**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**T120B516 Objektinis programų projektavimas**

**(Design Patterns)**

**Projektinio darbo ataskaita**

Atliko: Karolina Jašauskaitė, IFF-4/1,

Žygimantas Skinkys, IFF-4/1,

Aidas Balčaitis, IFF-4/1

Kaunas 2017

Turinys

[Įvadas 3](#_Toc500794607)

[Darbo tikslas 3](#_Toc500794608)

[1. Pirmas labaratorinis darbas 3](#_Toc500794609)

[Programavimo šablonai: 3](#_Toc500794610)

[1.1. Singleton 3](#_Toc500794611)

[1.2. Factory 4](#_Toc500794612)

[1.3. Abstract Factory 6](#_Toc500794613)

[1.4. Strategy 7](#_Toc500794614)

[1.5. Observer 8](#_Toc500794615)

[2. Antras labaratorinis darbas 12](#_Toc500794616)

[Programavimo šablonai 12](#_Toc500794617)

[2.1. Adapter 12](#_Toc500794618)

[2.2. Decorator 14](#_Toc500794619)

[2.3. Facade 16](#_Toc500794620)

[2.4. Command 18](#_Toc500794621)

[3. 1 ir 2 Labaratorinių darbų klasių diagramos 21](#_Toc500794622)

[4. Trečias labaratorinis darbas 22](#_Toc500794623)

[4.1. Template Method 22](#_Toc500794624)

[4.2. State 26](#_Toc500794625)

[4.3. Proxy 28](#_Toc500794626)

[4.4. Flyweight 31](#_Toc500794627)

[5. Ketvirtas labaratorinis darbas 34](#_Toc500794628)

[5.1. Chain Of Responsibility 34](#_Toc500794629)

[5.2. Mediator 36](#_Toc500794630)

[5.3. Memento 38](#_Toc500794631)

[5.4. Interpreter 40](#_Toc500794632)

[6. Labaratorinių darbų klasių diagramos 43](#_Toc500794633)

[Išvados 43](#_Toc500794634)

# Įvadas

## Darbo tikslas

Kuriant „Pacman” žaidimą išmokti taikyti projektavimo šablonus (angl. „design patterns“) ir susipažinti su jų naudojimo ypatumais.

„Pacman” žaidime vartotojas valdys pagrindinį herojų, kurio tikslas surinkti kuo daugiau taškį renkant “Candy”, ir išvengi priešų “Enemy”.

Žaidime naudojami 5 pagrindiniai objektai: žaidimas, žaidimo lenta, priešai, saldainiai ir žaidėjo valdomas objektas.

# Pirmas labaratorinis darbas

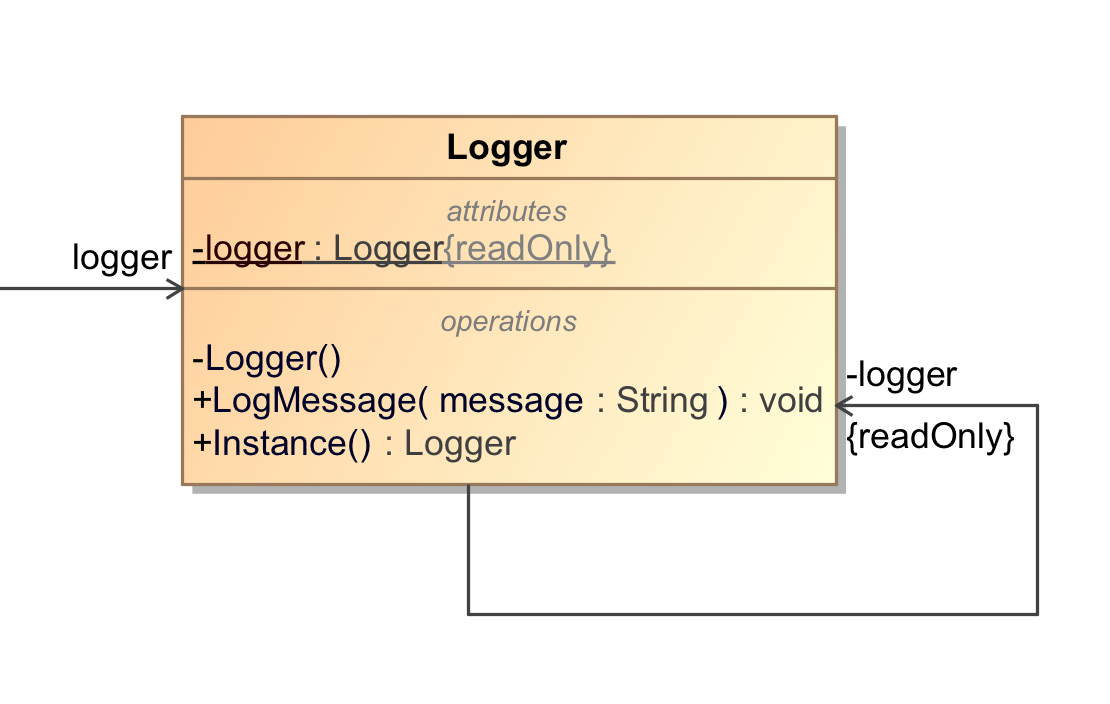
## Programavimo šablonai:

## Singleton

Panaudojimo tikslas:

Panaudojome Singleton programavimo šabloną, kad užtikrintume, kad vienu metu būtų sukurtas tik vienas logger objektas.

Uml diagrama:



Pav. 1 Singleton programavimo šablonas

Kodas:

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Pacman.Classes

{

class Logger

{

private static readonly Logger logger = new Logger();

static Logger() { } // Make sure it's truly lazy

private Logger() { } // Prevent instantiation outside

public static Logger Instance { get { return logger; } }

public void LogMessage(string message)

{

Debug.WriteLine(message);

}

}

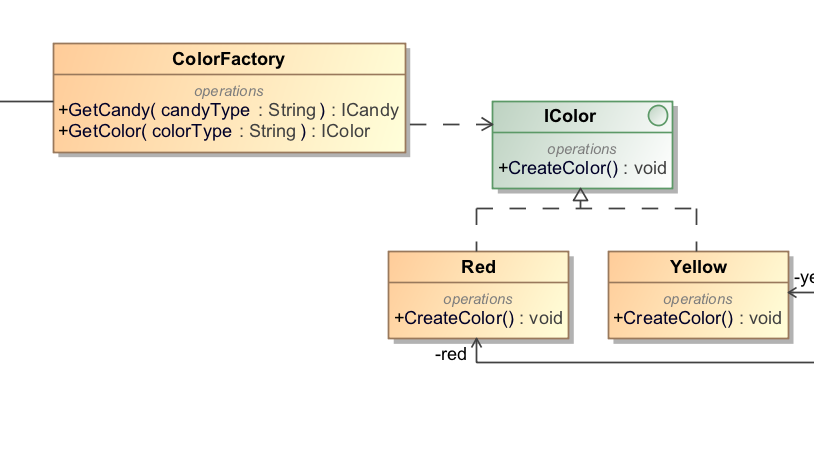
}

## Factory

Panaudojimo tikslas:

Kadangi turime 2 skirtingų tipų spalvas (Red, Yellow), norint jas realizuoti ir panaudoti savo žaidime, pasinaudojome “Factory” programavimo šabloną, kuris pagal gautą atributą sukurs saldainius.

