**UML dijagrami**

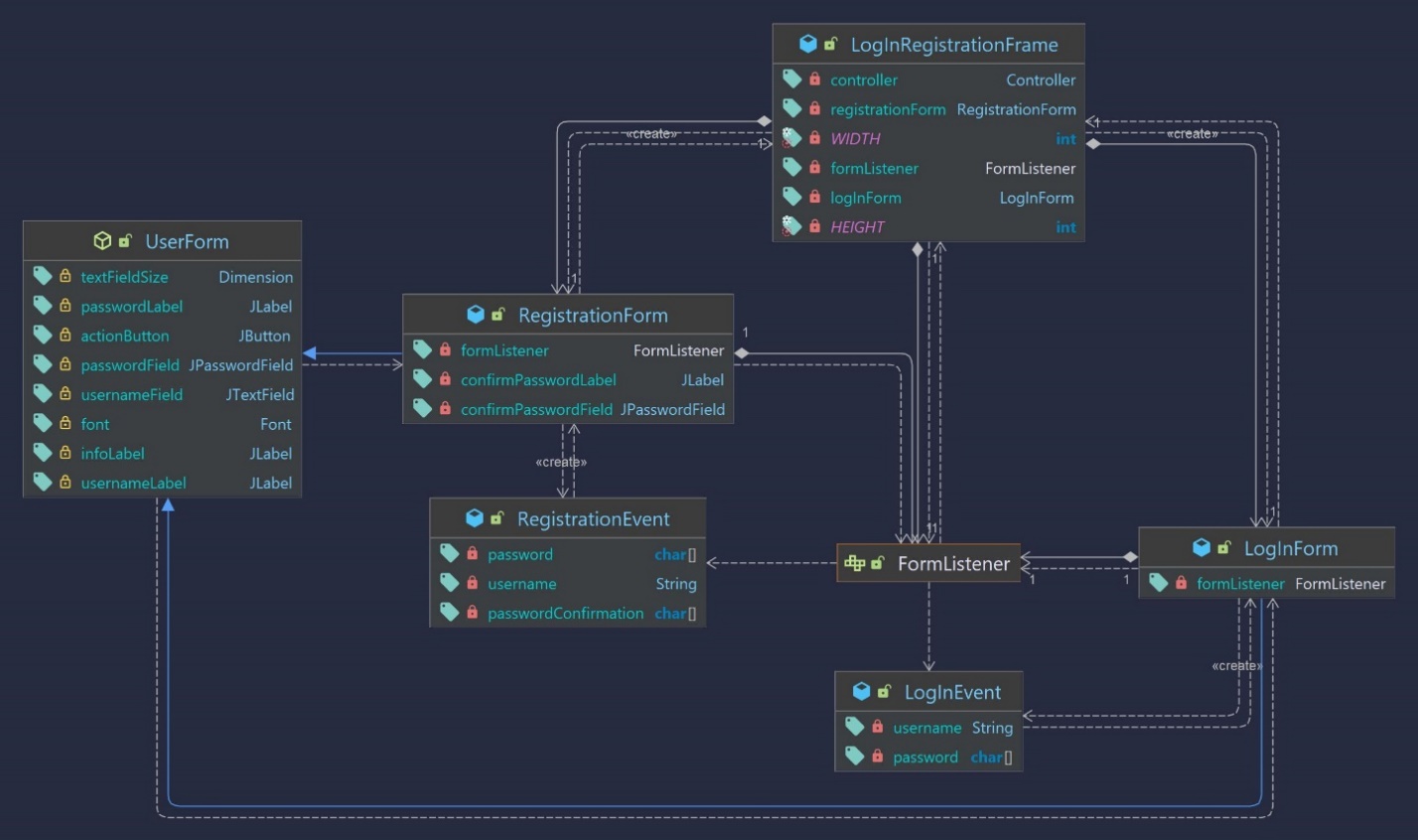
Sadržaj

[memory\_game\_client paket 2](#_Toc97494816)

[memory\_game\_server paket 6](#_Toc97494817)

