Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

**ЗВІТ**

про виконання комп’ютерного практикуму №2

з дисципліни

«Сучасні технології програмування»

Тема: «Шаблони проектування. Структурні шаблони»

Варіант 2

|  |  |
| --- | --- |
| Прийняв: | Виконав: |
| Ліщук  Катерина  Ігорівна | студент 3-го курсу  гр. ІП-51 ФІОТ  Гладишко  Валентин  Юрійович |

Київ – 2017

**ЗМІСТ:**

1. Мета РОБОТИ 3
2. Постановка задачі 4
3. Виконання завдання 5
4. Висновок 9
5. ЛІСТИНГ КОДУ ПРОГРАМИ 10

# Мета РОБОТИ

Ознайомитися з основними шаблонами проектування, навчитися застосовувати їх при проектуванні і розробці ПЗ.

# Постановка задачі

При виконанні комп'ютерного практикуму слід реалізувати наступні дії:

1. Реалізувати завдання згідно варіантів, наведених нижче (можна запропонувати реалізацію власного завдання).
2. Коротко описати архітектуру проекту.
3. Наведіть UML-діаграму класів.
4. Описати використовувані структурні паттерни. Для кожного паттерну потрібно вказати складові класи і їх призначення.

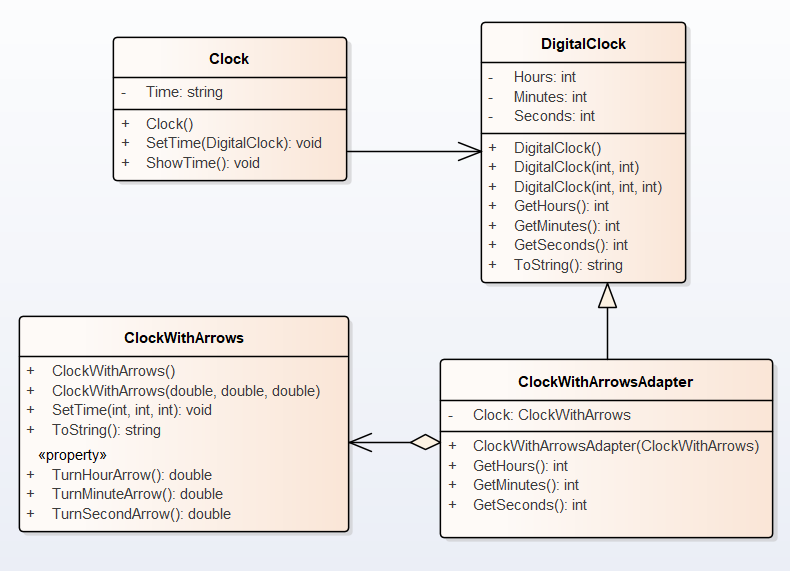
Реалізувати завдання "Годинник". Повинна бути реалізована можливість використання годинника зі стрілками так само, як і цифрового годинника. У класі "Годинник зі стрілками" зберігаються повороти стрілок.

# Виконання завдання

* 1. **Короткий опис архітектури проекту**

В програмі є основний клас "Годинник". Він містить метод який показує час та метод який отримує час з "Цифрового Годинника". Клас "Цифровий Годинник" зберігає секунди, хвилини та години в цифровому форматі. Клас "Годинник зі стрілками" зберігає повороти секундної, хвилинної та годинникової стрілок. Клас "Адаптер годинника зі стрілками" перетворює повороти стрілок в час в цифровому форматі.

* 1. **UML-діаграма класів**

****

* 1. **Опис Паттерну, який був використаний для ПЗ.**

Для даної задачі був використаний паттерн **Адаптер**.

**Адаптер** - це структурний паттерн проектування, який забезпечує спільну роботу класів з несумісними інтерфейсами.

Складові класи:

* **Існуючий інтерфейс** або клас вже працюючий і сумісний з основним кодом програми.
* **Сервіс** - це якийсь корисний клас, зазвичай сторонній, що потребує адаптації для Клієнта.
* **Адаптер** реалізує Клієнтський інтерфейс і містить посилання на об'єкт Службового класу.

Адаптер отримує виклики від Клієнта через методи Клієнтського інтерфейсу, а потім конвертує їх в виклики методів обгорненого об'єкта в правильному форматі.

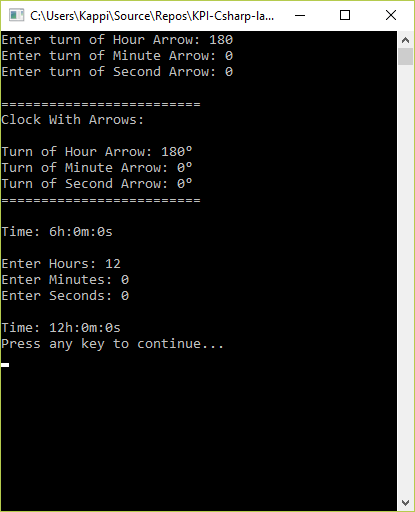
* **Клієнт** використовує Адаптер через Існуючий інтерфейс.

Це дозволяє додавати в програму нові класи Адаптерів незалежно від клієнтського коду (це потрібно якщо інтерфейс Сервісу раптом зміниться, наприклад після виходу нової версії сторонньої бібліотеки).

