|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  **НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ “КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”**  ННК «ІПСА»  (назва факультету, інституту)  Кафедра системного проектування  (назва кафедри)  **ПОЯСНЮВАЛЬНА ЗАПИСКА**  до курсової роботи  з дисципліни ―Об’єктно-орієнтоване програмування‖  на тему: ―Монополія‖  Виконав: Поплавський Владислав Олегович ст. гр. ДА-**92**  Поплавський В. О.  Київ – 2020 | | | | | | |
|  |  |  |  |  | *ДА92.15-0015.001* | *Арк* |
|  |  |  |  |  | 1 |
| *Зм.* | *Арк.* | *№ докум.* | *Підпис* | *Дата* |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Зміст  [ЗАВДАННЯ 3](#_Toc59703810)  [Вступ 4](#_Toc59703811)  [Структура програми 5](#_Toc59703812)  [Діаграма станів 5](#_Toc59703813)  [Use case діаграма 7](#_Toc59703814)  [Шаблон проектування ECS 8](#_Toc59703815)  [Реалізація функціоналу 10](#_Toc59703816)  [Основне вікно 10](#_Toc59703817)  [Дизайн карток 11](#_Toc59703818)  [Висновки 12](#_Toc59703819)  [Додаток К. Код розглянутого прикладу 13](#_Toc59703820) | | | | | | | | | | |
|  |  |  |  |  | *ДА92.15-0015.001* | | | | | |
|  |  |  |  |  |
| *Зм.* | *Арк.* | *№ докум.* | *Підпис* | *Дат* |
| *Розроб.* | | *Поплавський В.О* |  |  | **Зміст** | *Літ.* | | | *Арк.* | *Аркушів* |
| *Перевір.* | | *Булах Б.В.* |  |  |  |  |  | *2* | *13* |
| *Т. Контр.* | |  |  |  | НТУУ «КПІ» ННК «ІПСА» гр.ДА-92 | | | | |
| *Н. Контр.* | |  |  |  |
| *Затверд.* | |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ЗАВДАННЯ **на курсову роботу з дисципліни**  ―Об’єктно-орієнтоване програмування‖  **Тема роботи:** Розробка монополії з графічним інтерфейсом.  **Детальний опис:**  Змоделювати хід настільної гри типу “Монополії” з випадковою генерацією ігрового поля (бажано, відмінного від типового “квадрату”, див. варіанти на рис.) з різними об’єктами нерухомості, генерацією колод подій, штрафів та бонусів, та ін. Тематика може бути довільною (промисловість країни, космос, об’єкти в місті, туризм, спорт, ІТ). Підтримувати гру з кількома гравцями, включаючи “штучний інтелект” (проста реалізація).  Варіанти розподілу відповідальності в команді: а) проектування та підготовка вхідних даних (генерування або завантаження ігрового поля, колод подій); б) реалізація поведінки гравців, керованих “штучним інтелектом”, логіка торгівлі між гравцями; в) здійснення ходів, оновлення параметрів гравців після кожного ходу відповідно до правил та виконаних дій, ведення статистики, визначення переможця. | | | | | | |
|  |  |  |  |  | *ДА92.15-0015.001* | *Арк* |
|  |  |  |  |  | 3 |
| *Зм.* | *Арк.* | *№ докум.* | *Підпис* | *Дата* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вступ Поставлена задача розробки монополії з графічним інтерфейсом користувача. Існує багато різноманітних рішень та реалізацій монополії будь то комп’ютерна гра чи настільна. Наша програма буде базуватися на класичній настільній монополії. Також у якості тематики буде використана тематика КПІ, з відповідним дизайном карток, описом подій, будуть використовуватися знайомі локації.  Важливою частиною у розробці програми є її здібність до легкого масштабування. Саме цю проблему вирішує використання ООП підходу створення структури програми. Це позволить при довгостроковому підході легко додавати нові функції, модифікувати старий код, налагоджувати програму та тестувати її. Також такий підхід суттєво спрощує розробку у команді, бо можна обговорити загальну структуру та розподілити обов’язки таким чином, щоб кожен мав свою зону відповідальності, незалежну від інших, що також може вплинути на швидкість розробки програми.  Не менш важливої частиною розробки є оптимізація роботи коду програми. Перед нами стоять цілі:   * Оптимізованого коду; * Маштабованого коду; * Зручного середовища розробки інтерфейсу користувача;   Базуючись на усіх цих параметрах була вибрана мова С#. Та для раціонального управління пам’яттю та для результуючої швидкості роботи програми, було вибрано ООП-патерн ECS, про який піде мова детальніше у розділі “Структури програми”. Треба зазначити, що саме цей патерн реалізовує одий з найвідоміших та найпопулярніших сучасних игрових движків – Unity, що також вплинуло на подальше рішення. | | | | | | | | | | | | | | | | | | |
