

Informatikos fakultetas

**T120B516 Objektinis programų projektavimas**

**Laboratorinių darbų ataskaita**

**Žaidimas „Tower Defence“**

|  |  |  |
| --- | --- | --- |
|  | Studentai: | Mangirdas Kazlauskas, IFF-4/1 Simonas Baltulionis, IFF-4/1  Tautvydas Traška, IFF-4/2 |
| Dėstytojai: | Andrej Ušaniov  Dominykas Barisas |

Kaunas 2017

Turinys

[1. Projekto aprašymas 4](#_Toc496569265)

[2. Laboratorinis darbas Nr. 1 (1 dalis) 5](#_Toc496569266)

[2.1. Pradinė klasių diagrama 5](#_Toc496569267)

[2.2. Vienintelio objekto (Singleton) projektavimo pavyzdys 6](#_Toc496569268)

[2.2.1. Projektavimo šablono realizacija 6](#_Toc496569269)

[2.2.2. Projektavimo šablono naudojimo pagrindimas 7](#_Toc496569270)

[2.3. Abstraktaus fabriko (Abstract factory) projektavimo pavyzdys (Simonas) 7](#_Toc496569271)

[2.3.1. Projektavimo šablono realizacija 7](#_Toc496569272)

[2.3.2. Projektavimo šablono naudojimo pagrindimas 9](#_Toc496569273)

[2.4. Fabriko metodo (Factory method) projektavimo pavyzdys (Mangirdas) 9](#_Toc496569274)

[2.4.1. Projektavimo šablono realizacija 9](#_Toc496569275)

[2.4.2. Projektavimo šablono naudojimo pagrindimas 11](#_Toc496569276)

[2.5. Strategijos (Strategy) projektavimo pavyzdys (Tautvydas) 11](#_Toc496569277)

[2.5.1. Projektavimo šablono realizacija 11](#_Toc496569278)

[2.5.2. Projektavimo šablono naudojimo pagrindimas 13](#_Toc496569279)

[3. Laboratoris darbas Nr. 1 (2 dalis) 14](#_Toc496569280)

[3.1. Klasių diagrama 14](#_Toc496569281)

[3.2. Pritaikančio objekto (Adapter) projektavimo pavyzdys 15](#_Toc496569282)

[3.2.1. Projektavimo šablono realizacija 15](#_Toc496569283)

[3.2.2. Projektavimo šablono naudojimo pagrindimas 16](#_Toc496569284)

[3.3. Dekoratoriaus (Decorator) projektavimo pavyzdys (Simonas) 17](#_Toc496569285)

[3.3.1. Projektavimo šablono realizacija 17](#_Toc496569286)

[3.3.2. Projektavimo šablono naudojimo pagrindimas 18](#_Toc496569287)

[3.4. Komandos (Command) projektavimo pavyzdys (Mangirdas) 18](#_Toc496569288)

[3.4.1. Projektavimo šablono realizacija 18](#_Toc496569289)

[3.4.2. Projektavimo šablono naudojimo pagrindimas 19](#_Toc496569290)

[3.5. Tilto (Bridge) projektavimo pavyzdys (Tautvydas) 19](#_Toc496569291)

[3.5.1. Projektavimo šablono realizacija 19](#_Toc496569292)

[3.5.2. Projektavimo šablono naudojimo pagrindimas 22](#_Toc496569293)

# Projekto aprašymas

Laboratorinių darbų metu kuriamas žaidimas „Tower Defence“. „Tower Defence“ – tai vienas iš strateginių vaizdo žaidimų žanrų, kurio tikslas – apginti tam tikrą teritoriją nuo vis atsirandančių priešų. Gynybai nuo priešų statomi įvairių tipų pastatai – gynybinės struktūros, kurios atakuoja priešus, judančius tam tikru keliu iki turimos apginti teritorijos. Taip pat žaidime egzistuoja įvairių tipų priešai su skirtingais gyvybiniais parametrais, pvz., gyvybės taškais. Vienas iš tokio tipo žaidimų pavyzdžių pateiktas 1 paveikslėlyje.