Uml diagrama:



Pav. 2 Factory programavimo šablonas

Kodas:

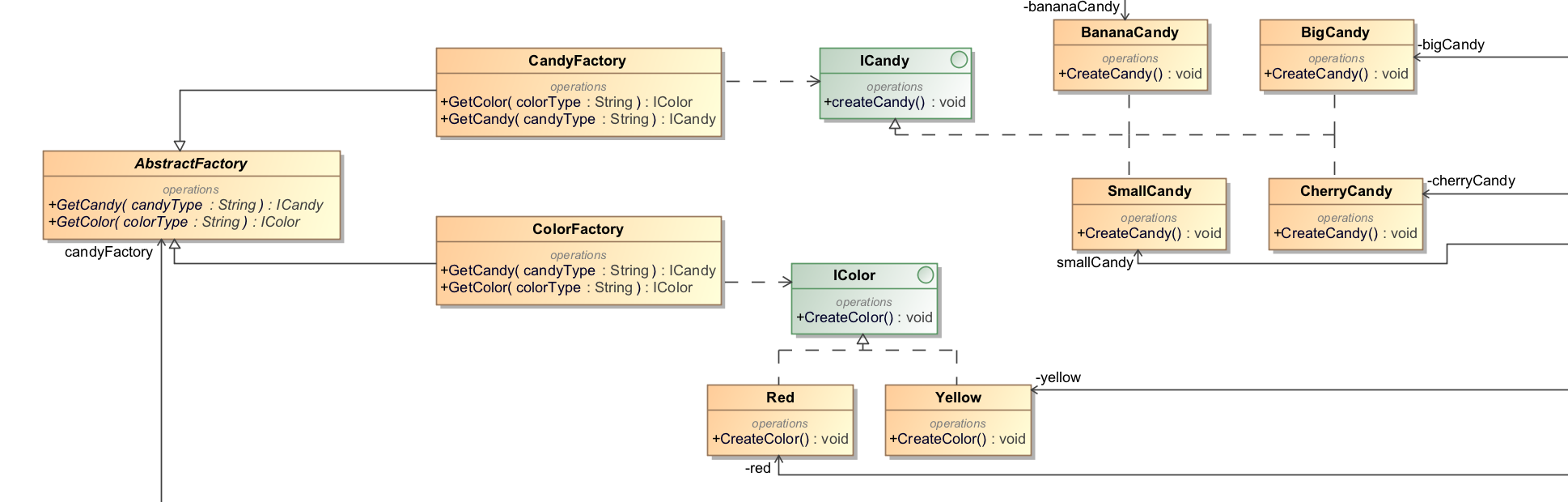
|  |
| --- |
| ColorFactory.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using Pacman.Classes.Colors;  namespace Pacman.Classes  {  class ColorFactory : AbstractFactory  {  public override ICandy GetCandy(string candyType)  {  return null;  }  public override IColor GetColor(string colorType)  {  if (colorType == null)  {  return null;  }  if (colorType.Equals("yellow", StringComparison.InvariantCultureIgnoreCase))  {  return new Yellow();  }  if (colorType.Equals("red", StringComparison.InvariantCultureIgnoreCase))  {  return new Red();  }  return null;  }  }  } |
| Red.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes.Colors  {  class Red : IColor  {  public void CreateColor()  {  Debug.WriteLine("Red color was created.");  }  }  } |
| Yellow.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes.Colors  {  class Yellow : IColor  {  public void CreateColor()  {  Debug.WriteLine("Yellow color was created.");  }  }  } |

## Abstract Factory

Panaudojimo tikslas:

Abstract Factory šabloną panaudojom apjungti ankščiau panaudotų Factory šablonų apjungimui.

Uml diagrama:



Pav. 3 Abstract Factory programavimo šablonas

Kodas:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Pacman.Classes

{

abstract class AbstractFactory

{

public abstract ICandy GetCandy(string candyType);

public abstract IColor GetColor(string colorType);

}

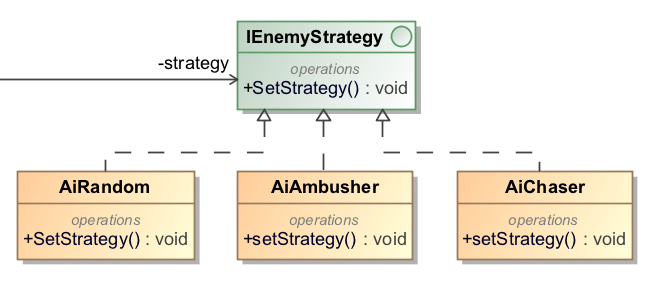
}

## Strategy

Panaudojimo tikslas:

Strategy šabloną panaudojom apibrėžti kelis Enemy algoritmus, juos paslėpti. Strategy šablonas leidžia pakeisti strategiją nekeiciant Enemy. Pasirinkome būtent šį projektavimo modelį, kadangi, mūsų nuomone, jis tinkamiausias iš elgsenos modulių mūsų projektui ir geriausiai atspindi mūsų norimo funkcionalumo realizaciją. Žemiau pateiktame paveikslėlyje matyti kaip panaudojant stategy pattern realizuoti veiksmai.

Uml diagrama:



Pav. 4 Strategy programavimo šablonas

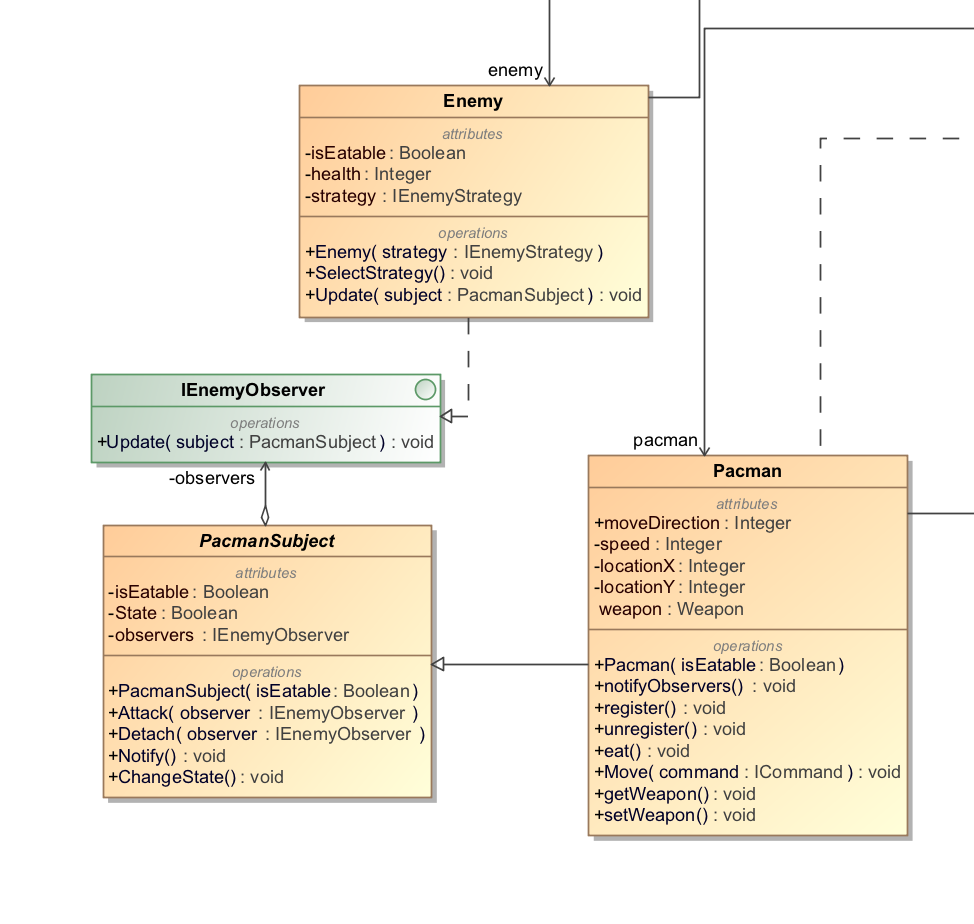
Kodas:

|  |
| --- |
| IEnemyStrategy.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  interface IEnemyStrategy  {  void SetStrategy();  }  } |
| AiRandom.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class AiRandom : IEnemyStrategy  {  public void SetStrategy()  {  Debug.WriteLine("Setting strategy to random");  }  }  } |
| AiAmbusher.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class AiAmbusher : IEnemyStrategy  {  public void SetStrategy()  {  Debug.WriteLine("Setting strategy to ambusher");  }  }  } |
| AiChaser.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class AiChaser : IEnemyStrategy  {  public void SetStrategy()  {  Debug.WriteLine("Setting strategy to chaser.");  }  }  } |

## Observer

Panaudojimo tikslas:

Observer šablonas buvo panaudotas tam , kad Pacman praneštų visiems priešams, kada jis suvalgė stebuklingą saldainį.



