

Informatikos fakultetas

**T120B165 Saityno taikomųjų programų projektavimas**

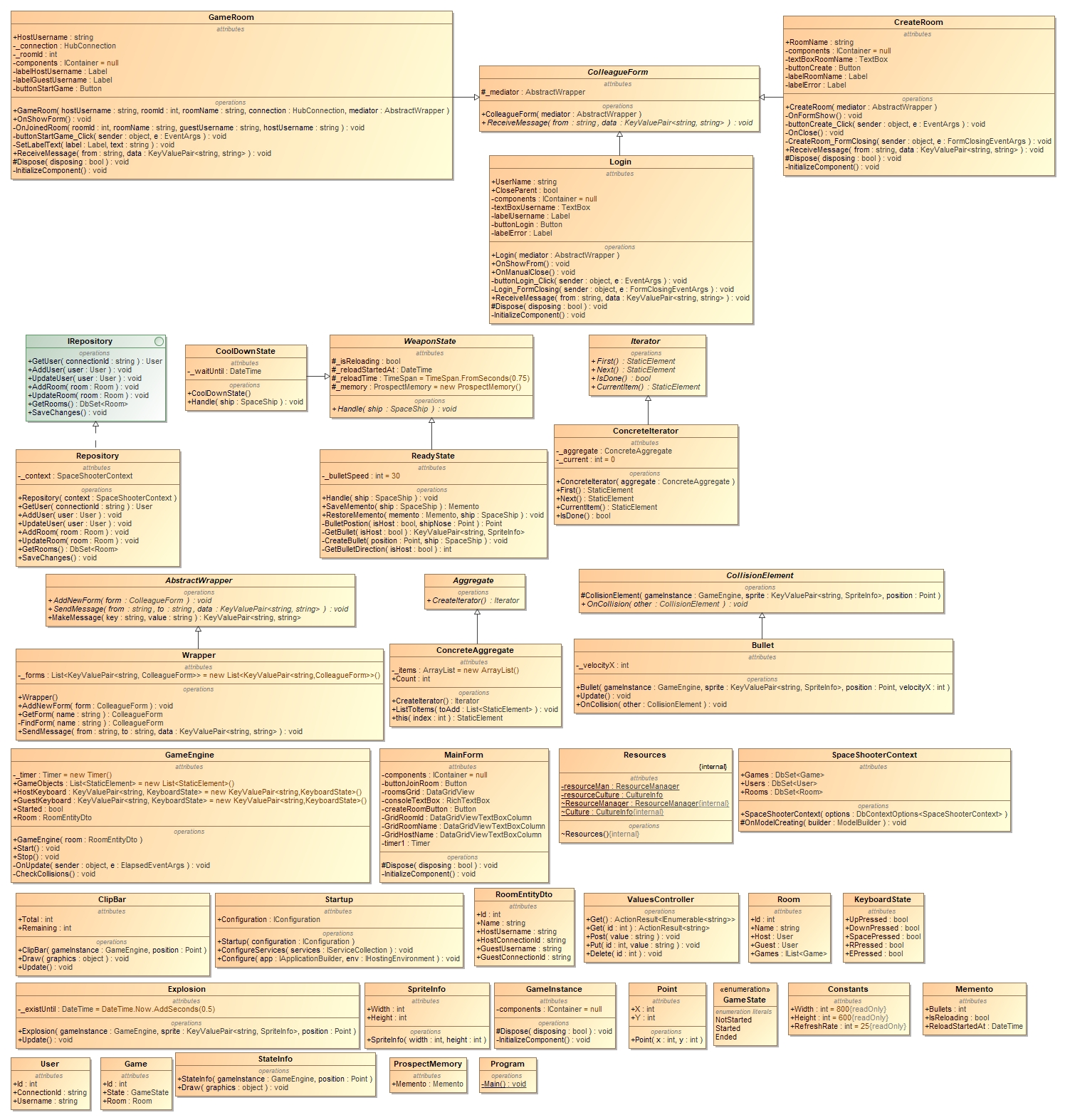
**Kursinio darbo ataskaita**

|  |  |  |
| --- | --- | --- |
|  | Studentai: | Mažvydas Vaicekauskas IFF 5/1  Gytis Apanavičius IFF 5/1  Lukas Gužauskas IFF 5/1  Mindaugas Nauronis IFF 5/1 |
| Dėstytojas: |  |