# memory\_game\_client paket

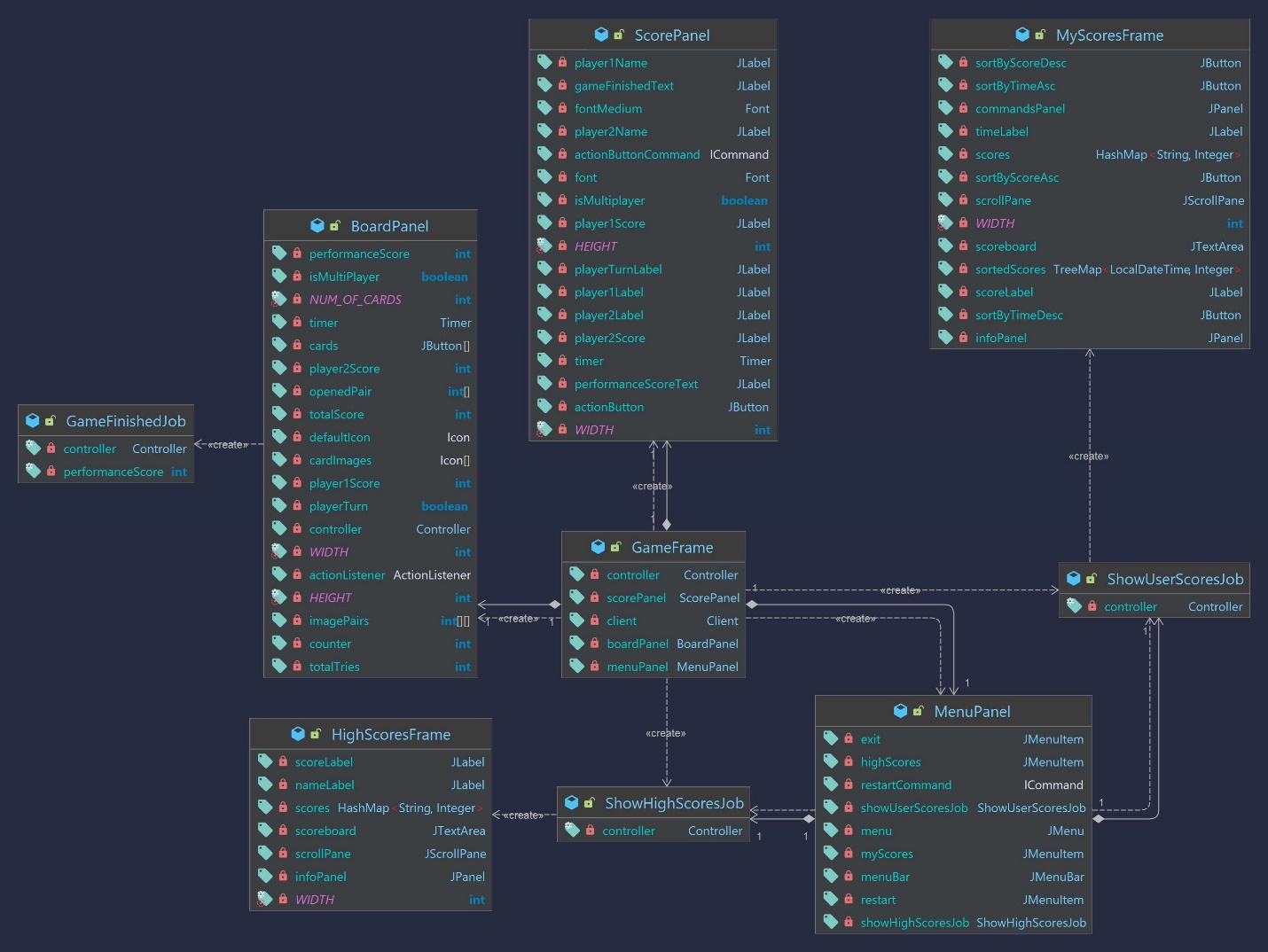
Sljedeća slika prikazuje UML dijagram za klase unutar view.logInRegistration paketa.



