

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](#_Toc496646028)

[2. Projektavimo šablonai (1 laboratorinis) 4](#_Toc496646029)

[2.1. Singleton 4](#_Toc496646030)

[2.1.1. Klasių diagrama 4](#_Toc496646031)

[2.1.2. Esminis kodas 4](#_Toc496646032)

[2.1.3. Naudojimo pagrindimas 5](#_Toc496646033)

[2.2. Abstract Factory 5](#_Toc496646034)

[2.2.1. Klasių diagrama 5](#_Toc496646035)

[2.2.2. Esminis kodas 6](#_Toc496646036)

[2.2.3. Naudojimo pagrindimas 8](#_Toc496646037)

[2.3. Builder 8](#_Toc496646038)

[2.3.1. Klasių diagrama 8](#_Toc496646039)

[2.3.2. Esminis kodas 9](#_Toc496646040)

[2.3.3. Naudojimo pagrindimas 9](#_Toc496646041)

[3. Projektavimo šablonai (2 laboratorinis) 10](#_Toc496646042)

[3.1. Adapter 10](#_Toc496646043)

[3.1.1. Klasių diagrama 10](#_Toc496646044)

[3.1.2. Esminis kodas 10](#_Toc496646045)

[3.1.3. Naudojimo pagrindimas 11](#_Toc496646046)

[3.2. Prototype 11](#_Toc496646047)

[3.2.1. Klasių diagrama 11](#_Toc496646048)

[3.2.2. Esminis kodas 11](#_Toc496646049)

[3.2.3. Naudojimo pagrindinis 12](#_Toc496646050)

[3.3. Command 12](#_Toc496646051)

[3.3.1. Klasių diagrama 12](#_Toc496646052)

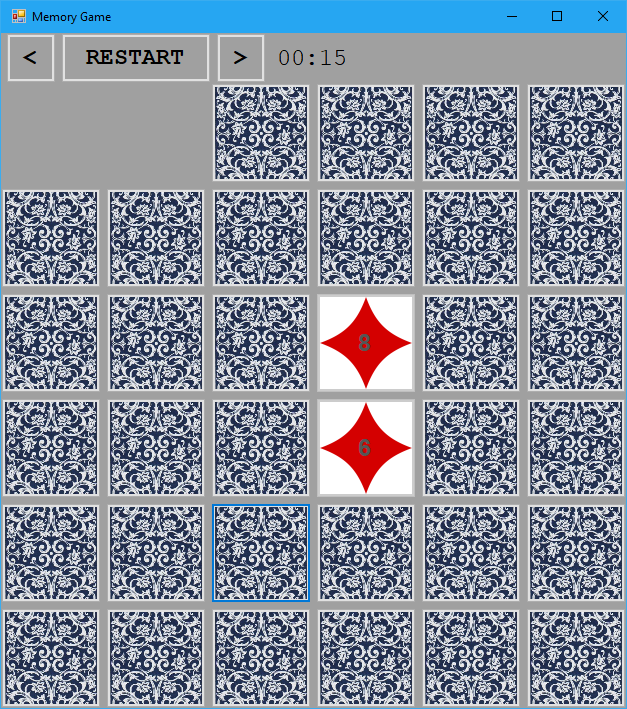
[3.3.2. Esminis kodas 13](#_Toc496646053)

[3.3.3. Naudojimo pagrindimas 14](#_Toc496646054)

[4. Išvados 15](#_Toc496646055)