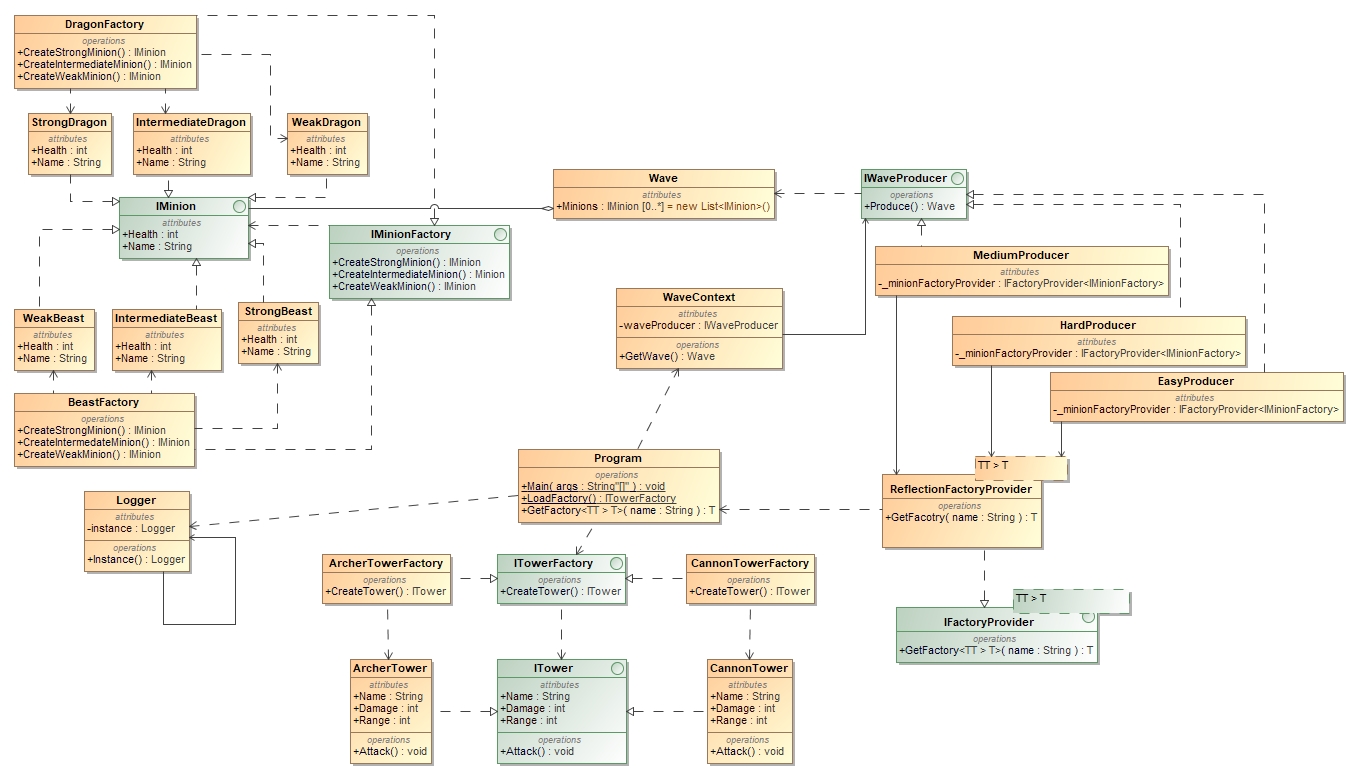


1 pav. Tower Defence tipo žaidimo pavyzdys (šaltinis: <https://i.pinimg.com/originals/02/f2/3f/02f23faf46a076f590a4f4167f365e4c.jpg>)

Tokio tipo žaidimą nutartą kurti dėl to, kad jį labai lengva plėtoti, skirtingi pastatų bei priešų tipai suteikia daug laisvės įvairių programinių struktūrų realizavimui bei dizaino šablonų pritaikymui.

