**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**T120B516** Objektinis programų projektavimas

**Laboratorinis darbas 1**

Darbą atliko:

Ignas Lapėnas

Andrius Jankauskas

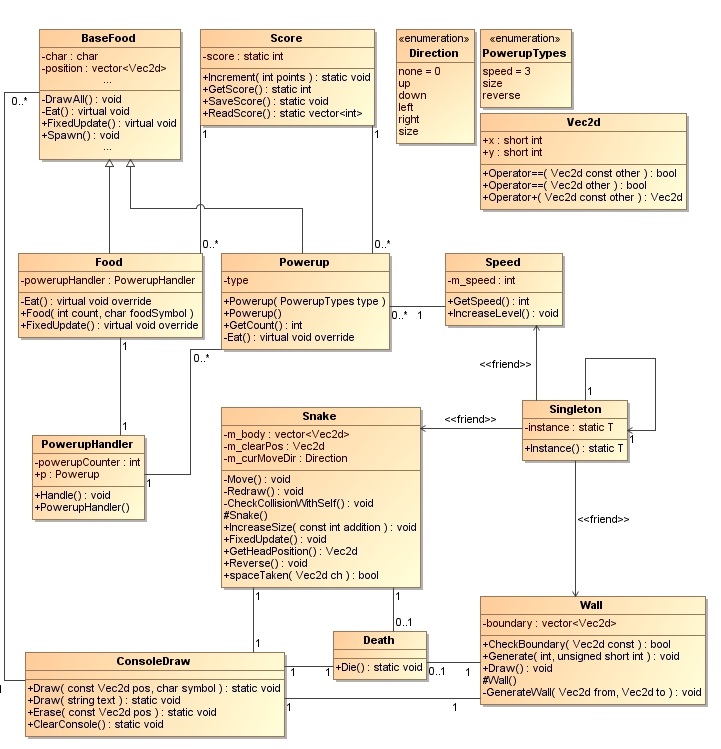
KTU 2016

# Užduotis

Pasirinkti užduotį ir ją aprašyti. Parengti svetainę el. ataskaitos talpinimui. Realizuoti Singleton šabloną: kodas, diagramos, pagrindimas. Turi būti visos sistemos pirminė klasių diagrama. Singleton veikimas turi būti pademonstruotas. Karkasų/frameforkų nenaudoti. Nemažiau 10 klasių! Šablonų veikimas turi būti pristatytas per veikiantį main() metodą.

# Realizacija

Suprojektuota klasių diagrama:



pav. 1 Suprojektuota klasių diagrama

# Singleton kodo realizacija

#pragma once

template<typename T>

class Singleton

{

private:

Singleton(Singleton const&) = delete;

Singleton(Singleton&&) = delete;

Singleton& operator=(Singleton const&) = delete;

Singleton& operator=(Singleton&&) = delete;

protected:

Singleton();

~Singleton();

public:

static T& Instance()

{

static T instance;

return instance;

}

};

## Snake realizacija su nustatytu duomenų tipu

#include <memory>

#include "Vec2d.h"

#include "Direction.h"

#include "Singleton.h"

class Snake

{

friend class Singleton<Snake>;

private:

std::vector<Vec2d> m\_body;

Vec2d m\_clearPos;

Direction m\_curMoveDir;

void Move();

void Redraw();

void CheckCollisionWithSelf();

protected:

Snake();

~Snake();

public:

void IncreaseSize(const int addition);

void FixedUpdate();

Vec2d GetHeadPosition() { return m\_body[0]; }

void Reverse();

bool spaceTaken(Vec2d ch) { return (std::find(m\_body.begin(), m\_body.end(), ch)) == m\_body.end() ? false : true; }

};

## Panaudojimas

void Wall::GenerateWall(Vec2d from, Vec2d to) {

if (from.x != to.x && from.y != to.y) {

throw std::exception("Bad coordinates. Can't make wall!");

return;

}

if (from.x >= to.x && from.y >= to.y)

swap(from, to);

while (from.x != to.x || from.y != to.y) {

if (from.x < to.x) {

from.x++;

}

if (from.y < to.y) {

from.y++;

}

if (!CheckBoundary(from) && !(Singleton<Snake>::Instance().GetHeadPosition() == from))

boundary.push\_back(from);

}

}

# Singleton paskirtis

Programų inžinerijoje, singleton šablonas yra projektavimo modelis kuris draudžia sukurti ir turėti daugiau nei vieną klasės objektą vienoje laiko linijoje. Tai yra naudinga, kai reikia būtent tik vieno objekto koordinuoti veiksmus visoje sistemoje. Kai kurios sistemos veikia efektyviau kai egzistuoja tik vienas konkrečios klasės objektas, arba kai yra draudžiama sukurti daugiau negu numatytas kiekis objektų. Sąvoka kilo iš matematinio singleton modelio.

## Panaudojome

Panaudota Snake, Wall ir Speed klasėse, nes pagal mūsų žaidimo mechaniką neturėtšų egzistuoti daugiau nei vienas žaidėjas. Jis žais tik viename žemėlapyje, o atnaujinimo laikas turi būti bendras ir globalus.

# Išvados

Patogus šablonas jeigu mes norime apriboti klasę, kad ji egzistuotų tik vieną kartą nuo pirmo panaudojimo iki programos gyvenimo pabaigos arba priverstinio atminties atlaisvinimo operacijos. Pagrindinis minusas kurį pastebėjome, kad šis šablonas paverčia klasę globalia, o to pasekoje jei komandos kviečiamos iš daugelio programos vietų atrasti klaidą tampa itin sudetinga.