# 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.



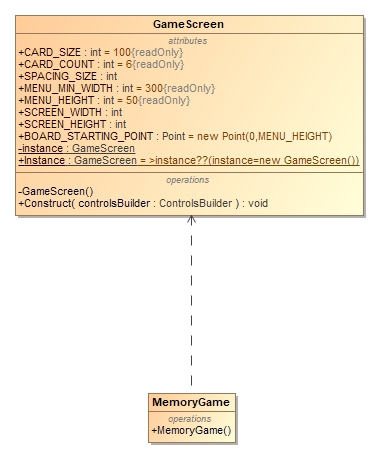
pav. 1. Žaidimo pagrindinis langas

# 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());

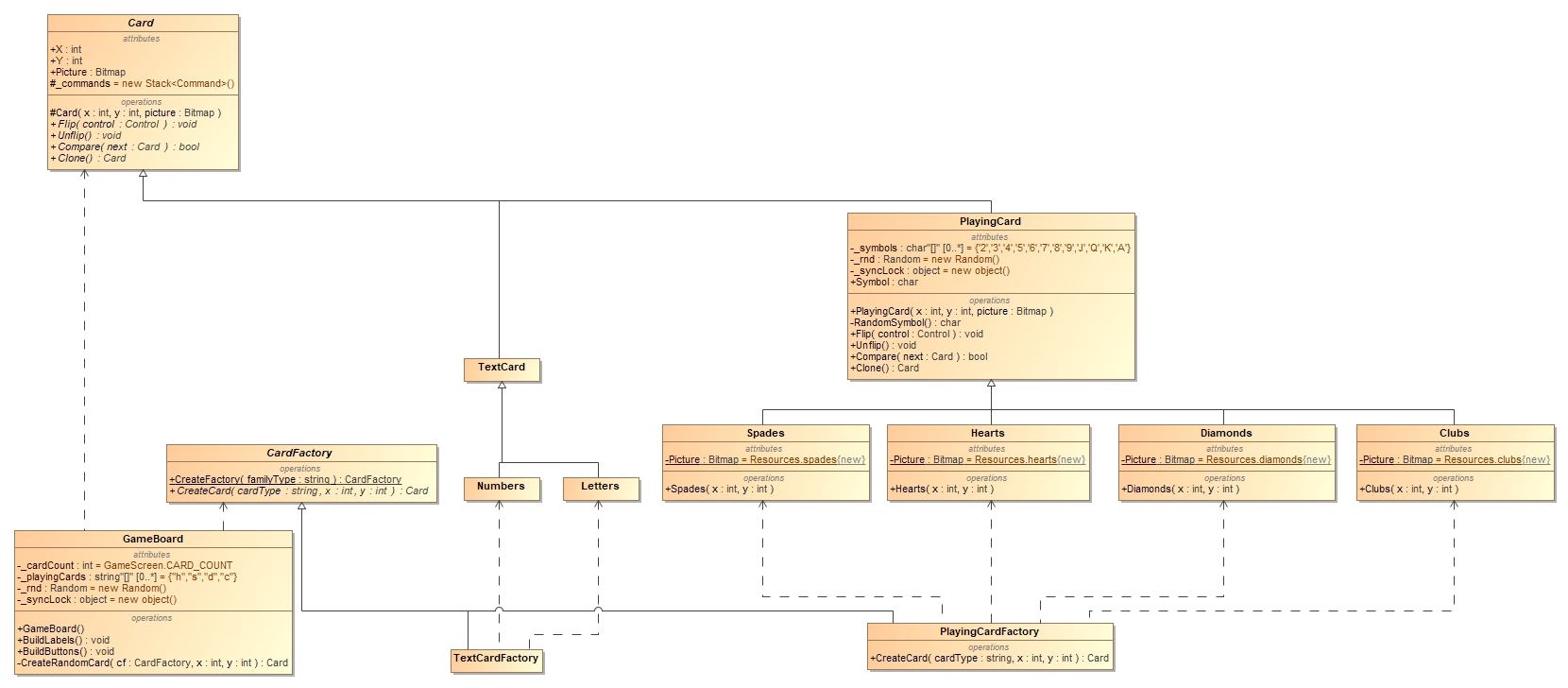
}

### 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. Singleton taip pat naudojamas laikyti konstantas (užtikrinama, kad visose klasės bus naudojamos tos pačios reikšmės).