# Laboratorinis darbas Nr. 1 (1 dalis)

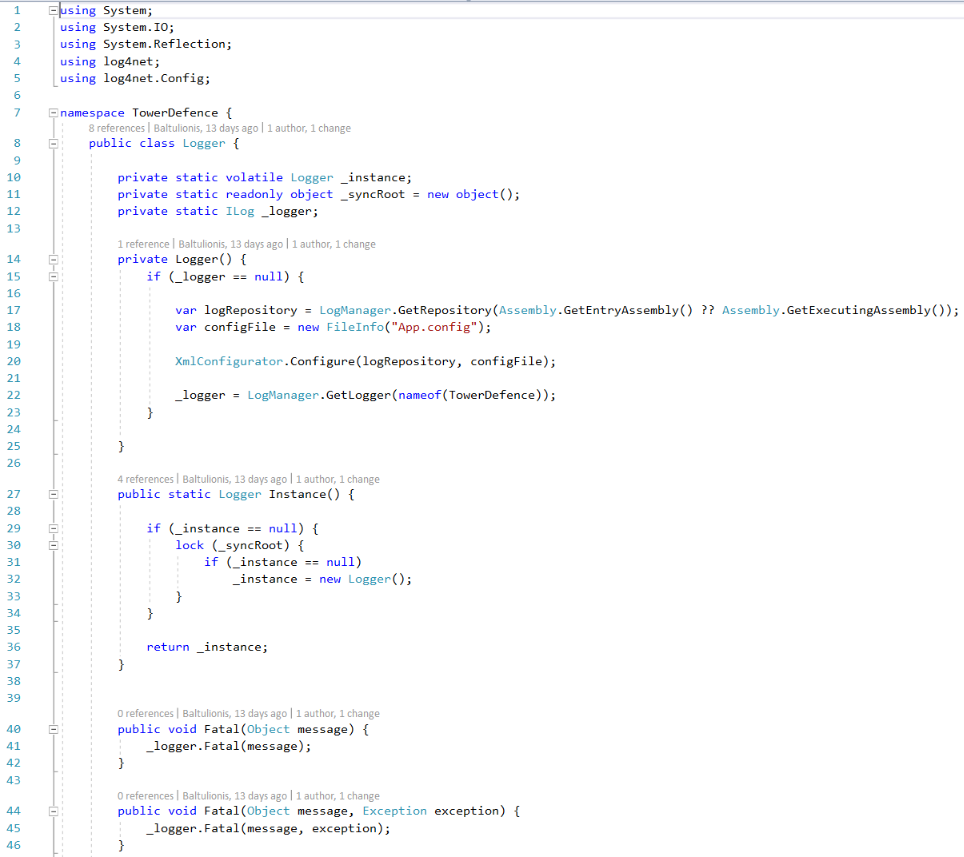
## Pradinė klasių diagrama

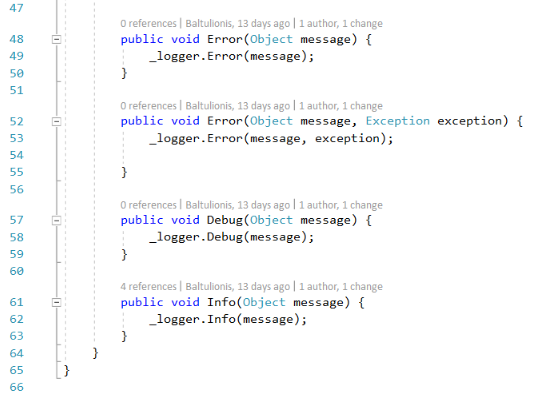
2 paveikslėlyje pateikta pradinė kuriamo žaidimo klasių diagrama.