LogInRegistrationFrame je prvi prozor koji se otvori pri pokretanju klijentske aplikacije. U njemu se nalaze 2 panela (LogInForm i RegistrationForm). Obe klase proširuju apstraktnu klasu UserForm.

Korisnik unosi podatke te se oni preko FormListenera pohranjuju u RegistrationEvent ili LogInEvent objekt. Taj se objekt tada u LogInRegistrationFrame-u preko kontrolera šalje bazi podataka na obradu. Kontroler vraća odgovor od baze podataka i korisnika se informira o konačnom ishodu.

Na sljedećoj slici je prikazan UML dijagram za klase unutar view i view.jobs paketa.



GameFrame je glavni prozor klijentske aplikacije koji se otvara nakon što se korisnik prijavi u aplikaciju i odabere game mode. U njemu se nalaze 3 panela (BoardPanel, ScorePanel i MenuPanel). MenuPanel sadrži JMenuBar u kojem korisnik ima JMenu s 4 itema, što mu nudi 4 opcije:

* ponovno pokretanje igre
* otvaranje HighScoresFrame prozora
* otvaranje MyScoresFrame prozora
* izlazak iz aplikacije

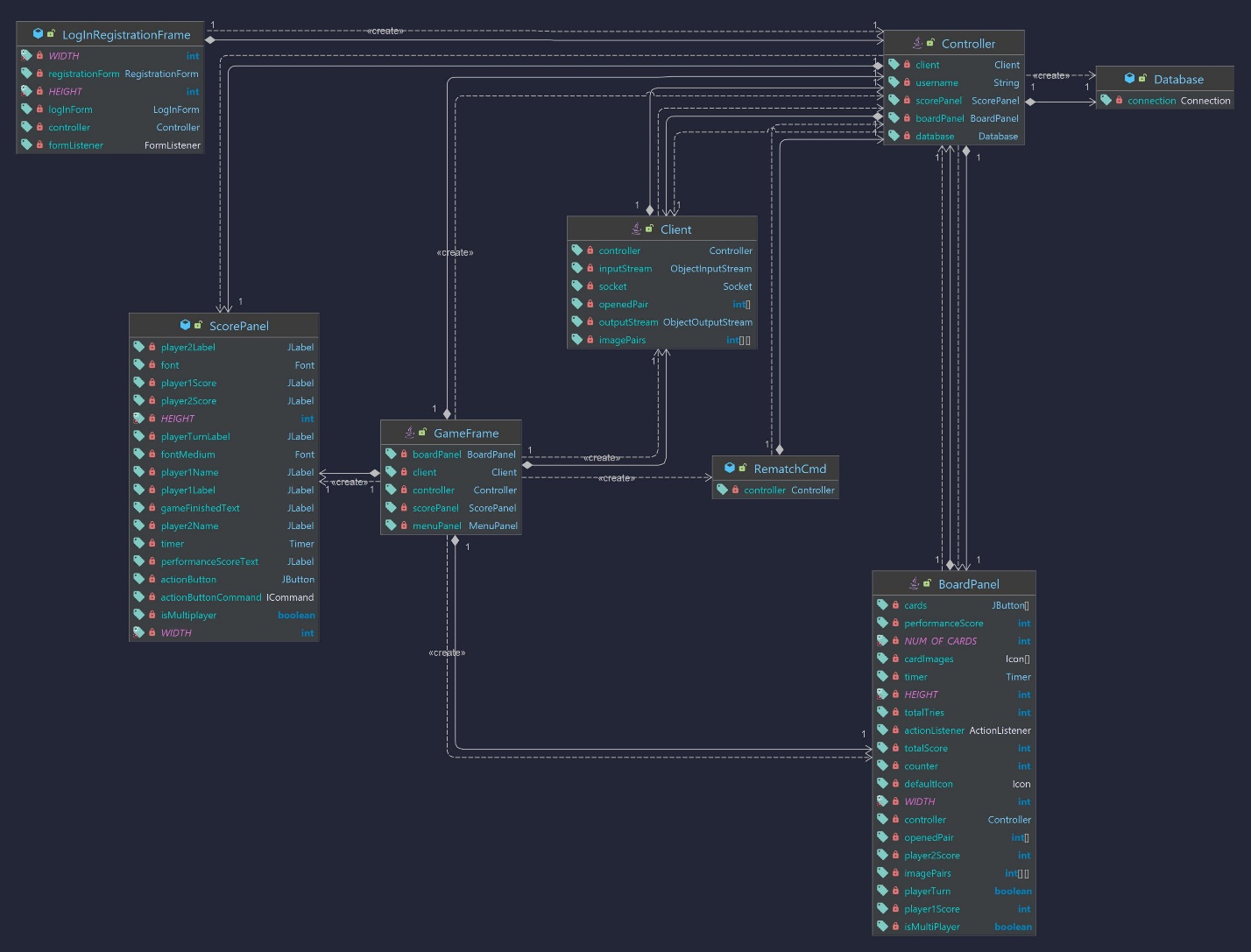
HighScoresFrame i MyScoresFrame prozori se otvaraju preko ShowHighScoresJob i ShowUserScoresJob. Oba Job-a implementiraju Runnable sučelje te u sebi pokreću novu nit koja preuzima posao otvaranja novog JFrame prozora. Također, pored otvaranja novog prozora, imaju i zadatak da se spoje na bazu podataka preko kontrolera kojega u sebi sadrže kompozicijski te da iz baze podataka povuku podatke koje će prikazati u otvorenom prozoru.

