

Informatikos fakultetas

**T120B516 Objektinis programų projektavimas**

**Projekto ataskaita**

|  |  |  |
| --- | --- | --- |
|  | Studentai: | Ignas Savickas IFF-4/2 |
| Jonas Ausevičius IFF-4/2 |
|  |  |
| Dėstytojai: | Lekt. dr. Andrej Ušianov |
| Lekt. Kęstutis Valinčius |

Kaunas 2017

Turinys

[1. Projekto aprašymas 3](#_Toc496625399)

[2. Projektavimo šablonai (1 laboratorinis) 4](#_Toc496625400)

[2.1. Singleton 4](#_Toc496625401)

[2.1.1. Klasių diagrama 4](#_Toc496625402)

[2.1.2. Esminis kodas 4](#_Toc496625403)

[2.1.3. Naudojimo pagrindimas 5](#_Toc496625404)

[2.2. Abstract Factory 5](#_Toc496625405)

[2.2.1. Klasių diagrama 5](#_Toc496625406)

[2.2.2. Esminis kodas 5](#_Toc496625407)

[2.2.3. Naudojimo pagrindimas 7](#_Toc496625408)

[2.3. Builder 8](#_Toc496625409)

[2.3.1. Klasių diagrama 8](#_Toc496625410)

[2.3.2. Esminis kodas 8](#_Toc496625411)

[2.3.3. Naudojimo pagrindimas 8](#_Toc496625412)

[3. Projektavimo šablonai (2 laboratorinis) 8](#_Toc496625413)

[3.1. Adapter 8](#_Toc496625414)

[3.1.1. Klasių diagrama 8](#_Toc496625415)

[3.1.2. Esminis kodas 8](#_Toc496625416)

[3.1.3. Naudojimo pagrindimas 8](#_Toc496625417)

[3.2. Prototype 8](#_Toc496625418)

[3.2.1. Klasių diagrama 9](#_Toc496625419)

[3.2.2. Esminis kodas 9](#_Toc496625420)

[3.2.3. Naudojimo pagrindinis 10](#_Toc496625421)

[3.3. Command 10](#_Toc496625422)

[3.3.1. Klasių diagrama 10](#_Toc496625423)

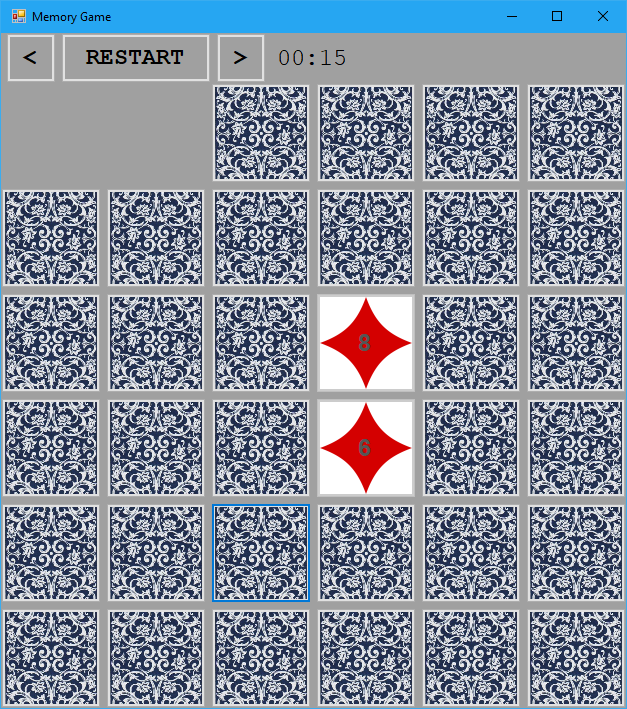
[3.3.2. Esminis kodas 10](#_Toc496625424)

[3.3.3. Naudojimo paskirtis 10](#_Toc496625425)

[4. Išvados 10](#_Toc496625426)

# Projekto aprašymas

Pasirinktas projektas – Atminties kortų žadinimas, panaudojant projektavimo šablonus. Programos pagrindiniame lange yra tam tikras užvertų kortų skaičius ir reikia rasti dvi (arba daugiau) vienodas kortas. Jeigu kita atverčiama korta yra tokia pati kaip prieš tai atversta, tai kortos pranyksta, o jeigu skiriasi – tai kortos užverčiamos.



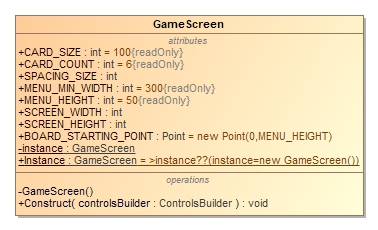
pav. 1. Žaidimo pagrindinis langas

# Projektavimo šablonai (1 laboratorinis)

Kiekvienam laboratoriniam darbui buvo pasirinkti atskiri projektavimo šablonai. Pirmajam – Singleton, Abstract Factory, Builder.

