**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**OBJEKTINIS PROGRAMŲ PROJEKTAVIMAS (T120B516)**

**Laboratorinis darbas Nr.2**

Atliko:

IFF – 4/3 gr. studentai:

Laima Skurdelytė

Eivydas Senkus

IFF – 4/2 gr. studentas

Rolandas Jasiūnas

2017 m. gruodžio 15 d.

Priėmė:

Lekt. Andrej Ušaniov

**KAUNAS 2017**

Turinys

[1. Projekto aprašymas 3](#_Toc501050422)

[2. Darbo eiga 3](#_Toc501050423)

[2.1. Template pattern panaudojimas 3](#_Toc501050424)

[2.2. State pattern panaudojimas (Rolandas Jasiūnas) 4](#_Toc501050425)

[2.3. Flyweight pattern panaudojimas (Laima Skurdelytė) 5](#_Toc501050426)

[2.4. Proxy pattern panaudojimas (Eivydas Senkus) 6](#_Toc501050427)

[2.5. Chain of Responsibility pattern panaudojimas 8](#_Toc501050428)

[2.6. Null Object pattern panaudojimas 9](#_Toc501050429)

[2.7. Memento pattern panaudojimas (Rolandas Jasiūnas) 11](#_Toc501050430)

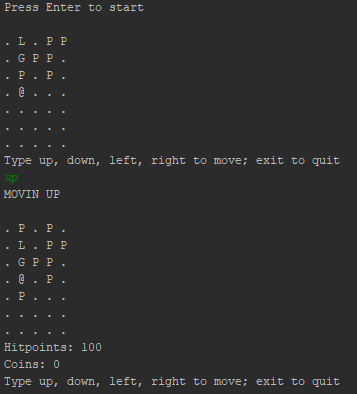
[2.8. Visitor pattern panaudojimas (Laima Skurdelytė) 13](#_Toc501050431)

[2.9. Mediator pattern panaudojimas (Eivydas Senkus) 14](#_Toc501050432)

[3. Išvados 15](#_Toc501050433)

# **Projekto aprašymas**

Kuriamas projektas – žaidimas labirintas. Pagrindinis tikslas rasti išėjimą iš labirinto, kai nėra matomas visas labirintas iš karto. Jis atvaizduojamas fragmentais po kiekvieno žaidėjo ėjimo. Žaidėjas pasirenka ėjimą įrašydamas komandą. Įvykdžius komandą matomas rezultatas: ar siena, ar laisvas praėjimas, ar spąstai, ar gyvybės papildomos ir panašiai. Žaidimas baigiamas žaidėjus radus išėjimą arba netekus visų gyvybių.