Turinys

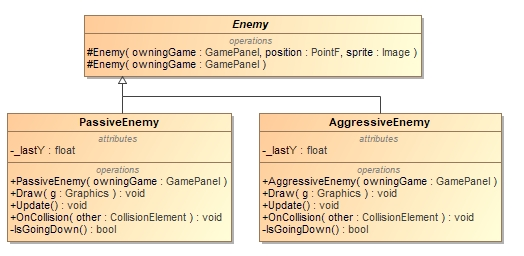
[Klasių diagrama 3](#_Toc533124718)

# Klasių diagrama



# Modeliai

## Template Method



### Kodas

#### public abstract class Enemy : CollisionElement

{

    int speed = 10;

    protected Enemy(GamePanel owningGame, PointF position, Image sprite)

        : base(owningGame, position, sprite)

    {

    }

    protected Enemy(GamePanel owningGame) : base(owningGame)

    {

    }

    public virtual void EnemyActions()

    {

        EnemyMovement();

        EnemyFire();

    }

    public virtual void EnemyMovement()

    {

        speed = 0;

    }

    public void EnemyFire()

    {

        double chargeTime = 0.5;

        double bulletSpeed = chargeTime \* 5;

        Bullet bullet = new DiagonalBullet(bulletSpeed);

        speed = 10;

    }

}

#### public class AggressiveEnemy : Enemy

    {

        private float \_lastY;

......

        public AggressiveEnemy(GamePanel owningGame) : base(owningGame)

        {

.....

        }

        public override void EnemyMovement()

        {

            speed = 15;

        }

        public override void EnemyFire()

        {

        }

........

}

public class PassiveEnemy : Enemy

{

    private float \_lastY;

    public PassiveEnemy(GamePanel owningGame) : base(owningGame)

    {

        \_lastY = Position.Y;

    }

    public override void EnemyMovement()

    {

        speed = 5;

    }

    public override void EnemyFire()

    {

        double chargeTime = 0.25;

        double bulletSpeed = chargeTime \* 5;

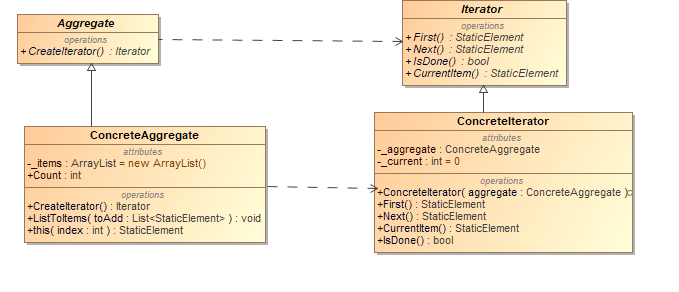
        Bullet bullet = new DiagonalBullet(bulletSpeed);

    }

........

}

## Iterator



### Kodas

#### abstract class Iterator

{

public abstract StaticElement First();

public abstract StaticElement Next();

public abstract bool IsDone();

public abstract StaticElement CurrentItem();

}

#### class ConcreteIterator : Iterator

{

private readonly ConcreteAggregate \_aggregate;

private int \_current = 0;

public ConcreteIterator(ConcreteAggregate aggregate)

{

this.\_aggregate = aggregate;

}

// Gets first iteration item

public override StaticElement First()

{

return \_aggregate[0];

}

// Gets next iteration item

public override StaticElement Next()

{

StaticElement ret = null;

if (\_current < \_aggregate.Count - 1)