## Singleton

### Klasių diagrama



pav. 2. Singleton projektavimo šablono pritaikymo klasė

### Esminis kodas

class GameScreen

{

#region CARDS

public const int CARD\_SIZE = 100;

public const int CARD\_COUNT = 6;

public readonly int SPACING\_SIZE;

#endregion CARDS

#region MENU

public const int MENU\_MIN\_WIDTH = 300;

public const int MENU\_HEIGHT = 50;

#endregion MENU

public readonly int SCREEN\_WIDTH;

public readonly int SCREEN\_HEIGHT;

public readonly Point BOARD\_STARTING\_POINT = new Point(0, MENU\_HEIGHT);

private static GameScreen instance;

private GameScreen()

{

SPACING\_SIZE = Convert.ToInt32(CARD\_SIZE \* 0.05);

SCREEN\_WIDTH = CARD\_SIZE \* CARD\_COUNT + SPACING\_SIZE \* (CARD\_COUNT - 1);

SCREEN\_HEIGHT = SCREEN\_WIDTH + MENU\_HEIGHT;

}

public static GameScreen Instance => instance ?? (instance = new GameScreen());

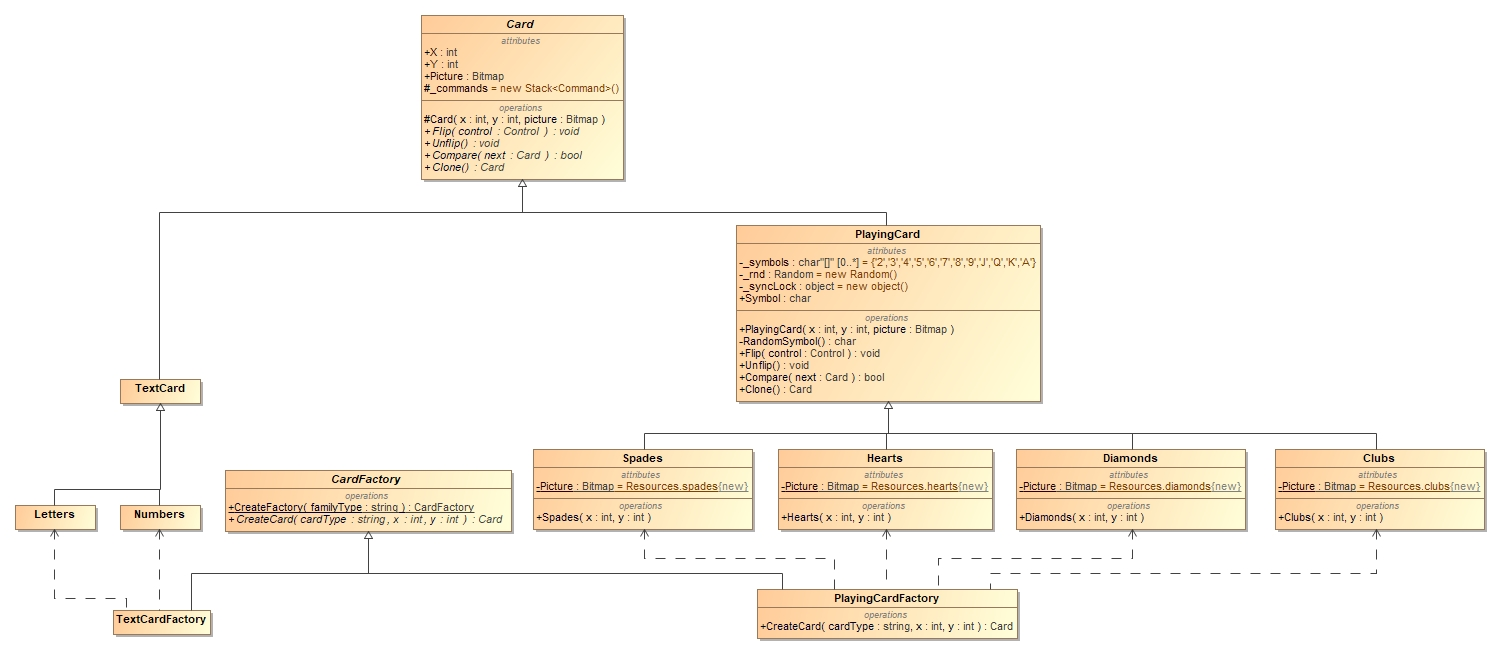
}

### Naudojimo pagrindimas

Per visą žaidimą bus tik vienas langas ir informacija bus atvaizduojama į jį ir ten bus saugomos konstantos. Kadangi bus naudojamas tik vienas objektas, tai čia pritaikomas Singleton projektavimo šablonas.