2 pav. Pradinė klasių diagrama

## Vienintelio objekto (Singleton) projektavimo pavyzdys

### Projektavimo šablono realizacija





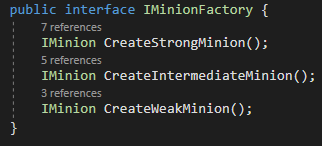
3 pav. Vienintelio objekto (Singleton) realizacija

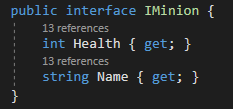
### Projektavimo šablono naudojimo pagrindimas

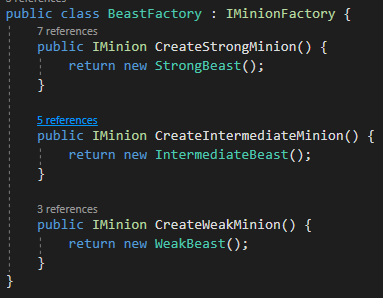
Vienintelio objekto (Singleton) projektavimo šablonas yra naudojamas tada, kai kuri nors klasė projekte yra pastoviai naudojama ir pakartotinai reikia sukurti tos klasės objektą, kas kainuoja resursų. Mūsų atveju tokia klasė – Logger klasė, kuri skirta programos pranešimų išvedimui. Kadangi ši klasė turi daug metodų, savyje talpina nemažai logikos, todėl tokio objekto sukūrimas kas kartą kainuotų nemažai resursų, todėl tikslingai panaudotas Singleton šablonas šią problemą išsprendžia.

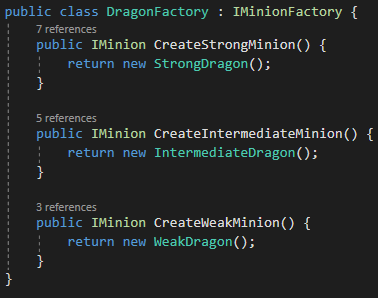
## Abstraktaus fabriko (Abstract factory) projektavimo pavyzdys (Simonas)

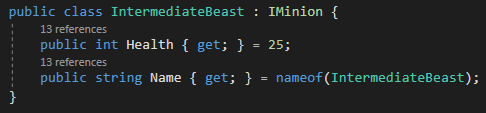
### Projektavimo šablono realizacija

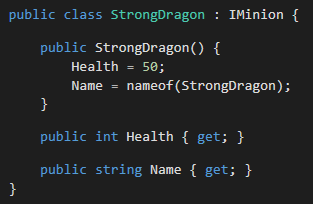












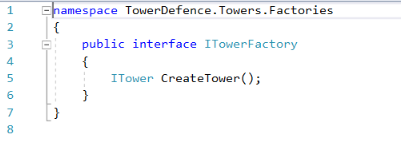
4 pav. Abstraktaus fabriko (Abstract factory) realizacija

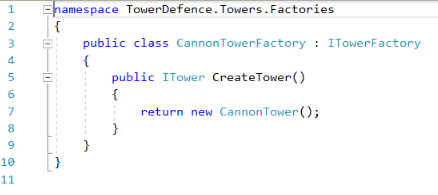
### Projektavimo šablono naudojimo pagrindimas

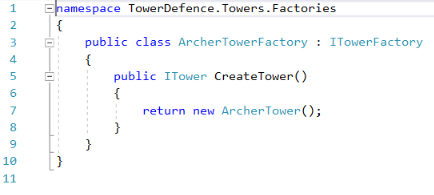
„Abstract factory“ šablonas yra skirtas sukurti objektų šeimoms. Mūsų atveju šablonas naudojamas sukurti priešų objektų šeimoms – galima sukurti trijų tipų žvėris arba tų pačių trijų tipų drakonus. Šie abstraktūs fabrikai gali būti ir bus panaudojami kuriant įvairaus tipo priešų bangas skirtinguose lygiuose.