{

ret = \_aggregate[++\_current];

}

return ret;

}

// Gets current iteration item

public override StaticElement CurrentItem()

{

return \_aggregate[\_current];

}

// Gets whether iterations are complete

public override bool IsDone()

{

return \_current >= \_aggregate.Count;

}

}

abstract class Aggregate

{

public abstract Iterator CreateIterator();

}

class ConcreteAggregate : Aggregate

{

private ArrayList \_items = new ArrayList();

public override Iterator CreateIterator()

{

return new ConcreteIterator(this);

}

public void ListToItems(List<StaticElement> toAdd)

{

\_items = new ArrayList();

foreach (StaticElement add in toAdd)

\_items.Add(add);

}

// Gets item count

public int Count

{

get { return \_items.Count; }

}

// Indexer

public StaticElement this[int index]

{

get { return (StaticElement)\_items[index]; }

set { \_items.Insert(index, value); }

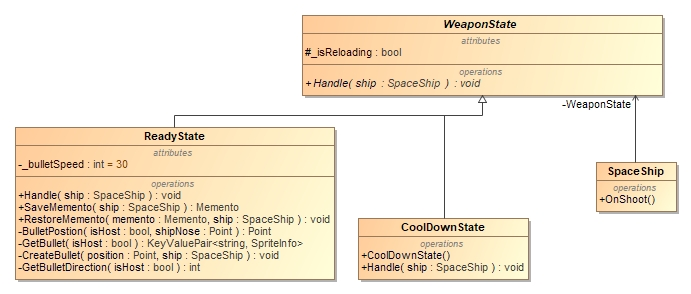
}

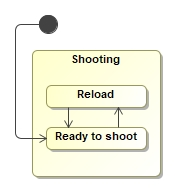
}

## Composite

## Flyweight

## State





### Kodas

#### public abstract class WeaponState

{

    protected bool \_isReloading;

    protected DateTime \_reloadStartedAt;

    protected readonly TimeSpan \_reloadTime = TimeSpan.FromSeconds(0.75);

    protected readonly ProspectMemory \_memory = new ProspectMemory();

    /// <summary>

    /// Handle shooting of space ship

    /// </summary>

    /// <param name="ship"></param>

    public abstract void Handle(SpaceShip ship);

}

#### public class ReadyState : WeaponState

    {

        private int \_bulletSpeed = 30;

        public override void Handle(SpaceShip ship)

        {

........

             else

            {

                if (ship.KeyboardState.SpacePressed && ship.Bullets > 0)

                {

                    Point bulletPosition = BulletPostion(ship.IsHost, ship.GetShipNose());

                    CreateBullet(bulletPosition, ship);

                    ship.WeaponState = new CoolDownState();

                    ship.Bullets--;

                }

.........

            }

        }

...........

    }

#### class CoolDownState : WeaponState

{

    private readonly DateTime \_waitUntil;

    public CoolDownState()

    {

        \_waitUntil = DateTime.Now.AddMilliseconds(200);

    }

    public override void Handle(SpaceShip ship)

    {

        if(DateTime.Now > \_waitUntil)

        {

            ship.WeaponState = new ReadyState();

            ship.StateInfo.Text = SpaceInfoStates.Ready;

        }

        else

        {

            ship.StateInfo.Text = SpaceInfoStates.Cooling;

        }

    }

}

#### public class SpaceShip : CollisionElement

{

..........

    public SpaceShip(GameEngine gameInstance, KeyValuePair<string, SpriteInfo> sprite, Point position, bool isHost)

        : base(gameInstance, sprite, position)

    {

..........