|  | |  | |  | |  | |  | | *ДА92.15-0015.001* | | | | | | | *Арк* | |
|  | |  | |  | |  | |  | | 4 | |
| *Зм.* | | *Арк.* | | *№ докум.* | | *Підпис* | | *Дата* | |
| Структура програми Почнемо розробку зі стровення самої абстрактної структури програми, використовуючи UML діаграму класів. Треба зазначити, що початкове уявлення структури ще не є ідеальним і з часом розробки може зазнати суттєвих змін.  Тому спочатку додаю зображення частин діаграми станів, для розуміння, яку логіку треба реалізовувати, та які мають міститися класи. Діаграма станів | | | | | | | | | | | | | | | | | | |
|  | |  | |  | |  | |  | | *ДА92.15-0015.001* | | | | | | | *Арк* | |
|  | |  | |  | |  | |  | | 5 | |
| *Зм.* | | *Арк.* | | *№ докум.* | | *Підпис* | | *Дата* | |
| Частина логіки ходу гравця.    Частина з реалізацією торгівлі | | | | | | | | | | | | | | | | | | |
|  | |  | |  | |  | |  | | *ДА92.15-0015.001* | | | | | | | *Арк* | |
|  | |  | |  | |  | |  | | 6 | |
| *Зм.* | | *Арк.* | | *№ докум.* | | *Підпис* | | *Дата* | |
| Use case діаграма Маємо також Use case діаграму для уявлення, які самі менеджери треба реалізовувати.    Також наявна діаграма класів, але у ході розробки програмо було вирішенно суттєво змінити структуру проекту, тому не бачу сенсу її демонструвати. Саме для реалізації патерну програмування, яким було завдання у третій лабораторній роботі, було вибрано ECS паттерн, якій відповідає меті у вступі. | | | | | | | | | | | | | | | | | | |
|  | |  | |  | |  | |  | | *ДА92.15-0015.001* | | | | | | | *Арк* | |
|  | |  | |  | |  | |  | | 7 | |
| *Зм.* | | *Арк.* | | *№ докум.* | | *Підпис* | | *Дата* | |
| Шаблон проектування ECS ECS - це шаблон проектування "Сутність Компонент Система" (Entity Component System). Якщо зовсім по-простому, то є "Сутності" (Entity) - об'єкти-контейнери, що не володіють властивостями, але виступаючі сховищами для "Компонентів". "Компоненти" - це блоки даних, що визначають всілякі властивості будь-яких ігрових об'єктів або подій. Всі ці дані, згруповані в контейнери, обробляються логікою, яка існує виключно у вигляді "Систем" - "чистих" класів з певними методами для виконання. Даний патерн є незалежним від будь-якого "движка" і може бути реалізований багатьма способами. Всі "сутності", "системи" і "компоненти" повинні десь зберігатися і якимось чином инициализироваться - все це є особливостями реалізації кожного ECS рішення для конкретного "движка". Тобто цей патерн має ідеальну структуру для подібних ігр зі складною логікою, тому був вибраний саме він.  Тепер маючи уявлення про використаний шаблон проектування, звернемся до готового проекту та розглянемо його структуру:  Як бачимо, не зважаючи на системні файли маємо саме описану структури програми, де є чіткий розподіл на логіку та дані.  Розглянемо приклад систем:  Бачимо зручне розташування усіх записаних логік в одному місці, які відповідають за передачу ходу, закінчення гри, пересування гравця, ввод гравця, реалізацію торгівлі та аукціону та багато інших. | | | | | | | | | | | | | | | | | | |
|  | |  | |  | |  | |  | | *ДА92.15-0015.001* | | | | | | | *Арк* | |
|  | |  | |  | |  | |  | | 8 | |
| *Зм.* | | *Арк.* | | *№ докум.* | | *Підпис* | | *Дата* | |
| Розглянемо так звані “компоненти”. Саме тут розташовані чисті данні, які використовуються для позначення клітинок, гравців,самої карти, їх станів, їх запитів до систем.  Далі продемонструю останню з важливих частин проекту – це поведінки компонентів.    Тобто можна бачити, шо системи лише маніпулюють порядком та випадком використання тієї чи іншої логіки, а вона розбита на прості логічні одиниці у вигляді поведінок (behavior) компонентів.  Також проект містить додаткові класи для визначення залежностей між усіми системами, класи вводу/вивіду даних, клас Context , через який взаємодіють усі системи з “entities”. Для зручного керування такими залежностями використовується пакет Autofac. Autofac - це контейнер IoC для Microsoft .NET 4.5, Silverlight 5, додатків для Windows Store і додатків для Windows Phone 8. Він керує залежностями між класами, щоб додатки залишалися легко змінюються по мірі зростання і складності. Це досягається шляхом обробки звичайних класів .NET як компонентів.  Також проект містить інши класи які відтворють штучний інтелект та ввод гравця, які знадобляться при створенні графічного інтерфейсу, тому як бачимо, програма має чітко визначену просту логічну структуру) | | | | | | | | | | | | | | | | | | |