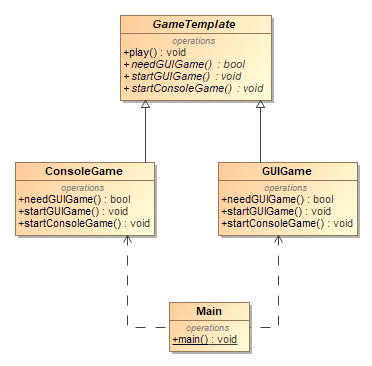


1 pav. Žaidimo konsolės langas

# **Darbo eiga**

## Template pattern panaudojimas

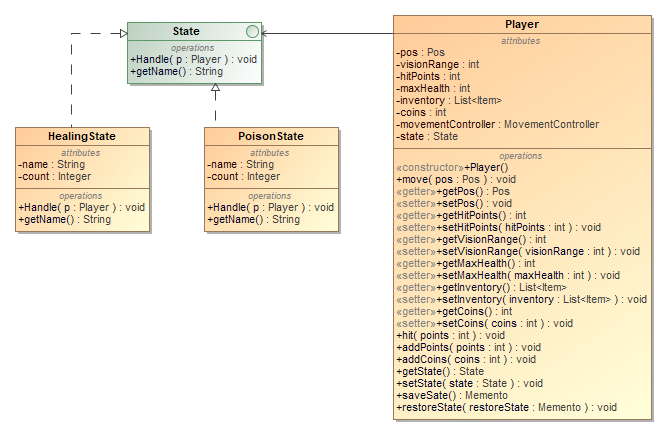
Template šablonas skirtas aprašyti algoritmo skeletas ir per paveldėjimą leidžia aprašyti algoritmo logiką nekenčia pačios algoritmo struktūros. Pritaikant šitą šabloną buvo sukurtos žaidimo abstrakti paleidimo klasė - GameTemplate ir jos viduje algoritmas. Sukurtos žaidimo paleidimo klasės(ConsoleGame, GUIGame) kur viena paleidžia tik konsolinę aplikaciją, kitu atveju ir vartotojo sąsajos aplikaciją pagal paduotus paleidimo argumentus.



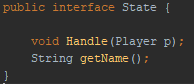
pav. Template šablono panaudojimo klasių diagrama

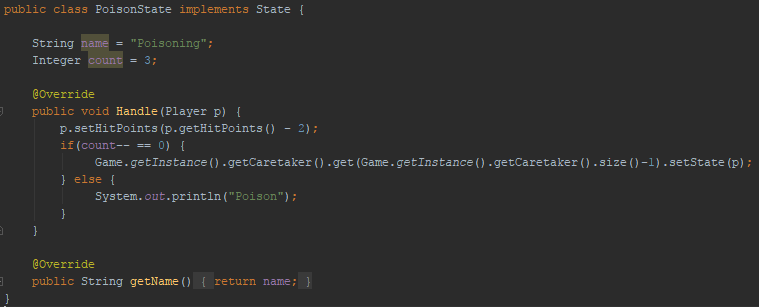
## State pattern panaudojimas (Rolandas Jasiūnas)

State šablonas skirtas vykdymo metu nurodyti kaip objektas turi, pakeitus būseną objekto veikla turi irgi pasikeisti atitinkamai nuo tos būsenos. Sablonas buvo pritaikytas keisti žaidėjo būseną priklausomai nuo kas įvyko žaidimo metu. Šablono taikymo metu buvo sukurta būsenos klasių (BaseState, HealingState, PoisoningState) ir pridėtas būsenos objekto saugojimas žaidėjo objekte. Programos paleidimo metu žaidėjui priskiriama pagrindinė būsena - BaseState kuri nieko nedaro, žaidimo metu žaidėjas užlipęs ant spąstų gali gauti PoisonState (mažina gyvybes kiekvieną ėjimą) ir pasiėmęs daiktą – HealingState (didina gyvybes kiekvieną ėjimą).