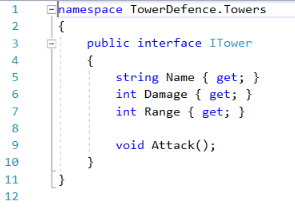
## Fabriko metodo (Factory method) projektavimo pavyzdys (Mangirdas)

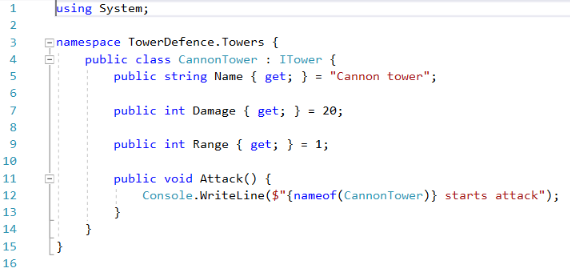
### Projektavimo šablono realizacija

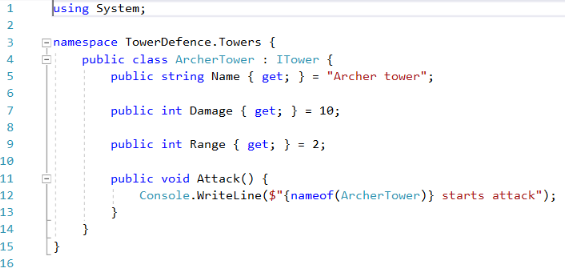


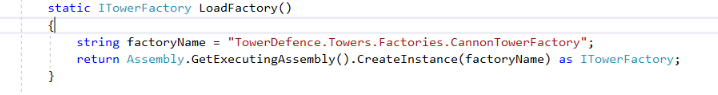












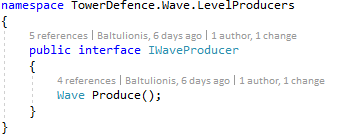
5 pav. Fabriko metodo (Factory method) realizacija

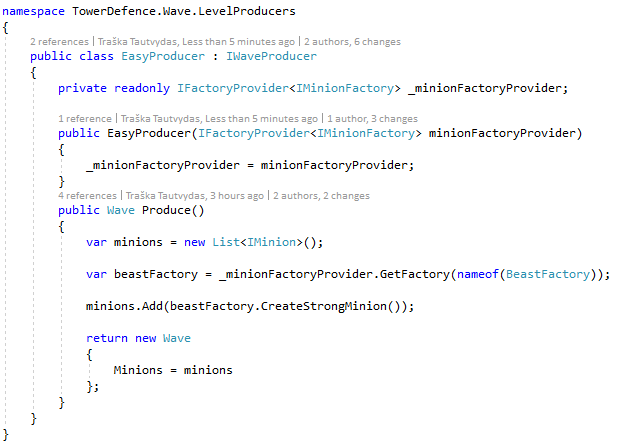
### Projektavimo šablono naudojimo pagrindimas

Fabriko metodo (Factory method) projektavimo šablono paskirtis – palengvinti objektų sukūrimą, viduje paslepiant pačią objekto sukūrimo logiką, o vietoj to naudojant sukurtą sąsają. Taip pat šablonas padeda sukurti objektus, tiksliai neapibrėžiant jų klasės – remiantis šia savybe, buvo sukurtos dvi sąsajos: gynybinių pastatų sąsaja ITower, bei gynybinių pastatų objektų sukūrimo sąsaja ITowerFactory. Naudojantis ITowerFactory sąsaja, galima lengvai sukurti skirtingų tipų fabrikus, kuriančius skirtingų ITower tipų pastatus. Toks projektavimo sprendimas leidžia lengvai plėsti galimų sukurti skirtingų tipų pastatų sąrašą, neapsunkinant ir nekeičiant jau esamų bei naujų pastatų objektų sukūrimo metodų.