Uml diagrama:

Pav. 5 Observer programavimo šablonas

Kodas:

|  |
| --- |
| IEnemyObserver.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  interface IEnemyObserver  {  void Update(PacmanSubject subject);  }  } |
| PacmanSubject.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Runtime.CompilerServices;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  abstract class PacmanSubject  {  private bool isEatable;  private List<IEnemyObserver> observers = new List<IEnemyObserver>();  public PacmanSubject(bool isEatable)  {  this.isEatable = isEatable;  }  public void Attach(IEnemyObserver observer)  {  observers.Add(observer);  }  public void Detach(IEnemyObserver observer)  {  observers.Remove(observer);  }  public void Notify()  {  foreach (IEnemyObserver observer in observers)  {  observer.Update(this);  }  }  public bool State  {  get { return isEatable; }  set  {  if (isEatable != value)  {  isEatable = value;  Notify();  }  }  }  public void ChangeState()  {  isEatable = !isEatable;  Notify();  }  }  } |
| Pacman.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class Pacman : PacmanSubject  {  public int moveDirection { set; get; }  private int speed;  private int locationX;  private int locationY;  public Weapon weapon { set; get; }  public Pacman(bool isEatable) : base(isEatable)  {  }  public void notifyObservers()  {    }  public void register()  {    }  public void unregister()  {    }  public void eat()  {    }  public void Move(ICommand command)  {  command.Move();  }  public void getWeapon()  {    }  public void setWeapon()  {    }  }  } |
| Enemy.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class Enemy : IEnemyObserver  {  private bool isEatable;  private int health;  public IEnemyStrategy strategy { get; set; }  public Enemy(IEnemyStrategy strategy)  {  this.strategy = strategy;  }  public void SelectStrategy()  {  strategy.SetStrategy();  }  public void Update(PacmanSubject subject)  {  Debug.WriteLine("Pacman changed it's state to " + subject.State);  }  }  } |

# Antras labaratorinis darbas

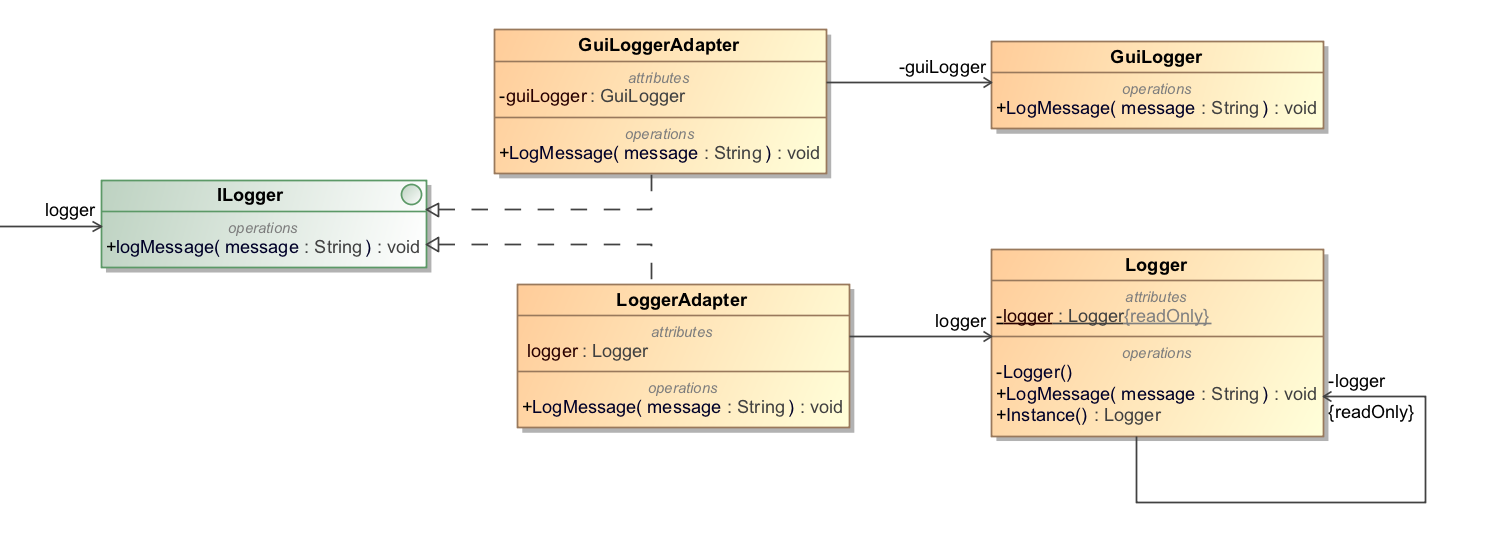
## Programavimo šablonai

## Adapter

Panaudojimo tikslas:

Adapter programavimo šabloną naudojame, kad turėtume galimybę paleisti programą ir su grafine sąsaja ir be jos.

Uml diagrama:



Pav. 6 Adapter programavimo šablonas

Kodas:

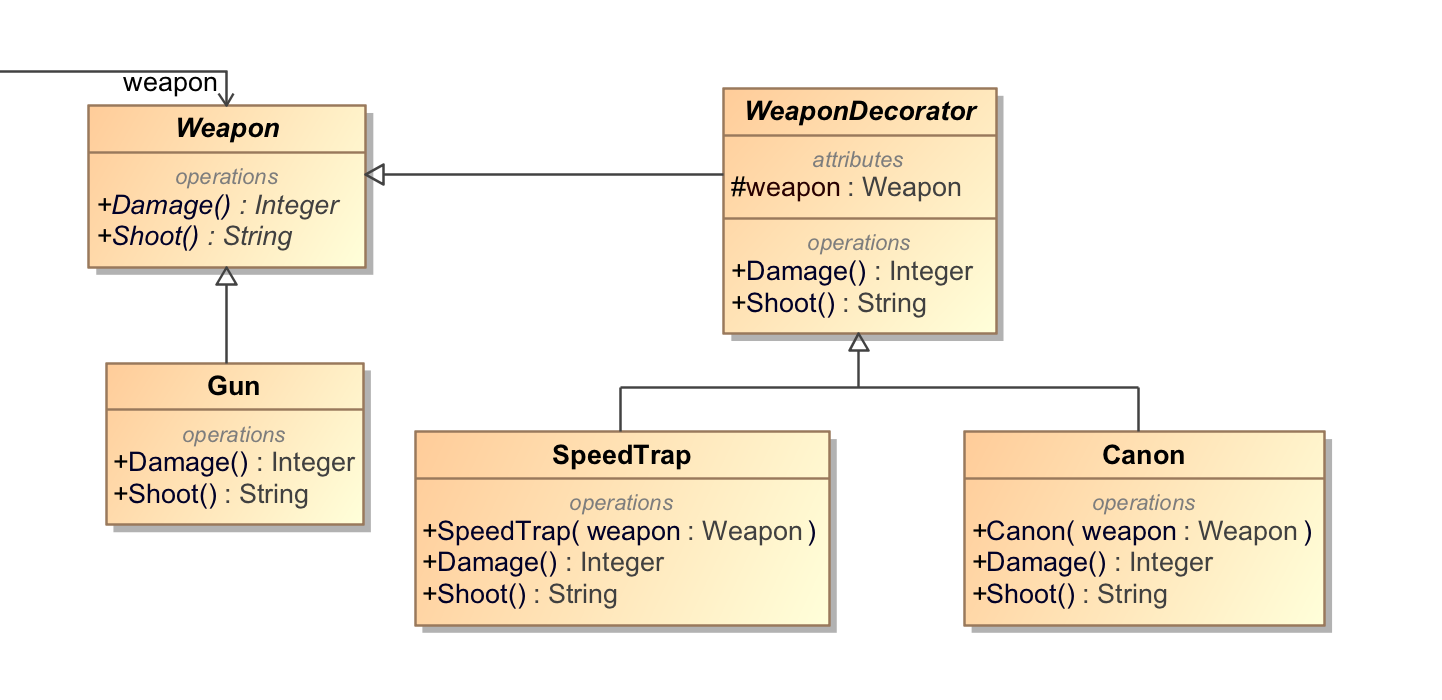
|  |
| --- |
| GuiLoggerAdapter.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class GuiLoggerAdapter : ILogger  {  private GuiLogger guiLogger = new GuiLogger();  public void LogMessage(string message)  {  guiLogger.LogMessage(message);  }  }  } |
| LoggerAdapter.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class LoggerAdapter : ILogger  {  public void LogMessage(string message)  {  Logger.Instance.LogMessage(message);  }  }  } |

## Decorator

Panaudojimo tikslas:

Pasirinkome naudoti Decorator programavimo šabloną, nes Pacman gali turėti daugiau nei vieną ginklą. Vietoje to kad naudoti switch case ar if sakinius priklausomai nuo pasirinkto ginklo, panaudojome Decorator programavimo šabloną, kad ginklai būtų pridedami be didesnių sunkumų.