GameFinishedJob također implementira Runnable. On se pokreće kada korisnik završi igru u single player modu. Pri pokretanju Job-a pokreće se nova nit koja preko kontrolera kojeg GameFinishedJob sadrži u sebi kompozicijski pokušava uspostaviti vezu sa bazom podataka kako bi spremila korisnikov rezultat u bazu. Također, GameFinishedJob ispisuje korisnikov rezultat na ScorePanel-u.

BoardPanel sadrži sve vizualne komponente za prikaz igraće ploče i karata te svu logiku za procesiranje radnji koje korisnik odrađuje na igraćoj ploči. Preko kontrolera šalje informacije o promjenama na ploči za ispis na ScorePanel-u. Također, u multiplayer modu preko kontrolera šalje korisnikove poteze protivničkom igraču.

ScorePanel ispisuje korisnikovo ime, bodove i konačni rezultat nakon završetka igre. U multiplayer modu ispisuje također protivnikovo ime, bodove te čiji je red za igru. Na kraju ispisuje ishod igre. ScorePanel sadrži i gumb za reset igre čiji pritisak pokreće RestartGameCmd (single player) ili RematchGameCmd (multiplayer). Ove komande koristi i MenuPanel za izvršavanje istih operacija. RestartGameCmd i RematchGameCmd klase te ICommand sučelje koje obe klase implementiraju, nisu prikazani na dijagramu zbog bolje preglednosti.

Sljedeća slika prikazuje UML dijagram za klase unutar controller i model paketa te klase iz view paketa koje imaju asocijaciju s klasom Controller.



Controller kompozicijski sadrži klasu Database koja sadrži sve metode za interakciju s bazom podataka.

LogInRegistrationFrame preko kontrolera komunicira s bazom podataka kada korisnik vrši registraciju i login.

Client klasa koja komunicira s multiplayer serverom, tj. ClientHandler klasom na serveru, sve primljene podatke preko Controller-a usmjerava u view paket.