## 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)

{

}

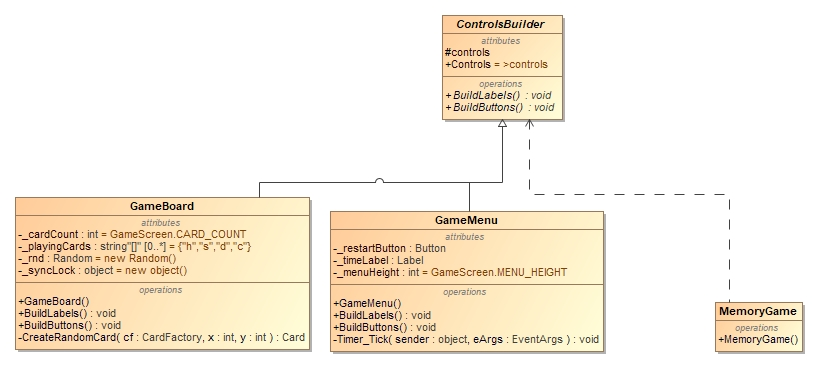
}

### Naudojimo pagrindimas

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

## Builder

### Klasių diagrama



pav. 4. Builder šablono pritaikymo klasių lentelės

### Esminis kodas

abstract class ControlsBuilder

{

protected List<Control> controls;

public List<Control> Controls => controls;

public abstract void BuildLabels();

public abstract void BuildButtons();

}

class GameBoard : ControlsBuilder

{

//....

public GameBoard()

{

controls = new List<Control>();

}

public override void BuildLabels() { }

public override void BuildButtons()

{

//...

}

//...

}

class GameMenu : ControlsBuilder

{

//...

public GameMenu()

{

controls = new List<Control>();

//...

}

public override void BuildLabels()

{

//...

}

public override void BuildButtons()

{

//...

}

}

### Naudojimo pagrindimas

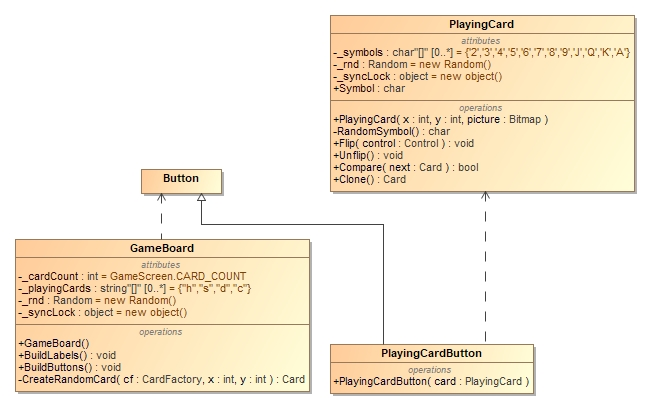
Builder šablonas leido pritaikyti tą patį objektų kūrimo procesą kuriant skirtingoms dalims. Naudojamas teksto ir mygtukų kūrimui skirtingoms ekrano dalims.

# Projektavimo šablonai (2 laboratorinis)

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

## Adapter

### Klasių diagrama



pav. 5. Adapter projektavimo šablono klasių diagrama

### Esminis kodas

public class PlayingCardButton : Button

{

public PlayingCardButton(PlayingCard card)

{

var xOffset = GameScreen.Instance.SPACING\_SIZE \* card.X;

var yOffset = GameScreen.Instance.SPACING\_SIZE \* card.Y;

var startPoint = GameScreen.Instance.BOARD\_STARTING\_POINT;

var cardSize = GameScreen.CARD\_SIZE;

Tag = card;

BackgroundImage = Resources.flipped\_card;

BackColor = Color.White;

BackgroundImageLayout = ImageLayout.Stretch;

Left = startPoint.X + xOffset + cardSize \* card.X;

Top = startPoint.Y + yOffset + cardSize \* card.Y;

Width = cardSize;

Height = cardSize;

//Text = card.Symbol.ToString();

ForeColor = Color.White;

Font = new Font("Arial", 18, FontStyle.Bold);

Click += GameControls.CardButton\_Click;