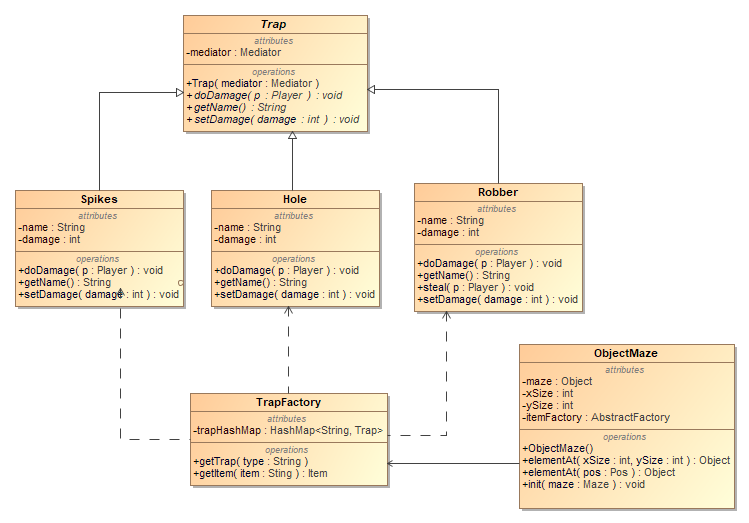
pav. State šablono klasių diagrama





## Flyweight pattern panaudojimas (Laima Skurdelytė)

Flyweight šablonas pirmiausia yra naudojamas siekiant sumažinti sukurtų objektų skaičių ir padidinti našumą. Šablonas bando pakartotinai panaudoti jau egzistuojančius išsaugotus objektus ir sukurti naują objektą tada, kai nėra išsaugotų tinkamų objektų. Todėl šį šabloną bandyta pritaikyti pakeičiant kodą taip, kad kiekvieną kartą prireikus kurti Trap objektą, būtų pirmiausia patikrinama ar nebuvo sukurtas tokio tipo Trap objektas. Pirmą kartą kuriami Trap objektai skirtingų tipų yra išsaugomi Hashmap išsaugant Trap tipą ir Trap objektą.

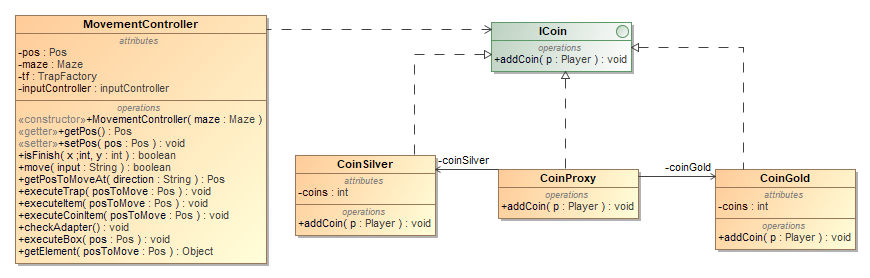


pav. Flyweight šablono panaudojimo klasių diagrama

|  |
| --- |
| Šiam šablonui pritaikyti buvo pakoreguota TrapFactory klasė, kur vietoj to, kad kiekvieną kartą sukurtų naują Trap tam tikro tipo objektą, tikrina ar nėra tokio objekto jau išsaugoto Hashmap, jei yra, tada jį grąžina, jei nėra sukuria naują ir įdeda į Hashmap. |
|  |
| Žemiau matomas veikimas, kad iš pradžių Hashmape nebuvo Spikes tipo objekto, o antrą kartą žaidėjui ant jo užlipus, Spikes objektas buvo nebe kuriamas, bet paimtas iš Hashmap. |
|  |

## Proxy pattern panaudojimas (Eivydas Senkus)

Proxy pattern yra naudojamas kaip apgaubiamasis objektas, kuris yra kviečiamas tada, kai klientas nori pasiekti tikrąjį objektą. Proxy gali tiesiog persiųsti užklausą arba pridėti papildomą funkcionalumą, kuris, pavyzdžiui, apribotų prieigą prie objekto arba įrašytų objektą į atmintį.

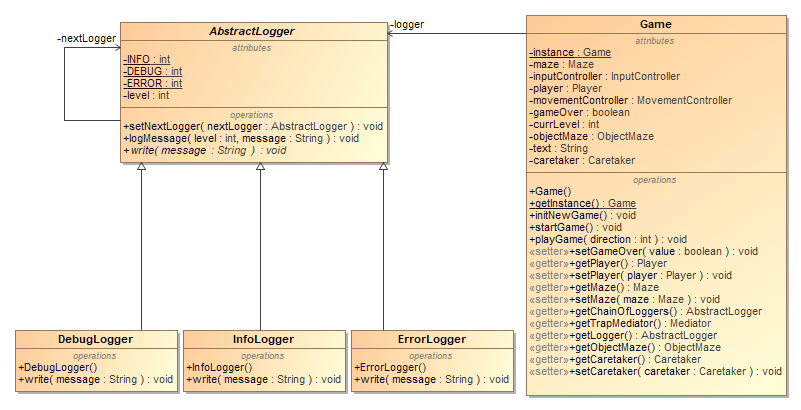


pav. Proxy šablono panaudojimo klasių diagrama

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

## Chain of Responsibility pattern panaudojimas

Šis šablonas sukuria užklausos gavėjų grandinę. Jis pagal užklausos tipą atskiria jos siuntėją ir gavėją. Šiame šablone įprastai gavėjas turi nuorodą į kitą gavėją. Jei vienas iš objektų negali apdoroti užklausos, tai ją perduoda kitam gavėjui ir t.t. Todėl buvo sukurta AbstractLogger ir ją realizuojančios klasės InfoLogger, DebugLogger ir ErrorLogge, kad įgyvendinti šį šabloną. Iš šių klasių sukuriama grandinė, kuri apdoroja norimą išspausdinti informaciją, ir nusprendžia kur ją išspausdinti: information, error ar debug lange.