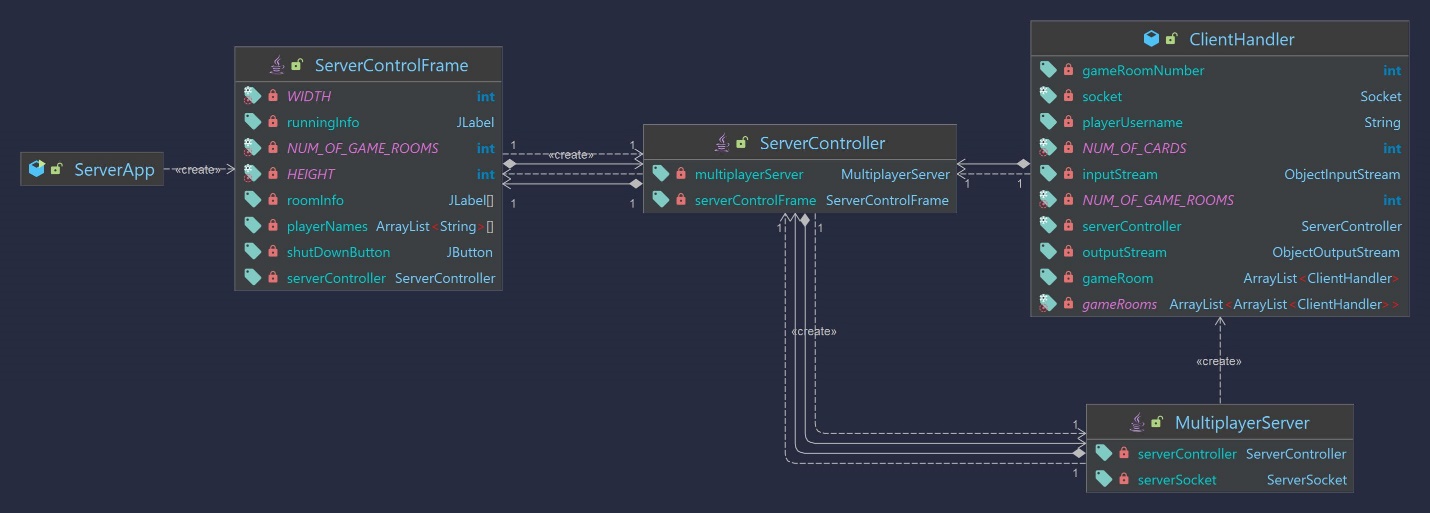
Controller koristi Client klasu kako bi slao informacije o akcijama klijenta multiplayer serveru, odnosno ClientHandler-u.

BoardPanel preko Controller-a šalje informacije o izmjenama na igraćoj ploči ScorePanel-u za ispis.

GameFrame pri inicijalizaciji aplikacije postavlja kontroleru polja client, scorePanel i boardPanel kako bi im kontroler imao pristup. Također, GameFrame postavlja controller polje klasi RematchCmd koja preko controllera šalje rematch request multiplayer serveru.

# memory\_game\_server paket

Sljedeća slika prikazuje sve klase iz memory\_game\_server paketa.



ServerControlFrame je glavni prozor koji se otvara pri pokretanju servera. U njemu se vide informacije o popunjenosti servera. Također, server se gasi pritiskom na shutDownButton ili zatvaranjem prozora.

MultiplayerServer klasa u sebi kompozicijski sadrži serverSocket koji se inicijalizira u samoj klasi. U navedenoj klasi se nalazi kod za pokretanje servera metodom startServer() kojom se serverSocket stavlja u stanje slušanja dolaznih pokušaja spajanja na server.

Pri uspješnom spajanju klijenta, stvara se nova nit kojoj se za izvršavanje daje objekt klase ClientHandler koja implementira Runnable. Objekt te klase preuzima komunikaciju s klijentom i postavlja sebe u prvu slobodnu gameRoom. gameRoom je ArrayLista koja sadrži dva ClientHandler objekta. Svaki ClientHandler komunicira s klijentom koji mu je dodijeljen. Pored toga, pošto ovi objekti u sebi imaju referencu na gameRoom u kojem se nalaze, to im omogućuje da šalju i primaju informacije klijentu koji je dodijeljen ClientHandler-u koji se nalazi u gameRoom-u u kojem se oni sami nalaze. Pojednostavljeno, svaki ClientHandler može komunicirati s klijentom od ostalih ClientHandler-a koji se nalaze u istom gameRoom-u. Pošto su gameRoom-ovi ograničeni na kapacitet od dva ClientHandlera, svaki gameRoom host-a jedan 1v1 match.

ServerController služi ClientHandler-ima za slanje informacija u spajanju/odspajanju korisnika na server kako bi to bilo prikazano u glavnom ServerControlFrame prozoru.

ClientHandler preko klase Client u paketu memory\_game\_client komunicira s klijentskom aplikacijom.