Uml diagrama:



Pav. 7 Decorator programavimo šablonas

Kodas:

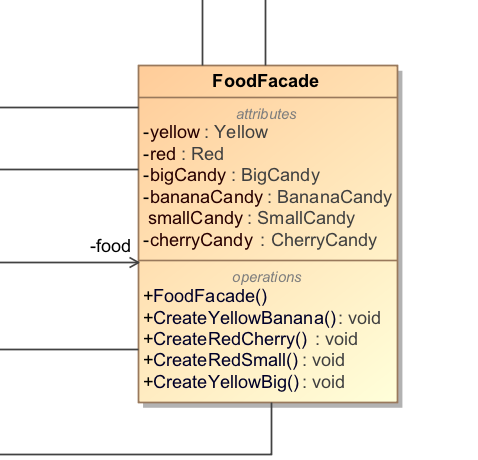
|  |
| --- |
| Gun.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class Gun : Weapon  {  public override int Damage()  {  return 10;  }  public override string Shoot()  {  return "Shooting simple bullet";  }  }  } |
| WeaponDecorator.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  abstract class WeaponDecorator : Weapon  {  protected Weapon weapon = null;  public override int Damage()  {  return weapon.Damage();  }  public override string Shoot()  {  return weapon.Shoot();  }  }  } |
| Weapon.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  abstract class Weapon  {  public abstract int Damage();  public abstract string Shoot();  }  } |
| SpeedTrap.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class SpeedTrap : WeaponDecorator  {  public SpeedTrap(Weapon weapon)  {  this.weapon = weapon;  }  public override int Damage()  {  return base.Damage() + 10;  }  public override string Shoot()  {  if (this.weapon != null)  {  return base.Shoot() + " and slowing bullet";  }  return string.Empty;  }  }  } |
| Cannon.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class Cannon : WeaponDecorator  {  public Cannon(Weapon weapon)  {  this.weapon = weapon;  }  public override int Damage()  {  return base.Damage() + 100;  }  public override string Shoot()  {  if (weapon != null)  {  return base.Shoot() + " and instant-kill bullets";  }  return string.Empty;  }  }  } |

## Facade

Panaudojimo tikslas:

Facade programavimo šabloną panaudojome, kadangi reikia sukurti tam tikro tipo ir tam tikros spalvos saldainius, taip palengvindami jų sukūrimą.

Uml diagrama:



Pav. 8 Facade programavimo šablonas

Kodas:

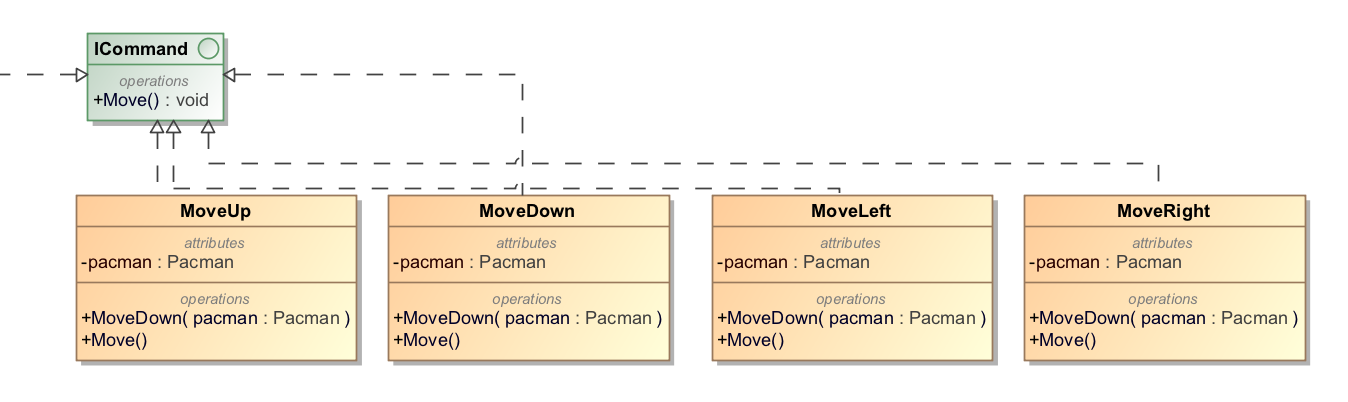
|  |
| --- |
| FoodFacade.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using Pacman.Classes.Candies;  using Pacman.Classes.Colors;  namespace Pacman.Classes  {  class FoodFacade  {  private BananaCandy bananaCandy;  private BigCandy bigCandy;  private SmallCandy smallCandy;  private CherryCandy cherryCandy;  private Yellow yellow;  private Red red;  public FoodFacade()  {  bananaCandy = new BananaCandy();  bigCandy = new BigCandy();  smallCandy = new SmallCandy();  cherryCandy = new CherryCandy();  yellow = new Yellow();  red = new Red();  }  public void CreateYellowBanana()  {  Debug.WriteLine("Creating colorful candy");  bananaCandy.CreateCandy();  yellow.CreateColor();  Debug.WriteLine("");  }  public void CreateRedCherry()  {  Debug.WriteLine("Creating colorful candy");  cherryCandy.CreateCandy();  red.CreateColor();  Debug.WriteLine("");  }  public void CreateRedSmall()  {  Debug.WriteLine("Creating colorful candy");  smallCandy.CreateCandy();  red.CreateColor();  Debug.WriteLine("");  }  public void CreateYellowBig()  {  Debug.WriteLine("Creating colorful candy");  bigCandy.CreateCandy();  yellow.CreateColor();  Debug.WriteLine("");  }  }  } |

## Command

Panaudojimo tikslas:

Command šabloną panaudojome tam, kad paslėptume komandų apdorojimų logiką. Tai mums leis vėliau lengvai pridėti naujų kamandų ir jas apdoroti objektiškai.

Uml diagrama:

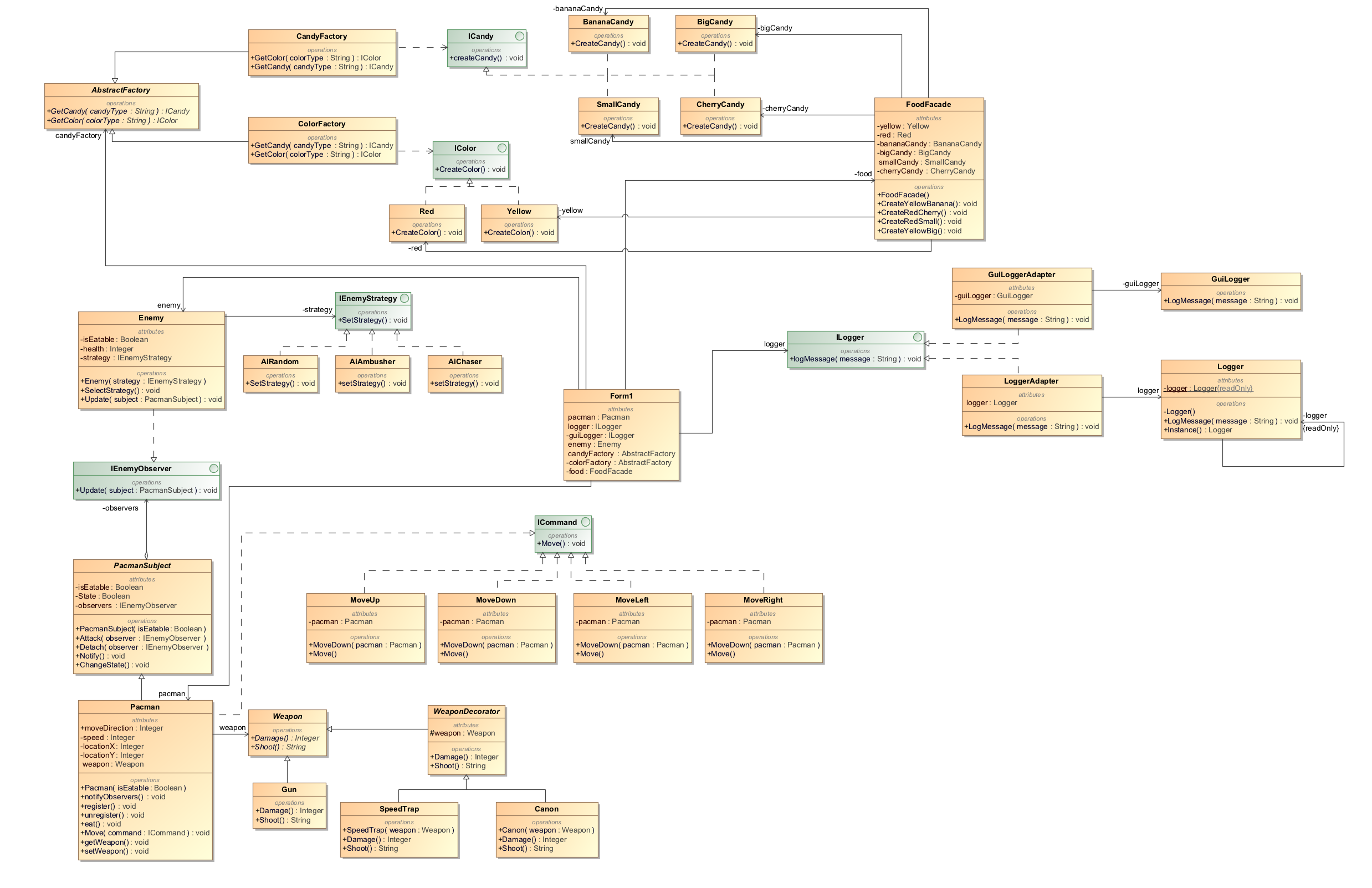


Pav. 9 Command programavimo šablonas

Kodas:

|  |
| --- |
| ICommand.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  interface ICommand  {  void Move();  }  } |
| MoveUp.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class MoveUp : ICommand  {  private Pacman pacman;  public MoveUp(Pacman pacman)  {  this.pacman = pacman;  }  public void Move()  {  pacman.moveDirection = 0;  Debug.WriteLine("Pacman changed direction to: up");  }  }  } |
| MoveRight.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class MoveRight : ICommand  {  private Pacman pacman;  public MoveRight(Pacman pacman)  {  this.pacman = pacman;  }  public void Move()  {  pacman.moveDirection = 1;  Debug.WriteLine("Pacman changed direction to: right");  }  }  } |
| MoveLeft.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class MoveLeft : ICommand  {  private Pacman pacman;  public MoveLeft(Pacman pacman)  {  this.pacman = pacman;  }  public void Move()  {  pacman.moveDirection = 3;  Debug.WriteLine("Pacman changed direction to: left");  }  }  } |
| MoveDown.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class MoveDown : ICommand  {  private Pacman pacman;  public MoveDown(Pacman pacman)  {  this.pacman = pacman;  }  public void Move()  {  pacman.moveDirection = 2;  Debug.WriteLine("Pacman changed direction to: down");  }  }  } |

# 1 ir 2 Labaratorinių darbų klasių diagramos



Pav. 10 Bendra UML diagrama

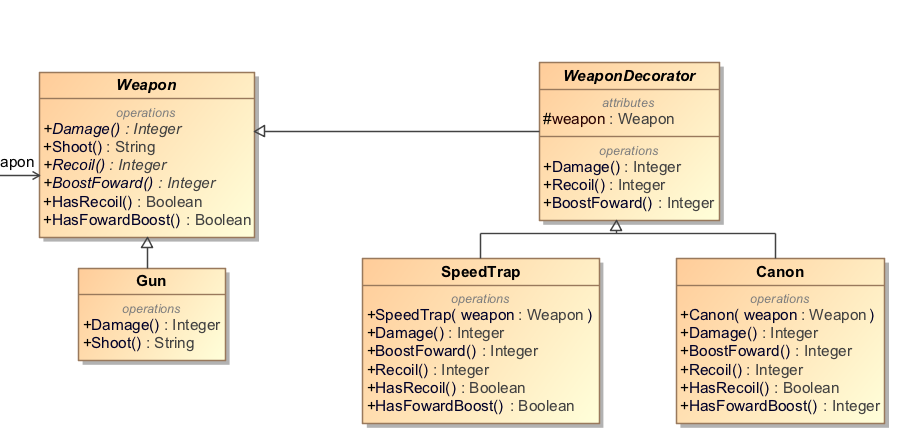
# Trečias labaratorinis darbas

## Template Method

Panaudojimo tikslas:

Panaudojome kartu su Decorator šablonu.

Uml diagrama:



Pav. 11 Template Method programavimo šablonas

Kodas:

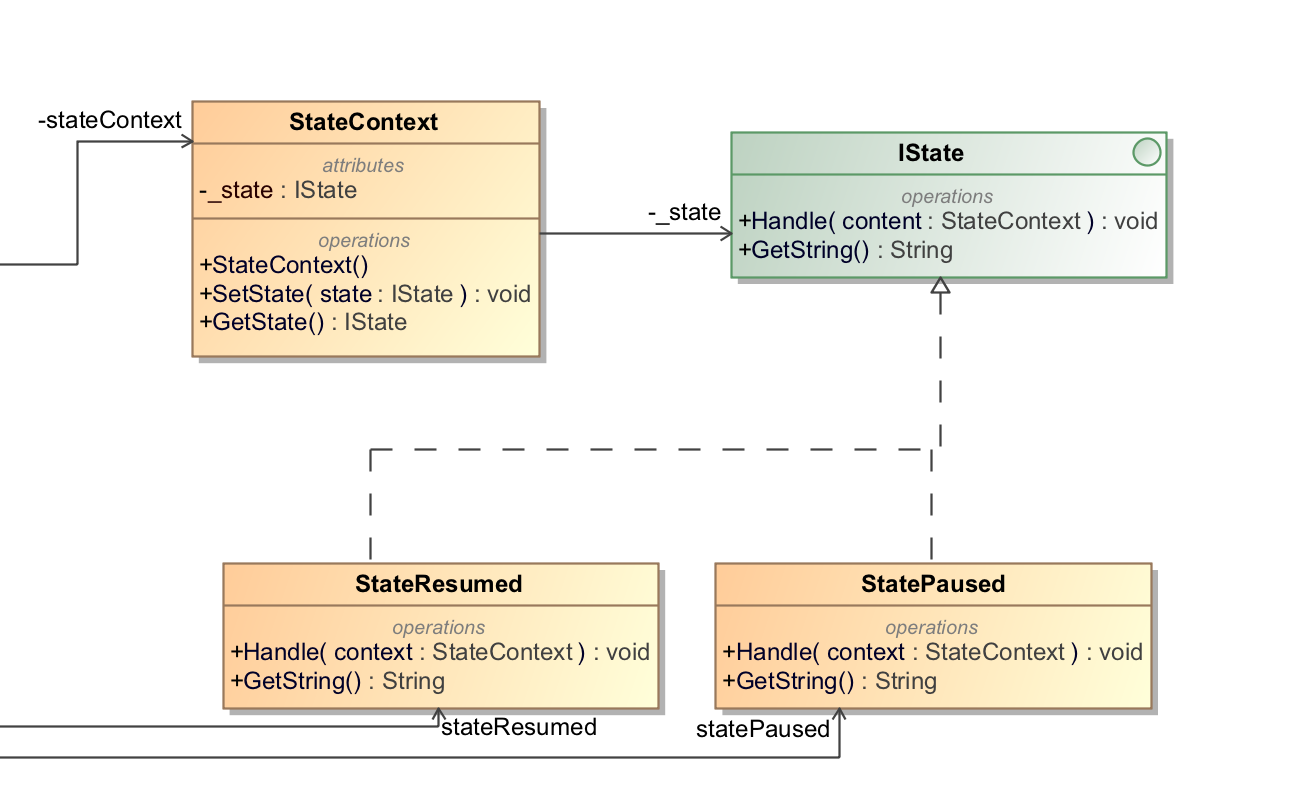
|  |
| --- |
| Weapon.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  abstract class Weapon  {  public abstract int Damage();  public string Shoot()  {  if (HasRecoil())  {  return "Shoots with " + Damage() + " damage, sends Pacman " + Recoil() + " steps back";  }  if (HasFowardBoost())  {  return "Shoots with " + Damage() + " damage, boosts Pacman " + BoostFoward() + " steps foward";  }  return "Shoots with " + Damage() + " damage";  }  public abstract int Recoil();  public abstract int BoostFoward();  public virtual bool HasRecoil() { return true; }  public virtual bool HasFowardBoost() { return true; }  }  } |
| WeaponDecorator.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  abstract class WeaponDecorator : Weapon  {  protected Weapon weapon = null;  public override int Damage()  {  return weapon.Damage();  }  public override int Recoil()  {  return weapon.Recoil();  }  public override int BoostFoward()  {  return weapon.BoostFoward();  }  }  } |
| Gun.cs |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class Gun : Weapon  {  public override int Damage()  {  return 10;  }  public override int Recoil()  {  return 0;  }  public override int BoostFoward()  {  return 0;  }  public override bool HasFowardBoost()  {  return false;  }  public override bool HasRecoil()  {  return false;  }  }  } |
| SpeedTrap.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class SpeedTrap : WeaponDecorator  {  public SpeedTrap(Weapon weapon)  {  this.weapon = weapon;  }  public override int BoostFoward()  {  return 2;  }  public override int Damage()  {  return base.Damage() + 10;  }  public override int Recoil()  {  return 0;  }  public override bool HasRecoil()  {  return false;  }  public override bool HasFowardBoost()  {  return true;  }  }  } |
| Canon.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class Cannon : WeaponDecorator  {  public Cannon(Weapon weapon)  {  this.weapon = weapon;  }  public override int BoostFoward()  {  return 0;  }  public override int Damage()  {  return base.Damage() + 100;  }  public override int Recoil()  {  return 2;  }  public override bool HasRecoil()  {  return true;  }  public override bool HasFowardBoost()  {  return false;  }  }  } |