}

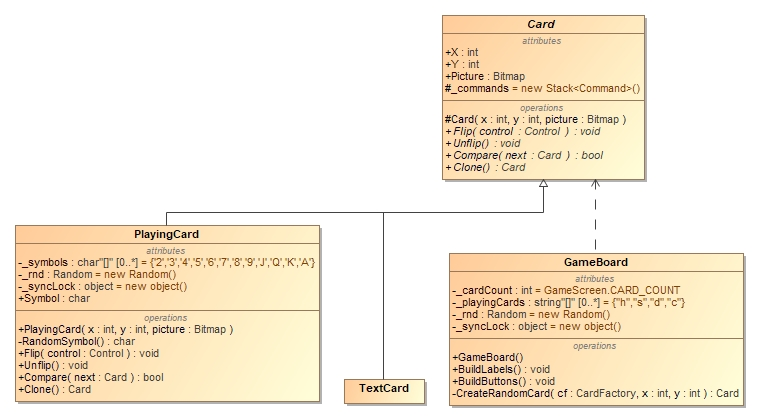
}

### Naudojimo pagrindimas

Adapter šablonas panaudotas pakeisti PlayingCard klasės objektą į Button objektą. Vietoj Button kūrimo ir PlayingCard objekto kintamųjų perkėlimo į jį vietose kur reikalingas kortos mygtukas, sukurtas šablonas leidžia lengvai pakeisti kortą į mygtuką, taip pat išsaugant kortos klasę, jei reikėtų panaudoti ne kaip mygtuką.

## Prototype

### Klasių diagrama



pav. 6. 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();

}

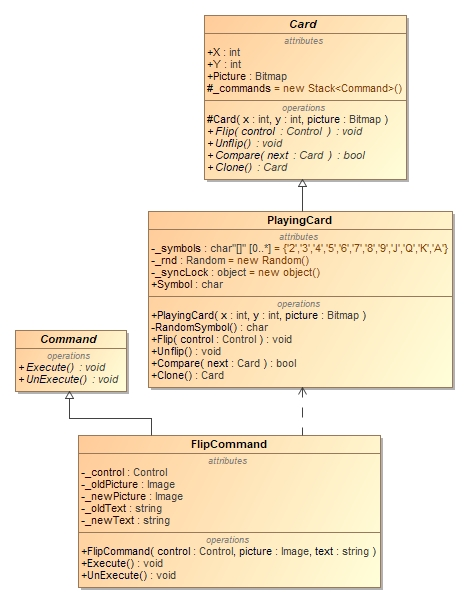
}

### Naudojimo pagrindinis

Kuriant kortas reikalinga sukurtos kortos kopija (arba kopijos), kad žaidimo lange būtų vienodų kortų žaidimo veikimui, bei, kad nebūtų bereikalingai konstruktorius ir *new*.

## Command

### Klasių diagrama



pav. 7. Command projektavimo šablono klasių diagrama

### Esminis kodas

public abstract class Command

{

public abstract void Execute();

public abstract void UnExecute();

}

class FlipCommand : Command

{

private readonly Control \_control;

private readonly Image \_oldPicture;

private readonly Image \_newPicture;

private readonly string \_oldText;

private readonly string \_newText;

public FlipCommand(Control control, Image picture, string text)

{

//...

}

public override void Execute()

{

//...

}

public override void UnExecute()

{

//...

}

}

public class PlayingCard : Card

{

//...

protected readonly Stack<Command> \_commands = new Stack<Command>();

public override void Flip(Control control)

{

Command command = new FlipCommand(control, Picture, Symbol.ToString());

command.Execute();

\_commands.Push(command);

}

public override void Unflip()

{

if (\_commands.Count > 0)

{

\_commands.Pop().UnExecute();

}

}

}

### Naudojimo pagrindimas

Komandos šablonas naudojamas kortos apvertimui užfiksuoti. Kortą atvertus išsaugojama jos buvusi stadija, kad galima būtų sugražinti ją užvertus. Sukurtas šablonas leidžia saugoti neribotą kiekį stadijų – korta gali būti „verčiama“ kelis kartus.

# Išvados

Laboratorinio darbo metu susipažinta su daugeliu įvairių projektavimo šablonų (Singleton – naudojamas konstantoms saugot, Abstract Factory – kortų šeimų ir susijusių kortų kūrimui nenurodant kortos klaės, Builder – žaidimo lango dalių kūrimo procesui, Adapter – konvertavimui iš kortos į mygtuko klasę, Prototype – kortos kopijai kurti, Command – kortos atvertimams saugoti) kurie buvo taikomi kuriamam atminties žaidimui. Šablonai leido paprasčiau projektuoti žaidimo logiką, kodas gali būti nesunkiai vystomas projekto kūrimo eigoje.