|  | |  | |  | |  | |  | | *ДА92.15-0015.001* | | | | | | | *Арк* | |
|  | |  | |  | |  | |  | | 9 | |
| *Зм.* | | *Арк.* | | *№ докум.* | | *Підпис* | | *Дата* | |
| Реалізація функціоналу Для відтворення графічного інтерфейсу було використано WPF. Windows Presentation Foundation (WPF) - це платформа для користувача інтерфейсу для створення клієнтських додатків для настільних систем.  Основне вікно:  Опис елементів:  Карта з гравцями Меню вводу команд, список гравців, збільшене зображення клітинки. | | | | | | | | | | | | | | | | | | |
|  | |  | |  | |  | |  | | *ДА92.15-0015.001* | | | | | | | *Арк* | |
|  | |  | |  | |  | |  | | 10 | |
| *Зм.* | | *Арк.* | | *№ докум.* | | *Підпис* | | *Дата* | |
| Дизайн карток | | | | | | | | | | | | | | | | | | |
|  | |  | |  | |  | |  | | *ДА92.15-0015.001* | | | | | | | *Арк* | |
|  | |  | |  | |  | |  | | 11 | |
| *Зм.* | | *Арк.* | | *№ докум.* | | *Підпис* | | *Дата* | |
| Висновки У ході виконання курсової роботи було здобуто навички роботи у команді, працювання з системою контроля версій git, закріплено знання основних принципів ООП, SOLID, їх реалізацїї у мові С#. Ознайомлено з шаблоном проектування ECS. Реалізовано програму Монополія зі штучним інтелектом. Передбачено змагння з компьютером чи з реальними гравцями. Реалізован інтуітивно зрозумілий графічний інтерфейс з картою, у якій всі клітинки власноруч виконані у КПІ тематиці. Теоретичні знання з ООП були втілені у практику. Література1. А.Н. Васильев Java. Объектно-ориентированное программирование / А.Н. Васильев. - М.: Питер, ****2014****. - 398 c. 2. А.Н. Васильев Java. Объектно-ориентированное программирование / А.Н. Васильев. - М.: Питер, ****2014****. - 400 c. 3. Алексей Васильев C#. Объектно-ориентированное программирование / Алексей Васильев. - М.: Питер, ****2017****. - 320 c. 4. Г.С. Иванова Объектно-ориентированное программирование / Г.С. Иванова, Т.Н. Ничушкина, Е.К. Пугачев. - М.: МГТУ им. Н. Э. Баумана, ****2014****. - 368 c. 5. Г.С. Иванова Объектно-ориентированное программирование / Г.С. Иванова. - М.: Московский Государственный Технический Университет (МГТУ) имени Н.Э. Баумана, 2014. - ****149**** c. 6. Дж. Кьоу Объектно-ориентированное программирование / Дж. Кьоу, М. Джеанини. - М.: Питер, ****2015****. - 240 c. 7. Е.В. Санников Курс практического программирования в Delphi. Объектно-ориентированное программирование / Е.В. Санников. - М.: Солон-Пресс, ****2017****. - 188 c. 8. Комлев Николай Юрьевич Объектно Ориентированное Программирование. Хорошая книга для Хороших Людей / Комлев Николай Юрьевич. - М.: Солон-Пресс, 2014. - ****892**** c. 9. Н.Н. Мартынов Алгоритмизация и основы объектно-ориентированного программирования на JavaScript. Информатика и ИКТ. Профильный уровень. 10 класс / Н.Н. Мартынов. - Москва: ****Высшая школа****, ****2014****. - 272 c. 10. П.Б. Хорев Объектно-ориентированное программирование / П.Б. Хорев. - М.: Academia, ****2017****. - 448 c. 11. Татьяна Павловская C/C++. Процедурное и объектно-ориентированное программирование. Учебник / Татьяна Павловская. - М.: Питер, 2015. - 496 c. | | | | | | | | | | | | | | | | |
|  | |  | |  | |  | |  | | *ДА92.15-0015.001* | | | | | | |
|  | |  | |  | |  | |  | |
| *Зм.* | | *Арк.* | | *№ докум.* | | *Підпис* | | *Дат* | |
| *Розроб.* | | | | *Поплавський.* | |  | |  | | **Висновки** | *Літ.* | | | *Арк.* | *Аркушів* | |
| *Перевір.* | | | | *Булах Б.В.* | |  | |  | |  |  |  | *12* | *13* | |
| *Т. Контр.* | | | |  | |  | |  | | НТУУ «КПІ» ННК «ІПСА» гр.ДА-92 | | | | | |
| *Н. Контр.* | | | |  | |  | |  | |
| *Затверд.* | | | |  | |  | |  | |