## Strategijos (Strategy) projektavimo pavyzdys (Tautvydas)

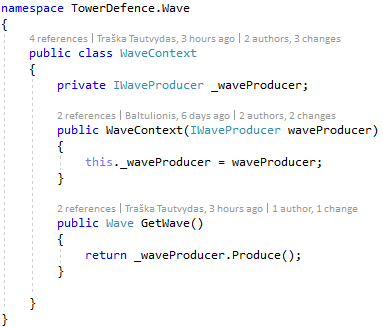
### Projektavimo šablono realizacija











6 pav. Strategijos (Strategy) realizacija

### Projektavimo šablono naudojimo pagrindimas

„Strategy“ šablonas yra skirtas keisti objekto elgseną ar algoritmą programos vykdymo metu. Mūsų atveju šis šablonas naudojamas siekiant parinkti priešų bangų sudėtingumą priklausomai nuo žaidimo lygio. Atsakinga priešų bangos konteksto klasė, objekto kūrimo metu gali priimti skirtingas strategijų implementacijas, lemiančias žaidimo įvairovę ir sudėtingumą.

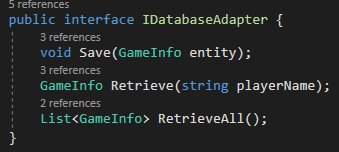
# Laboratoris darbas Nr. 1 (2 dalis)

## C:\Users\mangi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ClassDiagram.jpgKlasių diagrama

7 pav. 1 laboratorinio darbo klasių diagrama po antrosios dallies atlikimo

## Pritaikančio objekto (Adapter) projektavimo pavyzdys

### Projektavimo šablono realizacija





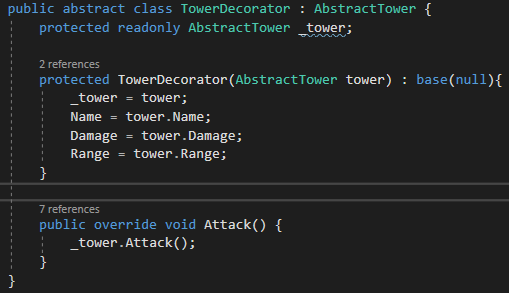
8 pav. Dekoratoriaus (Decorator) realizacija

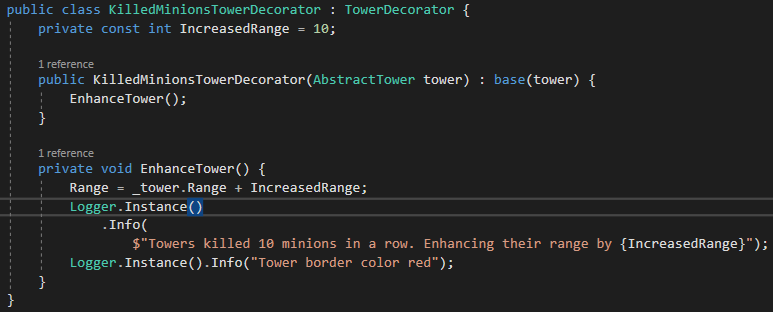
### Projektavimo šablono naudojimo pagrindimas

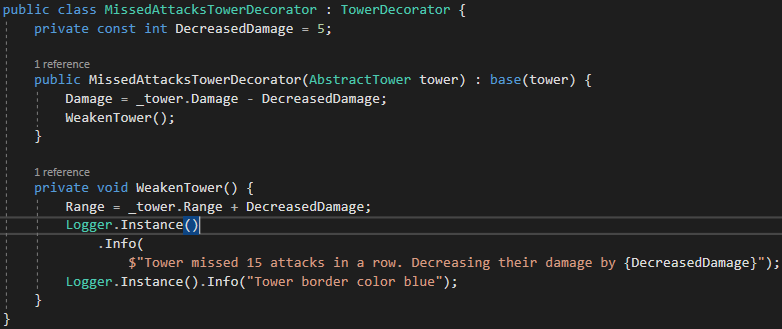
Mūsų žaidimui reikalinga duomenų bazė išsaugoti žaidėjų rezultatus, tačiau nenorime, kad žaidimas būtų priklausomas nuo konkrečios duomenų bazės ir bibliotekos, todėl sukūrėme savo sąsają, kurią realizavome naudodami Adapter šabloną. Tam, kad pademonstruotume Adapter šablono pritaikymą, realizavome RavenDB ir LiteDB sąsajas.