## State

Panaudojimo tikslas:

Pakeičia žaidimo būseną iš sustabdyto į žaidžiama ir atvirkščiai.

Uml diagrama:



Pav. 12 State programavimo šablonas

Kodas:

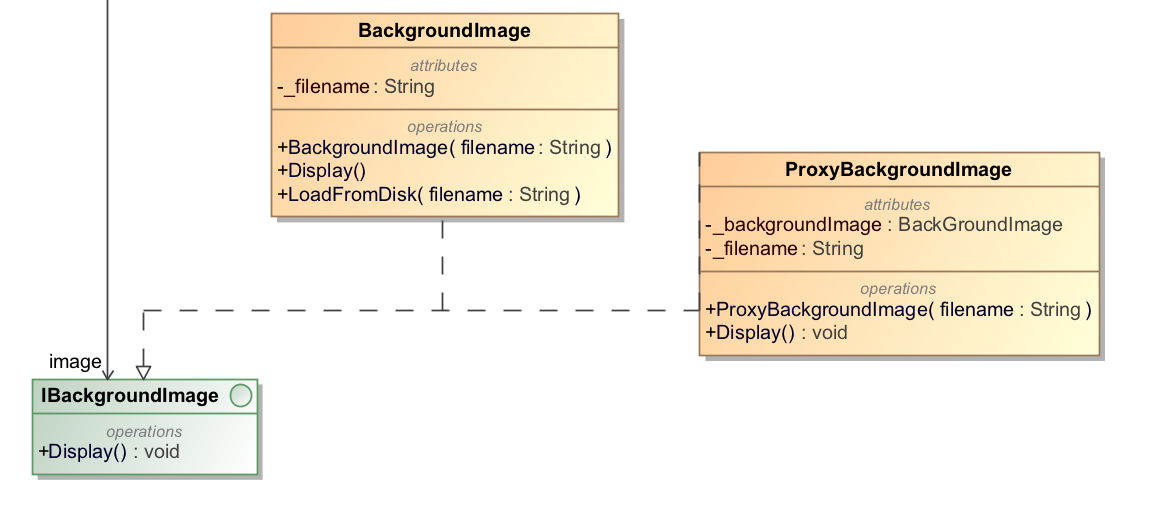
|  |
| --- |
| StateContext.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class StateContext  {  private IState \_state;  public StateContext()  {  \_state = null;  }  public void SetState(IState state)  {  \_state = state;  }  public IState GetState()  {  return \_state;  }  }  } |
| StateResumed.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class StateResumed : IState  {  public void Handle(StateContext context)  {  Console.WriteLine("The game is in resumed state");  context.SetState(this);  }  public String GetString()  {  return "Resumed state";  }  }  } |
| iState |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  interface IState  {  void Handle(StateContext context);  String GetString();  }  } |
| StatePaused.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class StatePaused : IState  {  public void Handle(StateContext context)  {  Console.WriteLine("The game is in paused state");  context.SetState(this);  }  public String GetString()  {  return "Paused state";  }  }  } |

## Proxy

Panaudojimo tikslas:

Rodo background paveikslėlį žaidimo lentai.

Uml diagrama:



Pav. 13 Proxy programavimo šablonas

Kodas:

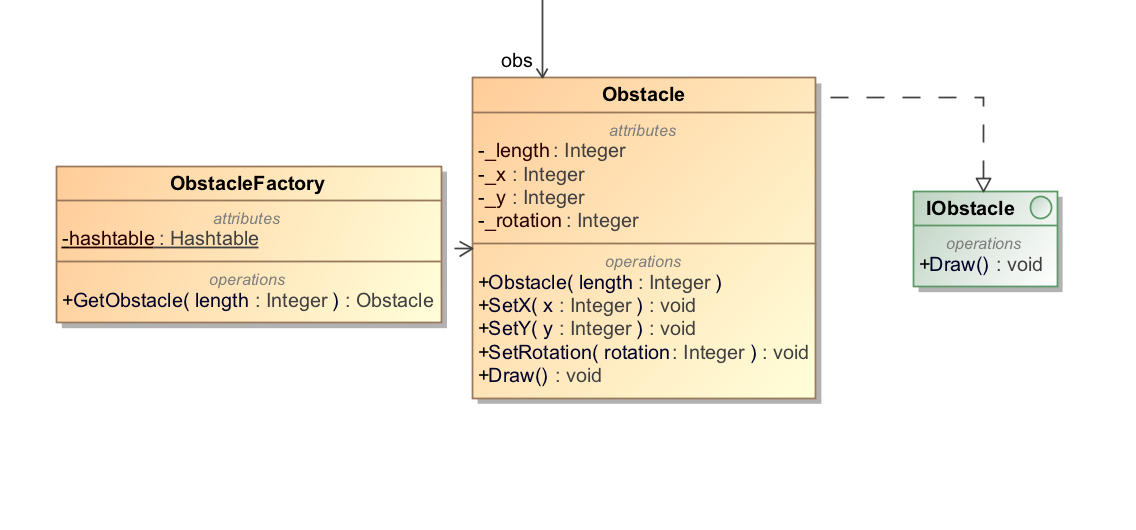
|  |
| --- |
| IBackgroundImage.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  interface IBackgroundImage  {  void Display();  }  } |
| BackgroundImage.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class BackgroundImage : IBackgroundImage  {  private String \_filename;  public BackgroundImage(String filename)  {  \_filename = filename;  LoadFromDisk(filename);  }  public void Display()  {  Console.WriteLine("Displaying background image: " + \_filename);  }  public void LoadFromDisk(String filename)  {  Console.WriteLine("Loading image from disk: " + \_filename);  }  }  } |
| ProxyBackgroundImage.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class ProxyBackgroundImage : IBackgroundImage  {  private BackgroundImage \_backgroundImage;  private String \_filename;  public ProxyBackgroundImage(String filename)  {  \_filename = filename;  }  public void Display()  {  if (\_backgroundImage == null)  {  \_backgroundImage = new BackgroundImage(\_filename);  }  \_backgroundImage.Display();  }  }  } |

## Flyweight

Panaudojimo tikslas:

Generuoti kliūtis, remiantis jų ilgiu.