    }

    private void OnShoot()

    {

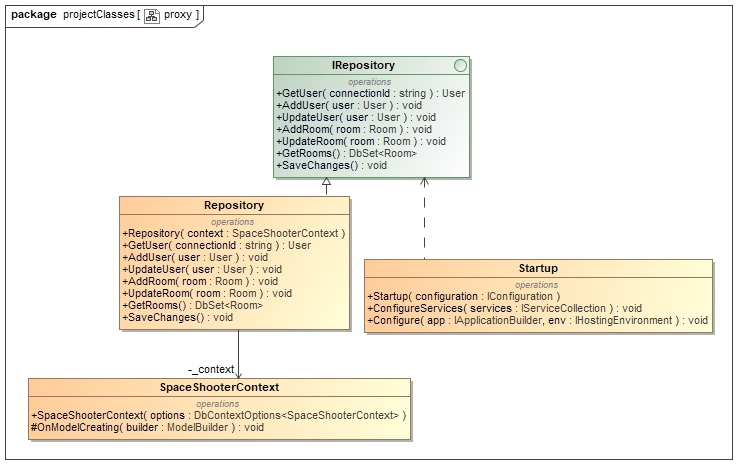
        WeaponState.Handle(this);

    }

    public WeaponState WeaponState { get; set; } = new ReadyState();

}

## Proxy



### Kodas

/// <summary>

    /// Proxy: Subject

    /// </summary>

#### public interface IRepository

    {

        User GetUser(string connectionId);

        void AddUser(User user);

        void UpdateUser(User user);

        void AddRoom(Room room);

        void UpdateRoom(Room room);

        DbSet<Room> GetRooms();

        void SaveChanges();

    }

/// <summary>

/// Proxy: Proxy

/// </summary>

#### public class Repository : IRepository

{

    private readonly SpaceShooterContext \_context;

    public Repository(SpaceShooterContext context)

    {

        \_context = context;

    }

    public User GetUser(string connectionId)

    {

        return \_context.Users.FirstOrDefault(x => x.ConnectionId == connectionId);

    }

    public void AddUser(User user)

    {

        \_context.Users.Add(user);

    }

    public void UpdateUser(User user)

    {

        \_context.Users.Update(user);

    }

    public void AddRoom(Room room)

    {

        \_context.Rooms.Add(room);

    }

    public void UpdateRoom(Room room)

    {

        \_context.Rooms.Update(room);

    }

    public DbSet<Room> GetRooms()

    {

        return \_context.Rooms;

    }

    public void SaveChanges()

    {

        \_context.SaveChanges();

    }

}

/// <summary>

/// Proxy: Real subject

/// </summary>

#### public class SpaceShooterContext : DbContext

{

    public SpaceShooterContext(DbContextOptions<SpaceShooterContext> options)

        : base(options)

    {

    }

    public DbSet<Game> Games { get; set; }

    public DbSet<User> Users { get; set; }

    public DbSet<Room> Rooms { get; set; }

    protected override void OnModelCreating(ModelBuilder builder)

    {

        builder.Entity<Room>(entity =>

        {

            entity.HasIndex(prop => prop.Name).IsUnique();

        });

    }

}

#### public class Startup

{

    public Startup(IConfiguration configuration)

    {

        Configuration = configuration;

    }

    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the container.

    public void ConfigureServices(IServiceCollection services)

    {

        services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);

        services.AddSignalR(options => options.EnableDetailedErrors = true);

        services.AddScoped<IRepository, Repository>();

        services.AddDbContext<SpaceShooterContext>(opt => opt.UseInMemoryDatabase("SpaceShooter"));

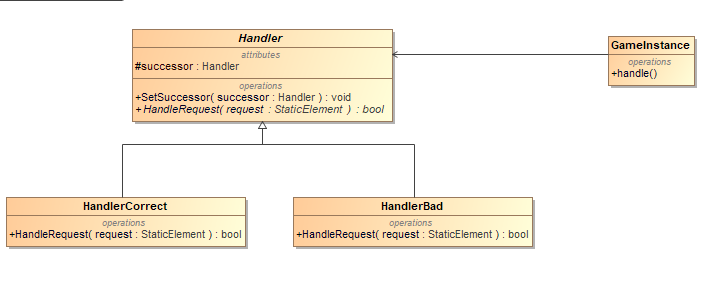
        services.AddScoped(p => new SpaceShooterContext(p.GetService<DbContextOptions<SpaceShooterContext>>()));

    }

.......

