per connettersi al server.
State - wait	Attesa delle richieste di connessione e sconnessione al server: quest'ultima è mediata dal'evento interno close(i).
State - open	In questo stato attività, viene scelto un nodo non connesso. Al termine, la regione torna nello stato di attesa di altre richieste di connessione. Alla transizione è associata la comunicazione al cliente del porto da usare per le richieste di servizio e l'apertura dello stesso.
Region - controlPort[2]	Questa regione descrive il comportamento del singolo porto di servizio. Una volta aperto, il server resta in attesa di richieste che serve immediatamente.
State - open	Il modo con cui sono calcolati i valori da restituire è lasciato alla progettazione di dettaglio.
Note - N/A	analogo a controlPort[2]

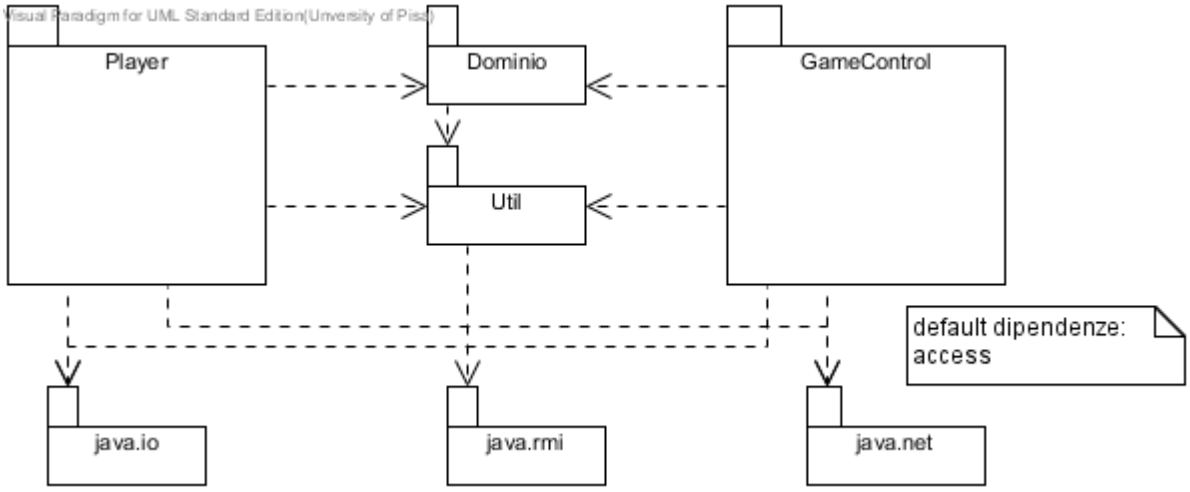
Documentation

Questa macchina mostra come avviene la gestione dei clienti da parte di GameControl, che nella fase iniziale si comporta da server.

Data l'impossibilità di parametrizzare il numero di regioni, se ne mostrano solo 2, identiche a meno di un indice. Analogamente, quella che qui è mostrata come operazion di open della connessione, in pratica potrà eaaere realizzata con la creazione dinamica della connesione stessa.

Package Diagram

ArchitetturaVistaStrutturale



Summary

Name	Documentation
Package - Player	Contiene le classi che realizzano il comportamento della componente Player.
Package - Dominio	Contiene le classi che reificano il dominio, ossia che rappresentano tutte le entità necessarie per mantenere lo stato del gioco.
Package - GameControl	contiene le classi che realizzano il comportamento di GameControl.
Package - Util	Contiene classi che sono usate da classi in più package, ma che non appartengono al dominio.
Note - N/A	default dipendenze: access

Documentation

Questa vista riguarda la struttura in package prevista pel codice java.

Sono messe in evidenza anche le librerie Java utilizzate per le connessioni tra le componenti (rmi e net) e per realizzare le interfacce verso gli utenti (java.io, dato che non c'è un requisito sull'interfaccia utente).