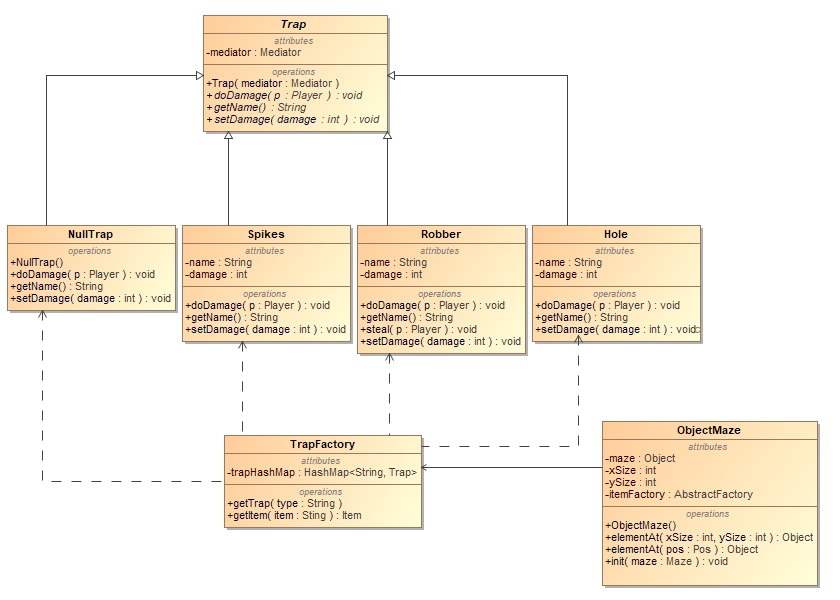


pav. Chain of Responsibility šablono panaudojimo klasių diagrama

|  |
| --- |
| Sukurta klasė AbstractLogger, turinti savyje nuorodą į sekančią grandį ir abstraktų metodą, kuris išspausdintų pranešimą. |
|  |
| Viena iš klasių, realizuojančių AbstractLogger, ir atspausdinanti informacinį pranešimą. |
|  |

## Null Object pattern panaudojimas

Sukurtai abstrakčiai klasei Trap, kurią realizuoja konkrečios klasės Spikes, Robber, Hole pridedame ir realizuojančią klasę NullTrap, kuri vietoj null grąžinimo tiesiog neatliktų kviečiamų operacijų.