}

## Chain of responsibility



### Kodas

#### abstract class Handler

{

protected Handler successor;

public void SetSuccessor(Handler successor)

{

this.successor = successor;

}

public abstract bool HandleRequest(StaticElement request);

}

#### class HandlerBad : Handler

{

public override bool HandleRequest(StaticElement request)

{

if (request.Position.X <= 0 && request.Position.Y <= 0)

{

return false;

}

return true;

}

}

#### class HandlerCorrect : Handler

{

public override bool HandleRequest(StaticElement request)

{

if (request.Position.X > 0 && request.Position.Y > 0)

{

return true;

}

else if (successor != null)

{

return successor.HandleRequest(request);

}

return false;

}

}

private void OnUpdate(string message)

{

GameObjects.Clear();

var objects = JsonConvert.DeserializeObject<List<StaticElement>>(message);

ConcreteAggregate aggregate = new ConcreteAggregate();

aggregate.ListToItems(objects);

Iterator.Iterator jsonItterator = aggregate.CreateIterator();

Handler h1 = new HandlerCorrect();

Handler h2 = new HandlerBad();

h1.SetSuccessor(h2);

StaticElement item = jsonItterator.First();

while (item != null)

{

if (h1.HandleRequest((StaticElement)item))

GameObjects.Add((StaticElement)item);

item = jsonItterator.Next();

}

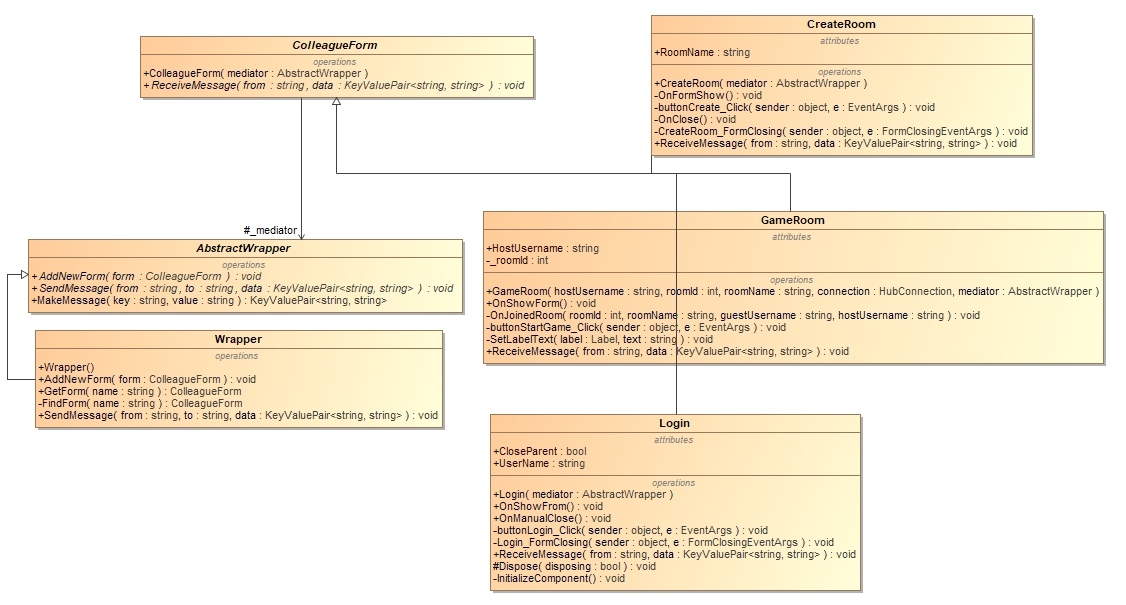
//objects.ForEach(x => GameObjects.Add(x));

this.Invalidate();

