Završna faza

Uno Card Game

AIPS projekat

Dejan Randjelovic 15843

Nikola Popović 15826

U ovom delu cemo vam demonstrirati projektne obrasce koje smo koristili za implementaciju projekta **Uno Card Game**.

Obrasci koje smo koristili su: **Singleton**, **Proxy** i **Observer**.

# Singleton

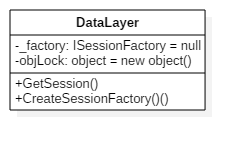
**Definicija**:

Obezbeđuje da određena klasa ima maksimalno jedan aktivni objekat, i obezbeđuje da on bude dostupan.

Definiše operaciju Instance koja obezbeđuje korisniku pristup jedinom kreiranom objektu.

Definiše konstruktor koji je odgovoran za kreiranje i održavanje svog jedinog objekta.



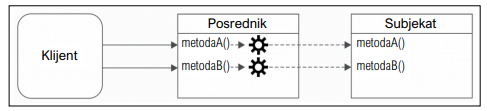


Postoji privatni atribut\_factory koji odgovara singleton promenljivoj kojoj ograničavamo broj instanci na jednu.Funkcija GetSession bi odgovarala funkciji Instance i ona prilikom poziva proverava da li je referenca \_factory null i ukoliko jeste poziva CreateSessionFactory privatnu funkciju kojoj samo ona može da pristupi i kreira instancu tipa SessionFactory i omogući da \_facotry pokazuje na nju a zatim vrati \_factory. CreateSessionFactory odgovara funkciji Singleton i ako \_factory nije null on ga samo vrati i na ovaj način ne dožvoljava ponovno kreiranje Session-a jer je to skupa operacija.

# Proxy

**Definicija**:

Posrednik (engl. proxy) jeste objekat koji kontroliše pristup drugom objektu, koji se zove subjekat (engl. subject). Posrednik i subjekat imaju identičan interfejs, što nam omogućava da transparentno jedan zamenjujemo drugim; u stvari, alternativno ime za ovaj obrazac je surogat, zamenik (engl. surrogate). Posrednik presreće sve ili samo neke operacije koje treba da se izvrše nad subjektom, čime proširuje ili dopunjuje njegovo ponašanje. Naredna slika prikazuje to predstavlja u obliku dijagrama:



U našem projektu, koristimo Proxy klasu čiji objekat se koristi za instanciranje objekta Modell klase i za poziv odgovarajućih metoda tog objekta. Modell klasa implementira interfejs IModel koji sadrži metode za pristup i rad sa bazom podataka.

IModel.cs:

public interface IModel

{

Player returnPlayer(string username);

Game returnGame(int id);

IList<Game> returnGames();

bool addPlayer(string username, string password);

Game checkGameName(string name, string maxcount, string username);

Game joinGame(string username, string name);

void exitGame(string name, string username);

void winCountInc(string name);

}

Proxy objekat se instancira i koristi u main klasi MasterServer-a.

Program.cs:

//

Proxy proxy = new Proxy();

//

Upotreba proxy:

//

Player i = proxy.ReturnPlayer(p[1]);

//

Proxy.cs (za navedeni primer):

//

private IModel model;

public Proxy()

{

model = new Modell();

}

public Player ReturnPlayer(string username)

{

return model.returnPlayer(username);

}

//

Modell.cs:

//

public Player returnPlayer(string username)

{

ISession s = DataLayer.GetSession();

IQuery q = s.CreateQuery("from Player p where p.username=:username");

q.SetString("username", username);

Player p = q.UniqueResult<Player>();

s.Close();

return p;

}

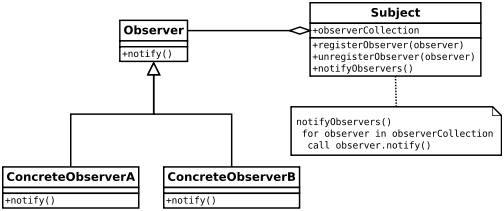
//

# Observer

**Definicija**:

Definiše jedan - na - više zavisnost između objekata tako da kada jedan objekat promeni svoje stanje, svi objekti koji zavise od njega se obaveštavaju o tome i automatski ažuriraju.

Održava konzistentnost između povezanih objekata.



