**КН-19-2**
**Голяк Д.В.**

# ЛАБОРАТОРНА РОБОТА 6
## Рефакторинг програмного коду

**Мета роботи:** навчитися виконувати реорганізацію програмного коду на підставі шаблонів рефакторінгу.

**Завдання:**

1. Вивчити теоретичні відомості.
2. Виконати аналіз програмного коду розроблюваного ПО і модульних тестів з метою виявлення погано організованого код.
3. Використовуючи шаблони рефакторінгу, виконати реорганізацію програмного коду розроблюваного ПО і модульних тестів.
4. Перевірити успішність виконання всіх модульних тестів.
5. Виконати опис вироблених операцій рефакторінгу (було-стало-шаблон рефакторінгу).
6. В разі необхідності коректувати проектну документацію (діаграми класів, послідовностей).
7. Зробити висновки результатами виконання роботи.

## Хід роботи:

**Рефакторинг:**

1. Рефакторинг у класах групи Move, що відповідають за рух фігур на полі відповідно до дій гравця.

   Зміни:

   - Extract Interface - Виділений абстрактний клас, що включає загальні особливості.
   - Encapsulate Field – Додані модифікатори доступу protected та readonly, для обмеження можливого зайвого використання полів.
   - Додані коментарі та розділення на блоки коду

   Було:

```
public class MoveLeft
	{
		protected PlayField _playField;

		public MoveLeft(PlayField playField) {
			_playField = playField;
		}

		public void Execute()
		{
			_playField.MoveLeft();
```

```
        }
    }

    public class MoveRight
    {
        protected PlayField _playField;

        public MoveRight(PlayField playField) {
            _playField = playField;
        }

        public void Execute()
        {
            _playField.MoveRight();
        }
    }

    public class MovePause
    {
        protected Game _game;

        public MovePause(Game game) {
            _game = game;
        }

        public void Execute()
        {
            _game.Paused = !_game.Paused;

            MenuPauseForm menu = new MenuPauseForm();
            menu.ShowDialog();

            _game.Paused = !_game.Paused;
        }
    }
```

Стало:

```
/// <summary>Клас Команди оголошує метод для виконання команд.</summary>
    public abstract class Command
    {
        protected readonly PlayField _playField;
        protected readonly Game _game;

        protected Command(PlayField playField)
        {
            _playField = playField;
        }

        protected Command(Game game)
        {
            _game = game;
        }

        public abstract void Execute();
    }

    /// <summary>Клас ініціалізатор команд, відправляє запит на команду.</summary>
    public class Invoker
    {
        /// <summary>Команда для виконання</summary>
        private Command _command;

        public Command Command
        {
            set => _command = value;
        }
        /// <summary>Запуск команди</summary>
        public void Run()
```

```
            {
                _command?.Execute(); // якщо є команда, то виконуємо
                _command = null; // після виконання очищуємо
            }
        }
    #region Команди руху та пауза
    public class MoveLeft : Command
        {
            public MoveLeft(PlayField playField) : base(playField) { }

            public override void Execute()
            {
                _playField.MoveLeft();
            }
        }

        public class MoveRight : Command
        {
            public MoveRight(PlayField playField) : base(playField) { }

            public override void Execute()
            {
                _playField.MoveRight();
            }
        }

        public class MoveDown : Command
        {
            public MoveDown(PlayField playField) : base(playField) { }

            public override void Execute()
            {
                _playField.MoveDown();
            }
        }

        public class MovePause : Command
        {
            public MovePause(Game game) : base(game) { }

            public override void Execute()
            {
                // змінюємо значення на протилежне
                _game.Paused = !_game.Paused;

                // відкриваємо форму паузи
                MenuPauseForm menu = new MenuPauseForm();
                menu.ShowDialog();

                // після закриття меню паузи продовжити гру
                _game.Paused = !_game.Paused;
            }
        }
    #endregion
```

2. Рефакторинг у класі UserManager, метод, що відповідає за виведення топу гравців.

   Зміни:

   - Rename Method – Підібране більш змістовне ім’я для методу, що отримує найкращих гравців.

- Change Parameter – Змінено параметр data, що повертає метод. Тепер він повертає списки імен та рахунків найкращих гравців у форматі словника (Dictionary): ключ – значення. Також обмежено словник до 10 значень.