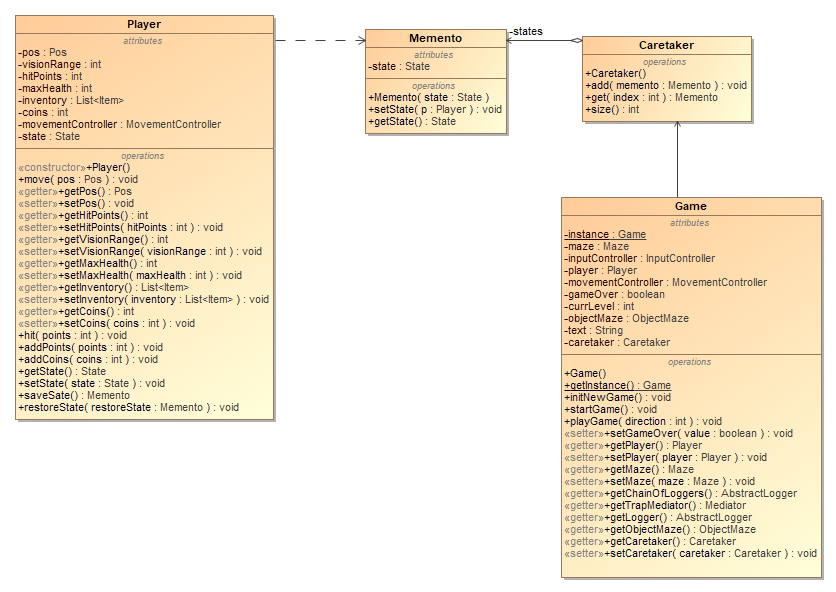


pav. Null Object šablono panaudojimo klasių diagrama

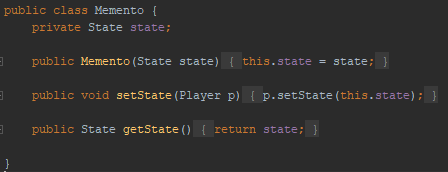
|  |
| --- |
| Sukurta klasė NullTrap, kurią objektą gražina, jei prašomas objekto tipas neatitinka jokio iš kitų tipų realių objektų. |
|  |
| Šią abstrakčią klasę apibrėžiančią operacijas, realizuoja NullTrap klasė |
|  |
| TrapFactory grąžina tam tikro tipo realų Trap objektą arba NullTrap objektą, jei prašomas tipas neatitinka realių tipų. |
|  |

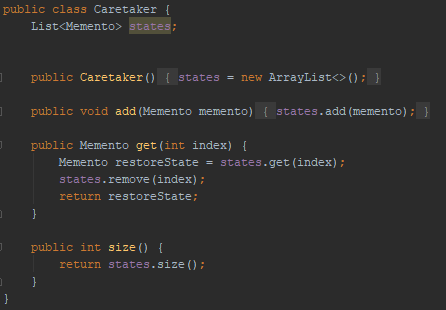
## Memento pattern panaudojimas (Rolandas Jasiūnas)

Momento šablonas pritaikytas gražinti buvusias būsenas objektams. Kiekvieno būsenos keitimo metu išsaugoma prieš tai buvusi būsena tam kad butų galimybė sugražinti tą būseną priklausant nuo situacijos. Caretaker turi būsenų sąrašą, bet pats nebendrauja su objektais kurių būsena bus gražinama, busena sugražinama naudojant Memento. Šablono kūrimo metu buvo sukurta klasė Caretakar turinti būsenų sąrašą, būsenos jame saugomos Memento objektais kad butų galima gražinti Žaidėjui būseną. Memento objektas turi būseną ir gali tą turimą būseną priskirti paduotam žaidėjui. Žaidėjas prieš keisdamas būseną išsaugo ją į Memento objektą ir nusiunčia į Caretaker. Realizuota taip kad kiekviena būsena po 3 ėjimų grįžta į ankstesnę būseną.



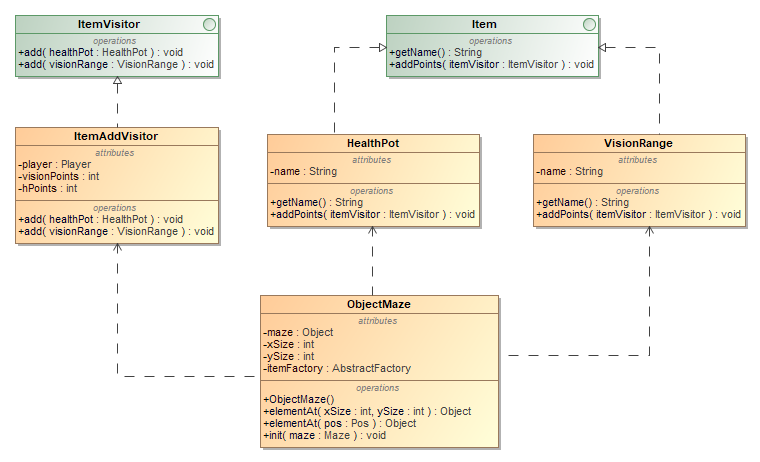
8 pav. Memento šablono panaudojimo klasių diagrama





## Visitor pattern panaudojimas (Laima Skurdelytė)

Visitor šablonas leidžia apibrėžti naują operaciją nekeičiant elementų klasių, kuriose jis veikia. Todėl buvo sukurtas ItemVisitor, kurį naudoja Item realizuojančios klasės HealthPot ir VisionRange. Taigi, ItemVisitor gali pridėti naujų operacijų nepakeičiant HealthPot ir VisiongRange klasių.