## Dekoratoriaus (Decorator) projektavimo pavyzdys (Simonas)

### Projektavimo šablono realizacija







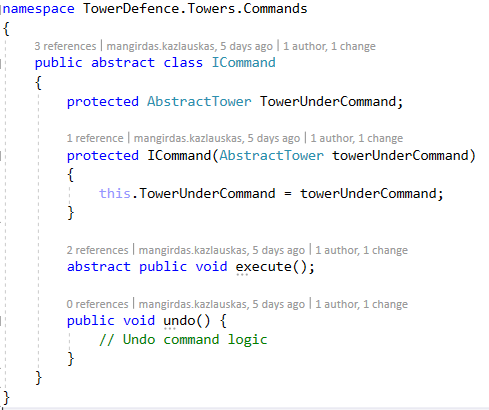
9 pav. Dekoratoriaus (Decorator) realizacija

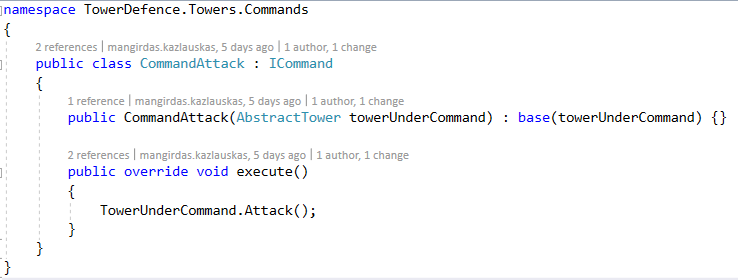
### Projektavimo šablono naudojimo pagrindimas

Žaidime priklausomai nuo to, kaip statome bokštus, priklauso žaidimo rezultatai. Jeigu bokštai sunaikina tam tikrą skaičių priešų iš eilės, įgauna papildomų savybių bei pakeičia spalvą arba nepataikius daug kartų iš eilės, bokštai yra susilpninami bei taip pat keičia spalvą. Tą reikia atlikti žaidimo veikimo metu (angl. runtime). Tam tinkamas Decorator šablonas, kuris leidžia praplėsti bokšto savybes, pridėti naujų funkcijų nekeičiant esamo bokšto realizacijos. Tam panaudojamas esamas bokšto objektas su jau esamomis savybėmis bei pridedama (dekoruojama) naujų funkcijų bei savybių.

## Komandos (Command) projektavimo pavyzdys (Mangirdas)

### Projektavimo šablono realizacija





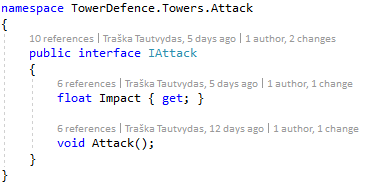
10 pav.Komandos (Command) realizacija

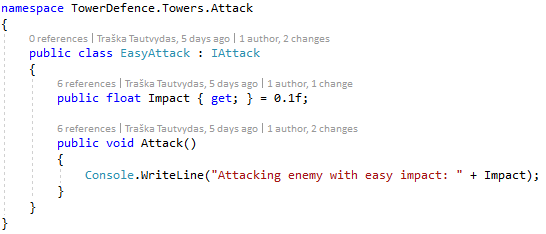
### Projektavimo šablono naudojimo pagrindimas