}

## Interpreter

## Mediator



### Kodas

#### public abstract class AbstractWrapper

{

    public abstract void AddNewForm(ColleagueForm form);

    public abstract void SendMessage(string from, string to, KeyValuePair<string, string> data);

    public KeyValuePair<string, string> MakeMessage(string key, string value)

    {

        return new KeyValuePair<string, string>(key, value);

    }

}

#### class Wrapper : AbstractWrapper

{

    private readonly List<KeyValuePair<string, ColleagueForm>> \_forms

        = new List<KeyValuePair<string, ColleagueForm>>();

    public Wrapper()

    {

        AddNewForm(new MainForm(this));

        AddNewForm(new Login(this));

        AddNewForm(new CreateRoom(this));

    }

    public override void AddNewForm(ColleagueForm form)

    {

        string formName = form.GetType().Name;

        \_forms.Add(new KeyValuePair<string, ColleagueForm>(formName, form));

    }

    private ColleagueForm FindForm(string name)

    {

        foreach (KeyValuePair<string, ColleagueForm> form in \_forms)

        {

            if (form.Key == name)

            {

                return form.Value;

            }

        }

        return null;

    }

    public override void SendMessage(string from, string to, KeyValuePair<string, string> data)

    {

        ColleagueForm receiver = FindForm(to);

        if (receiver == null)

        {

            string error =

                $"SendMessageError: receiver not found./n sender - {from}, data - {data.Key} => {data.Value}/n";

            throw new KeyNotFoundException(error);

        }

        receiver.ReceiveMessage(from, data);

    }

}

#### public partial class MainForm : ColleagueForm

{

    private readonly HubConnection \_connection = new HubConnectionBuilder()

        .WithUrl("http://localhost:5000/GameHub")

        //.WithUrl("http://spaceshooter-server.azurewebsites.net/GameHub")

        .Build();

    private string \_username;

    private int? \_roomId;

    public MainForm(AbstractWrapper mediator) :

        base(mediator)

    {

        InitializeComponent();

......

    }

......

private async void MainForm\_Load(object sender, EventArgs e)

{

    //TODO get rooms from server

    try

    {

......

        KeyValuePair<string, string> data = new KeyValuePair<string, string>

("showForm", "");

        \_mediator.SendMessage(this.GetType().Name, "Login", data);

    }

......

}

    private async void OnLogin(string username)

    {

        \_username = username;

        // Set username

        await \_connection.InvokeAsync("SetUsername", \_username);

        this.Text = \_username;

    }

    public override void ReceiveMessage(string from, KeyValuePair<string, string> data)

    {

        //message parser

        switch (data.Key.ToLower())

        {

            case "close":

                OnClose();

                break;

            case "username":

                OnLogin(data.Value);

                \_mediator.SendMessage(this.GetType().Name, from, \_mediator.MakeMessage("status", "OK"));

                break;

........

            default:

                KeyValuePair<string, string> error = new KeyValuePair<string, string>("status", "Cannot evulate");

                \_mediator.SendMessage(this.GetType().Name, from, error);

                break;

        }

    }

}

#### public partial class Login : ColleagueForm

{

    public string UserName { get; private set; }

    public bool CloseParent { get; private set; }

    public Login(AbstractWrapper mediator) :

        base (mediator)

    {

        InitializeComponent();

    }

    public void OnShowFrom()

    {

        this.ShowDialog();

    }

    private void buttonLogin\_Click(object sender, EventArgs e)

    {

......

        else

        {

            UserName = textBoxUsername.Text;

            KeyValuePair<string, string> message = new KeyValuePair<string, string>("username", UserName);

            \_mediator.SendMessage(this.GetType().Name, "MainForm", message);

            Close();

        }

    }

    public override void ReceiveMessage(string from, KeyValuePair<string, string> data)

    {

        //message parser

        switch (data.Key.ToLower())

        {

            case "showform":

                OnShowFrom();