Uml diagrama:



Pav. 14 Flyweight programavimo šablonas

Kodas:

|  |
| --- |
| ObstacleFactory.cs |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class ObstacleFactory  {  private static Hashtable hashtable = new Hashtable();  public static Obstacle GetObstacle(int length)  {  Obstacle obs = (Obstacle) hashtable[length];  if (obs == null)  {  obs = new Obstacle(length);  hashtable.Add(length, obs);  Console.WriteLine("Creating obstacle of length:" + length);  }  return obs;  }  }  } |
| Obstacle.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class Obstacle : IObstacle  {  private int \_length, \_x, \_y, \_rotation;  public Obstacle(int length)  {  \_length = length;  }  public void SetX(int x)  {  \_x = x;  }  public void SetY(int y)  {  \_y = y;  }  public void SetRotation(int rotation)  {  \_rotation = rotation;  }  public void Draw()  {  Console.WriteLine("Obstacle: Draw() [Length: " + \_length + ", x: " + \_x  + ", y:" + \_y + ", rotation: " + \_rotation);  }  }  } |
| IObstacle.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  interface IObstacle  {  void Draw();  }  } |

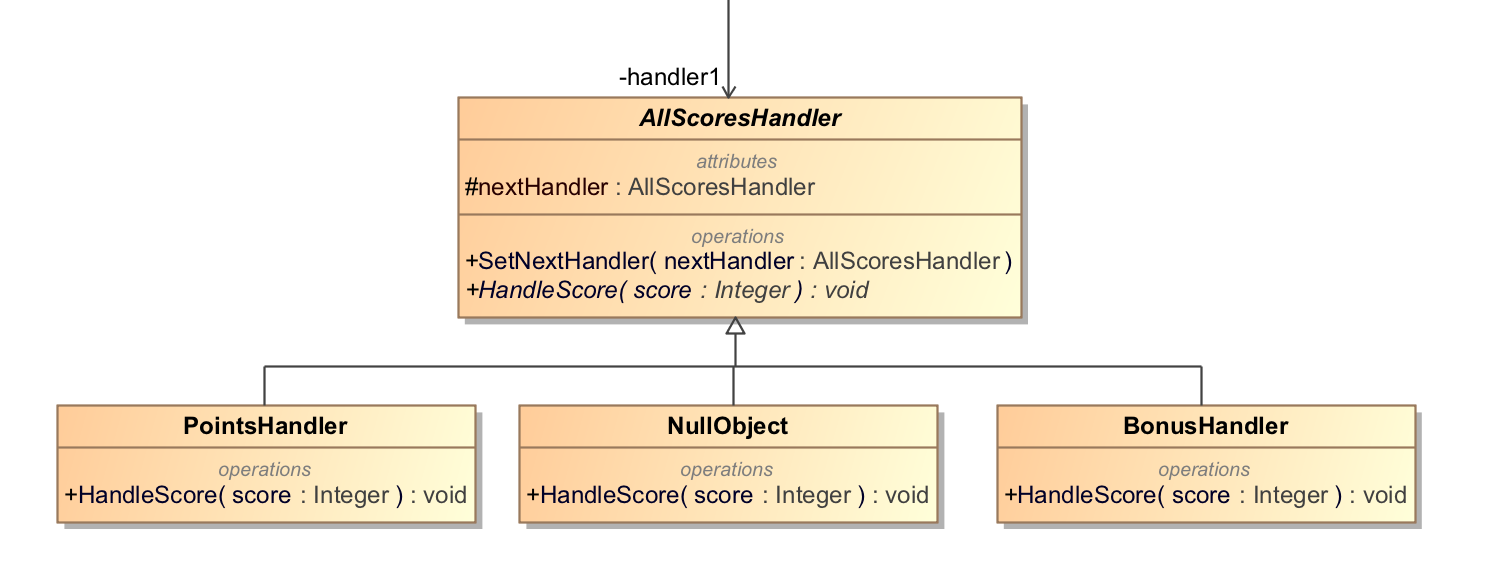
# Ketvirtas labaratorinis darbas

## Chain Of Responsibility

Panaudojimo tikslas:

Taškų skaičiavimui.

Uml diagrama:



Pav. 15 Chain Of Responsibility programavimo šablonas

Kodas:

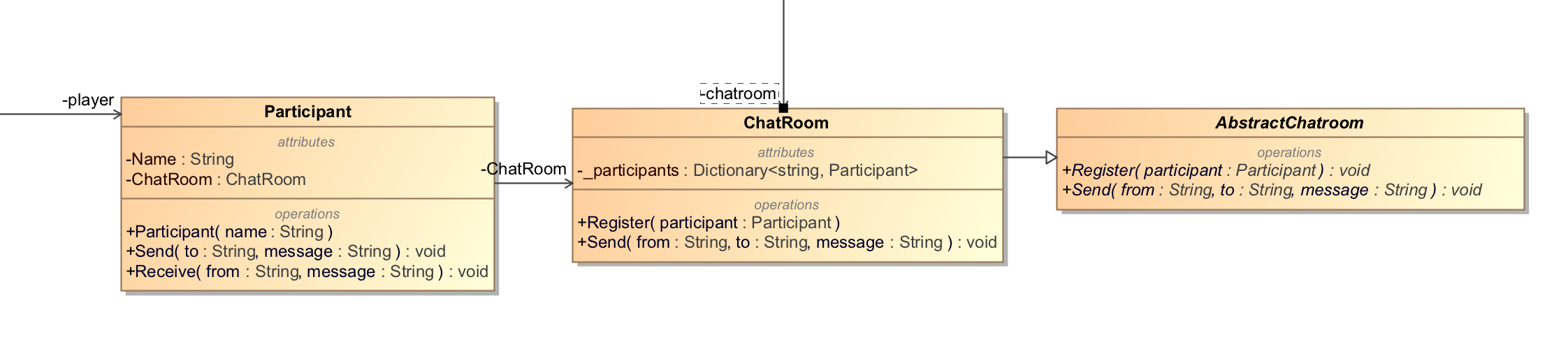
|  |
| --- |
| AllScoreHandler.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  abstract class AllScoresHandler  {  protected AllScoresHandler nextHandler;  public void SetNextHandler(AllScoresHandler nextHandler)  {  this.nextHandler = nextHandler;  }  public abstract void HandleScore(int score);  }  } |
| PointsHandler.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class PointsHandler : AllScoresHandler  {  public override void HandleScore(int score)  {  if (score < 10)  {  Console.WriteLine("PointsHandler handled request: " + score);  Highscore.Instance.score += score;  }  else if (nextHandler != null)  {  nextHandler.HandleScore(score);  }  }  }  } |
| NullObject.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class NullObject : AllScoresHandler  {  public override void HandleScore(int score)  {  if (score == 0)  {  Console.WriteLine("NullObject handled request: " + score);  }  else if (nextHandler != null)  {  nextHandler.HandleScore(score);  }  }  }  } |
| BonusHandler.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class BonusHandler : AllScoresHandler  {  public override void HandleScore(int score)  {  if (score >= 10)  {  Console.WriteLine("BonusHandler handled request: " + score);  Highscore.Instance.score += score \* 2;  }  else if (nextHandler != null)  {  nextHandler.HandleScore(score);  }  }  }  } |

## Mediator

Panaudojimo tikslas:

Panaudota pokalbių kambario implementavimui.

Uml diagrama:



Pav. 16 Mediator programavimo šablonas

Kodas:

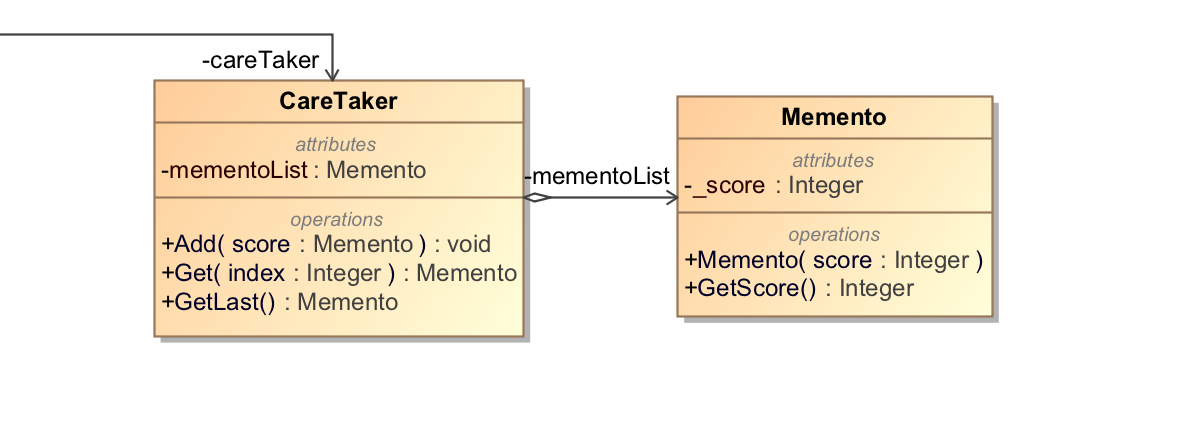
|  |
| --- |
| Participant.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class Participant  {  public Chatroom Chatroom { set; get; }  public string Name { get; }  public Participant(string name)  {  Name = name;  }  public void Send(string to, string message)  {  Chatroom.Send(Name, to, message);  }  public void Receive(string from, string message)  {  Console.WriteLine("Message from {0} to {1}: '{2}'", from, Name, message);  }  }  } |
| ChatRoom.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class Chatroom : AbstractChatroom  {  private Dictionary<string, Participant> \_participants =  new Dictionary<string, Participant>();  public override void Register(Participant participant)  {  if (!\_participants.ContainsValue(participant))  {  \_participants[participant.Name] = participant;  }  participant.Chatroom = this;  }  public override void Send(string from, string to, string message)  {  Participant participant = \_participants[to];  participant?.Receive(from, message);  }  }  } |
| AbstractChatroom.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  abstract class AbstractChatroom  {  public abstract void Register(Participant participant);  public abstract void Send(string from, string to, string message);  }  } |