Було:

```
public static async Task<List<string>[]> TopList()
		{
			var fileDetails = await
gitHubClient.Repository.Content.GetAllContentsByRef(owner, repoName, filePath, branch);
			XDocument xdoc = XDocument.Load(fileDetails[0].DownloadUrl);
			XElement root = xdoc.Element("users");
			var playersByTopScore = root.Elements("user");

			var names = playersByTopScore.Select(user =>
user.Attribute("name").Value).ToList().GetRange(0, 10);
			var scores = playersByTopScore.Select(user =>
user.Element("record").Value).ToList().GetRange(0, 10);
			var data = new List<string>[] { names, scores };


			return data;
		}
```

Стало:

```
public static async Task<Dictionary<string, string>> GetTopUsers()
		{
			var fileDetails = await
gitHubClient.Repository.Content.GetAllContentsByRef(owner, repoName, filePath, branch);
			XDocument xdoc = XDocument.Load(fileDetails[0].DownloadUrl);
			XElement root = xdoc.Element("users");
			var playersByTopScore = root.Elements("user").OrderByDescending(user =>
(int)user.Element("record"));

			var names = playersByTopScore.Select(user =>
user.Attribute("name").Value).ToList().GetRange(0, 10);
			var scores = playersByTopScore.Select(user =>
user.Element("record").Value).ToList().GetRange(0, 10);
			var data = names.Zip(scores, (k, v) => new { k, v })
			  .ToDictionary(x => x.k, x => x.v);

			return data;
		}
```

3. Рефакторинг звукової системи гри.

Зміни:

- Extract Class – Аудіо система додатку виділена в окремий клас.
- Encapsulate Field – Доданий модифікатор доступу private, для обмеження можливого зайвого використання полів.

Було:

```
public partial class GameForm : Form
	{
		public WaveOut sfx;
		 Stream file = new MemoryStream(Properties.Resources.MenuMusic);
		 Mp3FileReader reader = new Mp3FileReader(file);
		 Audio loop = new Audio(reader);
		 sfx = new WaveOut();
		 sfx.Volume = 1f;
		 sfx.Init(loop);
		 sfx.Play();

	}
```

```
public partial class MainForm : Form
      {
            public WaveOut sfx;

             private void OnGameOver()
             {
                  Stream file = new MemoryStream(Properties.Resources.over);
                  Mp3FileReader reader = new Mp3FileReader(file);
                  WaveOut fx = new WaveOut();
                  fx.Volume = 1f;
                  fx.Init(reader);
                  fx.Play();
             }
      }
```

Стало:

```
public class Sound
    {
        private byte[] _s;

        private WaveOut sfx;

        public Sound(byte[] s)
        {
            _s = s;
        }

        public void PlayLoop()
        {
            if (sfx == null)
            {
                Stream file = new MemoryStream(_s);
                Mp3FileReader reader = new Mp3FileReader(file);
                Audio loop = new Audio(reader);
                sfx = new WaveOut();
                sfx.Volume = 1f;
                sfx.Init(loop);
                sfx.Play();
            }
            else
            {
                Stop();
            }
        }

        public static void Play(byte[] s)
        {
            Stream file = new MemoryStream(s);
            Mp3FileReader reader = new Mp3FileReader(file);
            WaveOut fx = new WaveOut();
            fx.Volume = 1f;
            fx.Init(reader);
            fx.Play();
        }

        public void Stop()
        {
            sfx.Stop();
            sfx.Dispose();
            sfx = null;
        }
    }

public partial class MainForm : Form
      {
```

```
		private Sound menuSound = new Sound(Properties.Resources.MenuMusic);
		public MainForm()
		{
			menuSound.PlayLoop();
		}
	}

public partial class GameForm : Form
	{
		private void OnGameOver()
		{
			Sound.Play(Properties.Resources.over);
		}
	}
```

**Висновки**: я навчився виконувати реорганізацію програмного коду на підставі шаблонів рефакторінгу. Виконав аналіз програмного коду розроблюваного ПО і модульних тестів з метою виявлення погано організованого коду. Використовуючи шаблони рефакторінгу, виконав реорганізацію програмного коду розроблюваного ПО і модульних тестів. Перевірив успішність виконання всіх модульних тестів. Виконав опис вироблених операцій рефакторінгу (було-стало-шаблон рефакторінгу).