KAUNO TECHNOLOGIJOS UNIVERSITETAS

INformatikos fakultetas

T120B516 Objektinis programų projektavimas

Projekto ataskaita

Studentų komanda:

Mantas Zambacevičius IFF-4/1,

Ernestas Venckus IFF-4/3,

Žilvinas Abromavičius IFF-4/3

Dėstytojas:

lekt. Barisas Dominykas

KAUNAS, 2017

Turinys

[1 Kuriamo žaidimo aprašymas 3](#_Toc496478111)

[2 Naudojami šablonai 3](#_Toc496478112)

[2.1 Singleton 3](#_Toc496478113)

[2.2 Factory 3](#_Toc496478114)

[2.3 Abstract factory 3](#_Toc496478115)

[2.4 Observer 3](#_Toc496478116)

[2.5 Adapter 3](#_Toc496478117)

[2.6 Prototype 3](#_Toc496478118)

[2.7 Command 3](#_Toc496478119)

[2.8 Bridge 3](#_Toc496478120)

[3 Šablonų esminis kodas 3](#_Toc496478121)

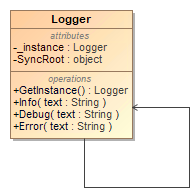
# Kuriamo žaidimo aprašymas

Projekto tikslas – sukurti nedidelį žaidimą, taikant projektavimo šablonus. Kuriamas žaidimas yra 2D šaudyklė.

# Naudojami šablonai

## Singleton

### Klasių diagrama



### Esminis kodas

using System;

namespace Shooter.Classes

{

public class Logger

{

private static volatile Logger \_instance;

private static readonly object SyncRoot = new object();

private Logger() { }

public static Logger Instance

{

get

{

if (\_instance == null)

{

lock (SyncRoot)

{

if(\_instance == null)

\_instance = new Logger();

}

}

return \_instance;

}

}

public void Info(string text) {…}

public void Debug(string text) {…}

public void Error(string text) {…}

private void WriteFormattedLog(LogLevel level, string text) {…}

private enum LogLevel {…}

}

}

### Naudojimo pagrindimas

Logger klasės paskirtis išvesti informacijos, derinimo ir klaidų pranešimus į konsolę. Kadangi visoje programoje reikalingas tik vienas šios klasės objektas, šiai klasėj pritaikomas „Singleton“ dizaino šablonas.

## Factory

### Klasių diagrama

### Esminis kodas

### Naudojimo pagrindimas

## Abstract factory

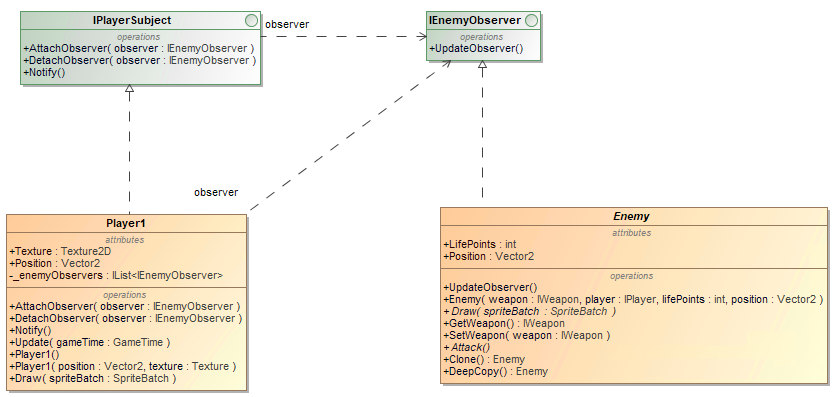
### Klasių diagrama

### Esminis kodas

### Naudojimo pagrindimas

## Observer

### Klasių diagrama



### Esminis kodas

namespace Shooter.Interfaces

{

interface IPlayerSubject

{

void AttachObserver(IEnemyObserver enemy);

void DetachObserver(IEnemyObserver enemy);

void Notify();

}

}

namespace Shooter.Interfaces

{

public interface IEnemyObserver

{

void UpdateObserver();

}

}

namespace Shooter.Classes

{

public abstract class Enemy : IEnemy, IEnemyObserver, IMapObject, EnemyPrototype

{

public Vector2 Position { get; set; }

public int LifePoints { get; set; }

private readonly IPlayer \_player;

protected IWeapon Weapon;

public abstract void Draw(SpriteBatch spriteBatch);

protected Enemy(IWeapon weapon, IPlayer player, int lifePoints, Vector2 position)

{

Weapon = weapon;

\_player = player;

LifePoints = lifePoints;

Position = position;

}

public IWeapon GetWeapon() {…}

public void SetWeapon(IWeapon weapon) {…}

public abstract void Attack();

public virtual void UpdateObserver()

{

Console.WriteLine($"Enemy notified of life points {\_player.LifePoints}");

}

public Enemy Clone() {…}

public Enemy DeepCopy() {…}

}

}

namespace Shooter.Classes

{

public class Player1 : IPlayer, IPlayerSubject, IMapObject

{

private readonly IList<IEnemyObserver> \_enemyObservers;

public Texture2D Texture { get; set; }

public Vector2 Position { get; set; }

public void Draw(SpriteBatch spriteBatch) {…}

public void Update(GameTime gameTime) {…}

public int LifePoints { get; set; }

public Player1() {…}

public Player1(Vector2 position, Texture2D texture) {…}

public void AttachObserver(IEnemyObserver observer)

{

\_enemyObservers.Add(observer);

}

public void DetachObserver(IEnemyObserver observer)

{

\_enemyObservers.Remove(observer);

}

public void Notify()

{

foreach (var enemyObserver in \_enemyObservers)

{

enemyObserver.UpdateObserver();

}

}

public void Move(KeyboardState keyboardState, KeyboardState previousState) {…}

}

}

### Naudojimo pagrindimas

„Observer“ dizaino šablono naudojimo paskirtis: kai egzistuoja „vienas-su-daug“ ryšis tarp objektų, ir norima vieno objekto būsenos pasikeitimus perduoti daugeliui kitų objektų. Projekto atveju, šis šablonas įgyvendintas tarp Player1 ir Enemy klasių: kai pasikeičia Player1 būsena (pvz. pasikeičia žaidėjo gyvybės taškai), tai yra informuojama visiems Enemy klasės objektams.

## Adapter

### Klasių diagrama

### Esminis kodas

### Naudojimo pagrindimas

## Prototype

### Klasių diagrama

### Esminis kodas

### Naudojimo pagrindimas

## Command

### Klasių diagrama

### Esminis kodas

### Naudojimo pagrindimas

## Bridge

### Klasių diagrama

### Esminis kodas

### Naudojimo pagrindimas

# Šablonų esminis kodas