## Memento

Panaudojimo tikslas:

Žaidėjo taškų saugojimas, vėliasniam naudojimui.

Uml diagrama:



Pav. 17 Memento programavimo šablonas

Kodas:

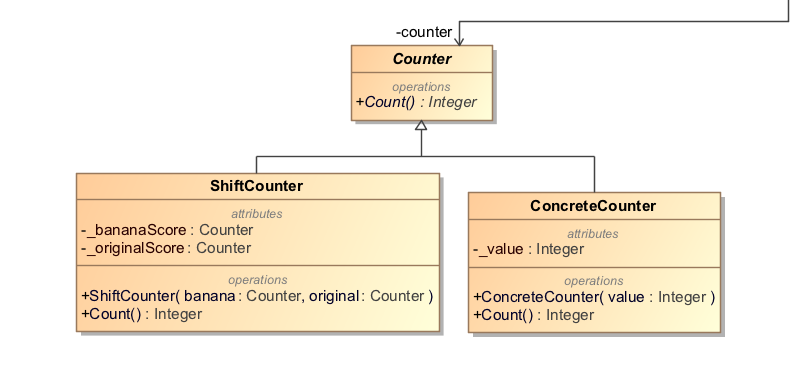
|  |
| --- |
| CareTaker.cs |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class CareTaker  {  private List<Memento> mementoList = new List<Memento>();  public void Add(Memento score)  {  mementoList.Add(score);  }  public Memento Get(int index)  {  return mementoList[index];  }  public Memento GetLast()  {  return mementoList[mementoList.Count-1];  }  }  } |
| Memento.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class Memento  {  private int \_score;  public Memento(int score)  {  \_score = score;  }  public int GetScore()  {  return \_score;  }  }  } |

## Interpreter

Panaudojimo tikslas:

Kai PACMAN suvalgo geltoną banana, prie rezultato pridedama 10 taškų

Uml diagrama:

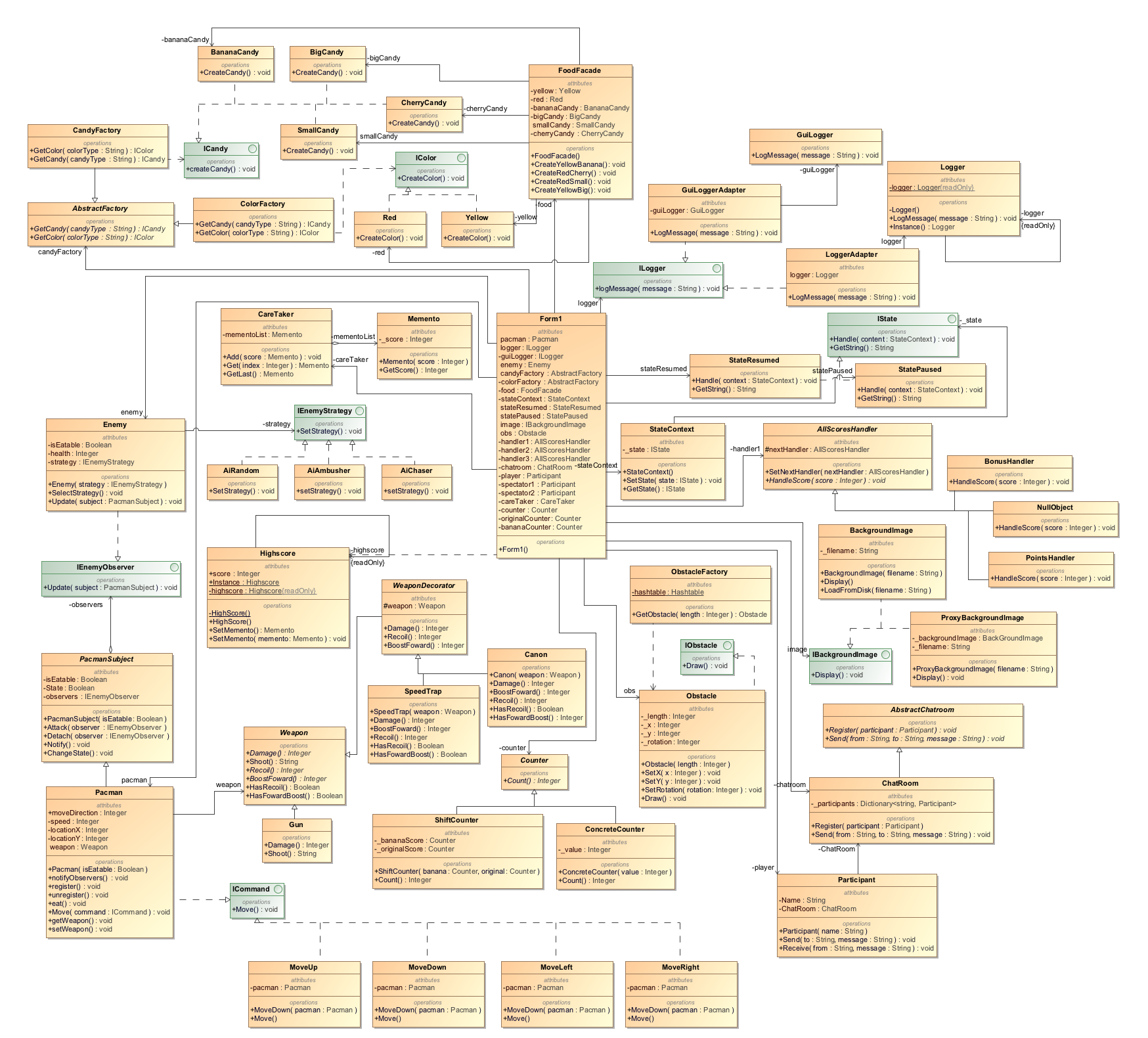


Pav. 18 Interpreter programavimo šablonas

Kodas:

|  |
| --- |
| Counter.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  abstract class Counter  {  public abstract int Count();  }  } |
| ShiftCounter.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class ShiftCounter : Counter  {  private Counter \_bananaScore, \_originalScore;  public ShiftCounter(Counter banana, Counter original)  {  \_bananaScore = banana;  \_originalScore = original;  }  public override int Count()  {  return \_bananaScore.Count() + (\_originalScore.Count() << 1);  }  }  } |
| ConcreteCounter.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Pacman.Classes  {  class ConcreteCounter : Counter  {  private int \_value;  public ConcreteCounter(int value)  {  \_value = value;  }  public override int Count()  {  return \_value;  }  }  } |

# Labaratorinių darbų klasių diagramos



Pav. 19 Galutinė UML Shema

# Išvados

Programos įgyvendinimui panaudoti programavimo šablonai: Observer, Strategy, Factory, Absract Factory, Singleton, Facade, Command, Adapter ir Decorator. Kai kuriose vietose šie šablonai padėjo sumažinti programos sudėtingumą, padaryti kodą labiau skaitomą ir lengviau modifikuojamą.

Tačiau, supratome, kad perteklinis šablonų panaudojimas, apsunkina kodo rašymą ir pailgina įgyvendinimą.