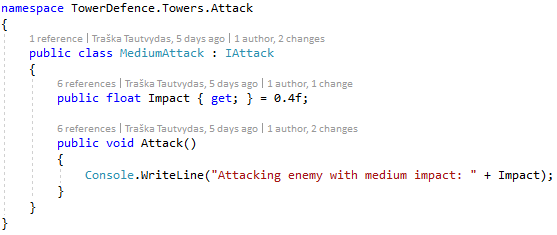
Komandos (angl. Command) projektavimo šablonas taikomas tada, kai norima, kad tam tikras veiksmas būtų ne klasės metodas, bet atskiras objektas, t.y., kai norima atskirti klientą (naudotoją), kuris vykdo komandą, nuo pačios komandos vykdymo detalių, priklausomybių nuo kitų objektų ir jų logikos. Mūsų pavyzdyje bokštų šaudymo komanda yra kaip atskiras objektas (klasė), o kai iš kliento pusės norima vykdyti bokšto šaudymo veiksmą, užtenka sukurti tos komandos objektą ir įvykdyti komandą execute() – nereikia komandai paduoti jokių parametrų, nereikia suprasti komandos logikos, užtenka ją iškviesti.

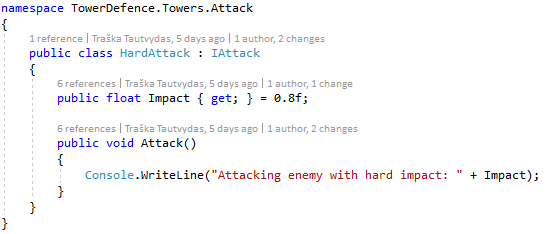
## Tilto (Bridge) projektavimo pavyzdys (Tautvydas)

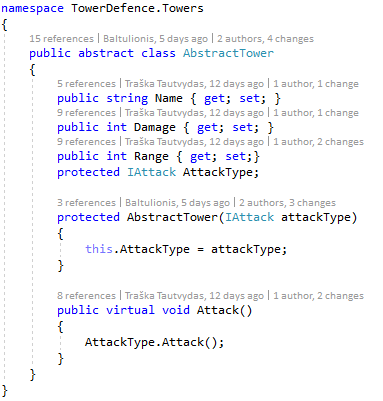
### Projektavimo šablono realizacija

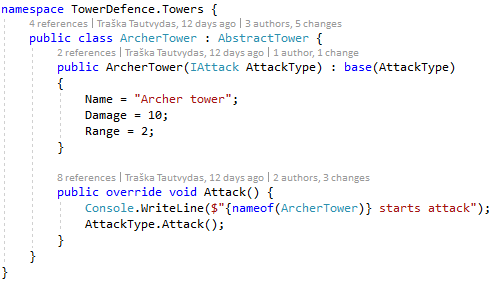


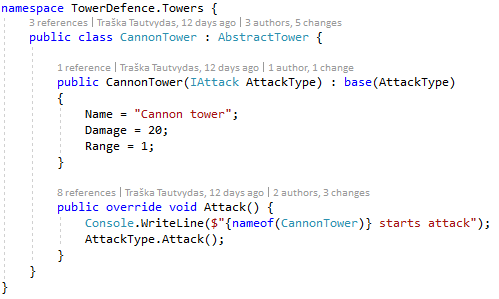












11 pav. Tilto (Bridge) realiazacija

### Projektavimo šablono naudojimo pagrindimas

Bridge šablonas yra skirtas, kai norima atskirti abstrakciją nuo implementacijos, taigi šios gali kisti nepriklausomai viena nuo kitos. „Tower Defence“ žaidimo atveju, šis šablonas panaudotas apsauginiams bokštams, kurie gali pulti priešus skirtinga jėga. Vienas konkretus bokštas, kuris paveldi abstraktaus bokšto savybes, gali turėti skirtingus atakos tipus, nepriklausomai nuo to, koks bokštas yra kuriamas. Taigi, atsiranda daugiau ir įvairesnių galimybių sukurti bokštą, kuris tampa labiau dinamiškas, nes nėra tiesiogiai susietas su konkrečiu atakos tipu.