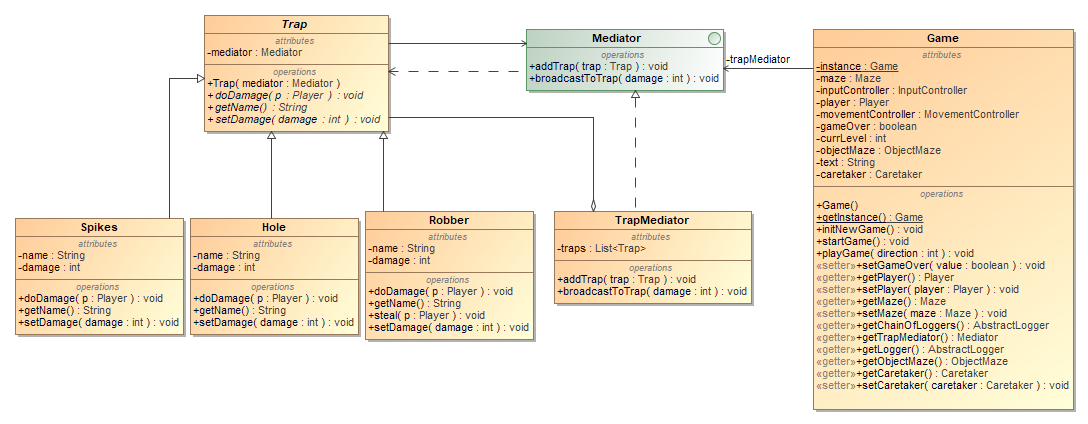


pav. Visitor šablono klasių diagrama

|  |
| --- |
| Sukurtas naujas interface ItemVisitor, kurį realizuoja ItemAddVisitor. |
|  |
| ItemAddVisitor realizuoja metodus, kuriais redaguojami žaidėjo taškai ir matymo diapazonas pagal tai kuris Item iškvietė metodą. |
|  |
| Žemiau matomas Item interface, kurį realziuoja HealthPot klasė. Ši klasė naudoja ItemVisitor, kad jis atliktų reikalingas operacijas. |
|  |

## Mediator pattern panaudojimas (Eivydas Senkus)

Mediator pattern yra skirtas aprašyti objektui, kuris apima kitų objektų aibę, ir nusako kaip tie objektai bendraus tarpusavyje. Mediator pattern esmė neleisti objektams bendrauti tarpusavyje, bet, vietoj to, bendrauti objektams su pattern objektu, o jis informaciją perduoda visiems kitiems objektams.



pav. Mediator šablono panaudojimo klasių diagrama

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

# **Išvados**

Darbo metu panaudoti Template, State, Flyweight, Proxy, Chain of Responsibility, Null Object, Memento, Visitor ir Mediator projektavimo šablonai.

Visi šalbonai buvo pritaikyti mūsų kuriamam žaidimui ir panaudoti su tikslu. Naudingiausi šablonai buvo State + Memento, Proxy, Chain of Responsibility ir Flyweight, kadangi jie labai tinka mūsų kuriamame žaidime. Kiti patterns buvo panaudoti tiesiog, nes juos panaudoti reikia – jie mums nei nėra naudingi, nei lengvai pritaikomi. Sunkiausia buvo sugalvoti Template pattern ir Mediator. Visi patterns veikia sklandžiai, jiems buvo sukurtos UML diagramos, užbaigta ataskaita.