## Abstract Factory

### Klasių diagrama



pav. 3. Abstract Factory projektavimo šablono pritaikymo klasės

### Esminis kodas

public abstract class CardFactory

{

public static CardFactory CreateFactory(string familyType)

{

if (familyType.Equals("pc"))

{

return new PlayingCardFactory();

}

return null;

}

public abstract Card CreateCard( string cardType, int x, int y);

}

public class PlayingCardFactory : CardFactory

{

public override Card CreateCard(string cardType, int x, int y)

{

if (cardType.Equals("h"))

{

return new Hearts(x, y);

}

if (cardType.Equals("s"))

{

return new Spades(x, y);

}

if (cardType.Equals("c"))

{

return new Clubs(x, y);

}

if (cardType.Equals("d"))

{

return new Diamonds(x, y);

}

return null;

}

}

public abstract class Card

{

public int X { get; set; }

public int Y { get; set; }

public Bitmap Picture { get; set; }

protected Card(int x, int y, Bitmap picture)

{

X = x;

Y = y;

Picture = picture;

}

}

public class PlayingCard : Card

{

private readonly char[] \_symbols = { '2', '3', '4', '5', '6', '7', '8', '9', 'J', 'Q', 'K', 'A' };

private readonly Random \_rnd = new Random();

private readonly object \_syncLock = new object();

public char Symbol { get; set; }

public PlayingCard(int x, int y, Bitmap picture) : base(x, y, picture)

{

Symbol = RandomSymbol();

}

private char RandomSymbol()

{

lock (\_syncLock)

{

var index = \_rnd.Next(0, 11);

return \_symbols[index];

}

}

}

public class Diamonds : PlayingCard

{

private new static readonly Bitmap Picture = Resources.diamonds;

public Diamonds(int x, int y) : base(x, y, Picture)

{

}

}

public class Hearts : PlayingCard

{

private new static readonly Bitmap Picture = Resources.hearts;

public Hearts(int x, int y) : base(x, y, Picture)

{

}

}

public class Clubs : PlayingCard

{

private new static readonly Bitmap Picture = Resources.clubs;

public Clubs(int x, int y) : base(x, y, Picture)

{

}

}

public class Spades : PlayingCard

{

private new static readonly Bitmap Picture = Resources.spades;

public Spades(int x, int y) : base(x, y, Picture)

{

}

}

### Naudojimo pagrindimas

Galima kurti šeimas ir susijusių kortų, nenurodant kortos klasės, ir sukurtas kortas atvaizduoti žaidimo lange.

## Builder

### Klasių diagrama

### Esminis kodas

### Naudojimo pagrindimas

# Projektavimo šablonai (2 laboratorinis)

Antrajame laboratoriniame darbe buvo naudojami 3 projektavimo šablonai: Adapter, Prototype, Command.

## Adapter

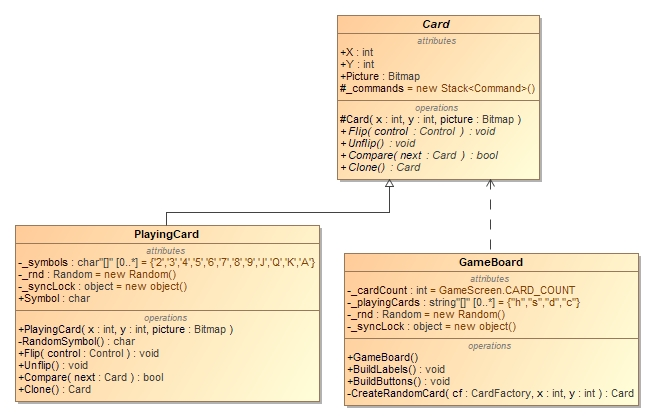
### Klasių diagrama

### Esminis kodas

### Naudojimo pagrindimas

## Prototype

### Klasių diagrama



pav. 4. Prototype projektavimo šablono pritaikymo klasės

### Esminis kodas

public abstract class Card

{

public int X { get; set; }

public int Y { get; set; }

public Bitmap Picture { get; set; }

protected Card(int x, int y, Bitmap picture)

{

X = x;

Y = y;

Picture = picture;

}

public abstract Card Clone();

}

public class PlayingCard : Card

{

private readonly char[] \_symbols = { '2', '3', '4', '5', '6', '7', '8', '9', 'J', 'Q', 'K', 'A' };

private readonly Random \_rnd = new Random();

private readonly object \_syncLock = new object();

public char Symbol { get; set; }

public PlayingCard(int x, int y, Bitmap picture) : base(x, y, picture)

{

Symbol = RandomSymbol();

}

private char RandomSymbol()

{

lock (\_syncLock)

{

var index = \_rnd.Next(0, 11);

return \_symbols[index];

}

}

public override Card Clone()

{

return (Card) this.MemberwiseClone();

}

}

### Naudojimo pagrindinis

Kuriant kortas reikalinga sukurtos kortos kopija (arba kopijos), kad žaidimo lange būtų vienodų kortų žaidimo veikimui.

## Command

### Klasių diagrama

### Esminis kodas

### Naudojimo paskirtis

# Išvados