                KeyValuePair<string, string> message = new KeyValuePair<string, string>("status", "OK");

                \_mediator.SendMessage(this.GetType().Name, from, message);

                break;

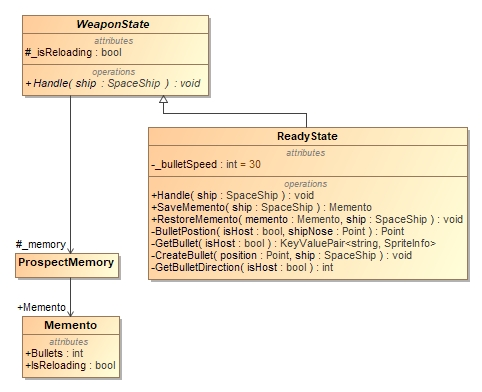
.......

        }

    }

}

## Memento



### Kodas

public class ProspectMemory

{

    public Memento Memento { get; set; }

}

public class Memento

{

    public int Bullets { get; set; }

    public bool IsReloading { get; set; }

    public DateTime ReloadStartedAt { get; set; }

}

public abstract class WeaponState

{

    protected bool \_isReloading;

    protected DateTime \_reloadStartedAt;

    protected readonly TimeSpan \_reloadTime = TimeSpan.FromSeconds(0.75);

    protected readonly ProspectMemory \_memory = new ProspectMemory();

    public abstract void Handle(SpaceShip ship);

}

public class ReadyState : WeaponState

    {

        private int \_bulletSpeed = 30;

        public override void Handle(SpaceShip ship)

        {

            if (\_isReloading)

            {

                if (ship.KeyboardState.EPressed)

                {

                    RestoreMemento(\_memory.Memento, ship);

                    ship.StateInfo.Text = SpaceInfoStates.Ready;

                }

.....

            }

            else

            {

.....

                else if (ship.KeyboardState.RPressed)

                {

                    \_memory.Memento = SaveMemento(ship);

                    \_isReloading = true;

                    ship.StateInfo.Text = SpaceInfoStates.Reloading;

                    \_reloadStartedAt = DateTime.Now;

                }

            }

        }

        public Memento.Memento SaveMemento(SpaceShip ship)

        {

            return new Memento.Memento

            {

                Bullets = ship.Bullets,

                IsReloading = \_isReloading,

                ReloadStartedAt = \_reloadStartedAt

            };

        }

        public void RestoreMemento(Memento.Memento memento, SpaceShip ship)

        {

            ship.Bullets = memento.Bullets;

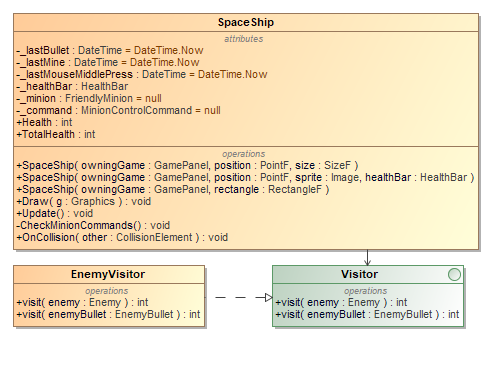
            \_isReloading = memento.IsReloading;

            \_reloadStartedAt = memento.ReloadStartedAt;

        }

    }

## Visitor



### Kodas

public interface Visitor

{

int visit(Enemy enemy);

int visit(EnemyBullet enemyBullet);

}

public class EnemyVisitor : Visitor

{

public int visit(Enemy enemy)

{

return 7;

}

public int visit(EnemyBullet enemyBullet)

{

return 3;

}

}

# Bandymai

šbandyti

Proxy

šabloną duomenų ištraukimui iš duomenų bazės.

Atlikti

greitavaikos bei naudojamos atminties matavimus naudojant

Flyweight

šabloną.

panaudoti

Null Object

šabloną

Pademonstruoti veikiantį

2D tinklo

žaidimą