# Додаток К. Код розглянутого прикладу

using MonopolyPreUnity.Components.SystemRequest.Output;

using MonopolyPreUnity.Components.SystemRequest.PlayerInput;

using MonopolyPreUnity.Entity;

using MonopolyPreUnity.Entity.ContextExtensions;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Text;

namespace MonopolyPreUnity.Systems

{

class ChangeTurnSystem : ISystem

{

private readonly Context \_context;

public void Execute()

{

var request = \_context.GetComponent<EndTurn>();

if (request == null)

return;

var turnInfo = \_context.TurnInfo();

\_context.PlayerObservables[turnInfo.CurTurnPlayer].IsCurTurn = false;

turnInfo.CurTurnPlayer = (turnInfo.CurTurnPlayer + 1) % turnInfo.TurnOrder.Count;

\_context.PlayerObservables[turnInfo.CurTurnPlayer].IsCurTurn = true;

var curTurnPlayer = \_context.GetPlayer(turnInfo.CurTurnPlayerId);

curTurnPlayer.CanMove = true;

curTurnPlayer.RolledJailDiceThisTurn = false;

//\_context.Add(new ClearOutput());

\_context.Add(new PrintFormattedLine

($"Next turn. It's time for |player:{turnInfo.CurTurnPlayerId}| to make a move!", OutputStream.GameLog));

\_context.Remove<EndTurn>();

}

public ChangeTurnSystem(Context context)

{

\_context = context;

}

}

}