U našem projektu, akcije jednog igrača utiču na tok igre i druge igrače (promena boje na terenu, promena poteza, igrač napušta igru pre nego igra da počne ili dok je igra u toku, igrač nema više nijednu kartu u ruci itd.). Potrebno je obavestiti ostale igrače (klijente) o promeni što se postiže na klijent strani korišćenjem StartListening() i GameStartListening() funkcija, a na server strani odgovarajućim odgovorima. GameStartListening() funkcija se odvija na posebnom thread-u, i služi za automatsko update-ovanje liste igrača pre nego igra da krene. Funkcija je aktivna dok je bool vrednost gameStart jednaka true. Uspostavlja konekciju sa serverom putem exchange kanala ,,GameStartCall’’ i routing key-a ,,gameStart’’ ukoliko je broj igrača veći od 1 (nema nikog da obavesti) i ukoliko igra nije počela (GameStatus objekat GameLogic klase je null). Odgovor klijent aplikacije primaju preko ,,GameStartResponse’’ exchange kanala i routingKey property-ja koji je jednak ,,game *nazivIgre*’’ čime se vrši update trenutne liste igrača. Kada u igri broj trenutnih igrača bude jednak broju maksimalnih igrača igra počinje, kreira se GameStatus objekat kojim se vrši sinhronizacija stanja igre izmedju klijenata preko funkcije StartListening(). StartListening() se odvija na posebnom thread-u i osluškuje kanal ,,GameStatusResponse’’ sa routingKey ,,game *idIgre*’’. U toku igre ovde se vrši update GameForm forme (lista igrača, prikaz ko je na potezu...). Koristi se GameLogic klasa, vezana za klijent stranu, i GameStatus klasa koja se šalje preko servera i update-uje GameLogic objekat. GameLogic objekat sadrži instancu GameStatus klase.

Funkcija GameStartListening() u GameForm.cs:

Task.Run(() =>

{

//

channel.ExchangeDeclare(exchange: "gameStartResponse", type: "direct");

var queueName = channel.QueueDeclare().QueueName;

var routing = "game" + g.name.ToString();

channel.QueueBind(queue: queueName, exchange: "gameStartResponse",

routingKey: routing);

//

consumer.Received += (model, ea) =>

{

var body1 = ea.Body;

var message1 = Encoding.UTF8.GetString(body1);

messageR = message1;

var game = JsonConvert.DeserializeObject<Game>(messageR, settings);

this.g = game;

ClearList();

foreach (Player player in this.g.players)

{

ListViewItem item = new ListViewItem(player.username);

item.SubItems.Add("");

AddPlayerToList(item);

}

};

channel.BasicConsume(queue: queueName, autoAck: true, consumer: consumer);

if (g.currentPlayerCount > 1 && gameLogic.ReturnGameStatus() == null)

{

var message = g.id;

var body = Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(message, settings));

channel.BasicPublish(exchange: "gameStartCall",

routingKey: "gameStart",

basicProperties: null,

body: body);

}

while (gameStarted)

{

System.Threading.Thread.Sleep(1000);

}

//

Server strana MasterServer.Program.cs:

int s1 = Int32.Parse(message);

Game game = proxy.ReturnGame(s1);

var routing = "game" + game.name.ToString();

var body1 = Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(game, settings));

channel.BasicPublish(exchange: "gameStartResponse",routingKey: routing,

basicProperties: null,

body: body1);

Funkcija StartListening():

Task.Run(() =>

{

//

channel.ExchangeDeclare(exchange: "gameStatusResponse", type: "direct");

var queueName = channel.QueueDeclare().QueueName;

var routing = "game" + g.id.ToString();

channel.QueueBind(queue: queueName,

exchange: "gameStatusResponse",

routingKey: routing);

//

consumer.Received += (model, ea) =>

{

MakeVisible();

var body1 = ea.Body;

var message1 = Encoding.UTF8.GetString(body1);

messageR = message1;

var gs = JsonConvert.DeserializeObject<GameStatus>(messageR,settings);

gameLogic.GetStatus(gs);

this.players = gs.playerCards;

ClearList();

this.listPlayers.Items.Clear();

foreach (PlayerCards p in players)

{

ListViewItem item = new ListViewItem(p.Name);

item.SubItems.Add(p.Cards.Count.ToString());

AddPlayerToList(item);

}

if (!gameLogic.ReturnWinner().Equals(""))

{

SetWinnerLabel();

SetOnTurnLabel(gameLogic.ReturnWinner());

HideEndTurnButton();

this.finish = true;

}

else

{

SetOnTurnLabel(gameLogic.GetOnTurn());

this.Invoke(new Action(() => RefreshCards()));

}

SetOnStatus();

};

channel.BasicConsume(queue: queueName,

autoAck: true,

consumer: consumer);

if (g.currentPlayerCount == g.maxPlayerCount)

{

var message = gameLogic.CreateStatus();

var body = Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(message, settings));

channel.BasicPublish(exchange: "gameStatusCall",

routingKey: "gameStatus",

basicProperties: null,

body: body);

}

while (isListening)

{

System.Threading.Thread.Sleep(1000);

}

//

Server strana:

var s1 = JsonConvert.DeserializeObject<GameStatus>(message, settings);

//provera broja igrača i da li imamo pobednika

var routing = "game" + s1.gameId.ToString();

var body1 = Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(s1, settings));

channel.BasicPublish(exchange: "gameStatusResponse",routingKey: routing,

basicProperties: null,

body: body1);