Цей паттерн був використаний тому що в нас є два класи для зберігання інформації про час: "Цифровий годинник" і "Годинник зі стрілками". Клас "Годинник" використовує клас "Цифровий годинник", тому нам потрібно створити клас "Адаптер годинника зі стрілками", який дасть змогу конвертувати повороти стрілок в час.

* 1. **Результати виконання програми**

*Результат виконання:*



# Висновок

Отже, я ознайомився з основними шаблонами проектування і навчився застосовувати їх при розробці ПЗ. За допомогою шаблону Адаптер я реалізував своє завдання згідно варіанту. Також я побудував діаграму класів для своєї предметної області враховуючи вимоги шаблону.

Посилання на виконану лабораторну:

<https://github.com/ValentineGladyshko/KPI-Csharp-labs/tree/master/lab2.1>

1. **ЛІСТИНГ КОДУ ПРОГРАМИ**

«Program.cs»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab2.\_1

{

class Program

{

static void Main(string[] args)

{

Console.Write("Enter turn of Hour Arrow: ");

double HourTurn = Convert.ToDouble(Console.ReadLine());

Console.Write("Enter turn of Minute Arrow: ");

double MinuteTurn = Convert.ToDouble(Console.ReadLine());

Console.Write("Enter turn of Second Arrow: ");

double SecondTurn = Convert.ToDouble(Console.ReadLine());

ClockWithArrows ClockArrow = new ClockWithArrows(HourTurn, MinuteTurn, SecondTurn);

Console.WriteLine(ClockArrow.ToString());

Clock Clock = new Clock();

Clock.SetTime(new ClockWithArrowsAdapter(ClockArrow));

Clock.ShowTime();

Console.Write("\nEnter Hours: ");

int Hours = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Minutes: ");

int Minutes = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Seconds: ");

int Seconds = Convert.ToInt32(Console.ReadLine());

//ClockArrow.SetTime(Hours, Minutes, Seconds); //показує трасформацію часу в повороти стрілок

//Console.WriteLine(ClockArrow.ToString());

Console.WriteLine();

Clock.SetTime(new DigitalClock(Hours, Minutes, Seconds));

Clock.ShowTime();

Console.WriteLine("Press any key to continue...");

Console.ReadKey(true);

}

}

}

«Clock.cs»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab2.\_1

{

class Clock

{

private string Time;

public Clock()

{

Time = "";

}

public void SetTime(DigitalClock clock)

{

Time = String.Format("{0}h:{1}m:{2}s", clock.GetHours(), clock.GetMinutes(), clock.GetSeconds());

}

public void ShowTime()

{

Console.WriteLine("Time: {0}", Time);

}

}

}

«DigitalClock.cs»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab2.\_1

{

class DigitalClock

{

private int Hours;

private int Minutes;

private int Seconds;

public DigitalClock()

{

Seconds = 0;

Minutes = 0;

Hours = 0;

}

public DigitalClock(int hours, int minutes)

{

Hours = (hours + minutes / 60) % 24;

Minutes = minutes % 60;

Seconds = 0;

}

public DigitalClock(int hours, int minutes, int seconds)

{

Hours = (hours + (minutes + seconds / 60) / 60) % 24;

Minutes = (minutes + seconds / 60) % 60;

Seconds = seconds % 60;

}

public virtual int GetSeconds()

{

return Seconds;

}

public virtual int GetMinutes()

{

return Minutes;

}

public virtual int GetHours()

{

return Hours;

}

public override string ToString()

{

return "\n=========================\nDigital Clock:\n\nHours: " + Hours + "\nMinutes: " +

Minutes + "\nSeconds: " + Seconds + "\n=========================\n";

}

}

}

«ClockWithArrows.cs»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab2.\_1

{

class ClockWithArrows

{

public double TurnSecondArrow { get; protected set; }

public double TurnMinuteArrow { get; protected set; }

public double TurnHourArrow { get; protected set; }

public ClockWithArrows()

{

TurnHourArrow = 0;

TurnMinuteArrow = 0;

TurnSecondArrow = 0;

}

public ClockWithArrows(double turnHourArrow, double turnMinuteArrow, double turnSecondArrow)

{

TurnHourArrow = ((turnSecondArrow + turnMinuteArrow \* 60 + turnHourArrow \* 720) / 720) % 720;

TurnMinuteArrow = ((turnSecondArrow + turnMinuteArrow \* 60 + turnHourArrow \* 720) % 21600) / 60;

TurnSecondArrow = ((turnSecondArrow + turnMinuteArrow \* 60 + turnHourArrow \* 720) % 360);

}

public void SetTime(int hours, int minutes, int seconds)

{

TurnHourArrow = ((double)(seconds \* 6 + minutes \* 360 + hours \* 21600) / 720) % 720;

TurnMinuteArrow = ((double)(seconds \* 6 + minutes \* 360 + hours \* 21600) % 21600) / 60;

TurnSecondArrow = ((double)(seconds \* 6 + minutes \* 360 + hours \* 21600) % 360);

}

public override string ToString()

{

return "\n=========================\nClock With Arrows:\n\nTurn of Hour Arrow: " + TurnHourArrow +

"°\nTurn of Minute Arrow: " + TurnMinuteArrow + "°\nTurn of Second Arrow: " +

TurnSecondArrow + "°\n=========================\n";

}

}

}

«ClockWithArrowsAdapter.cs»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab2.\_1

{

class ClockWithArrowsAdapter : DigitalClock

{

private ClockWithArrows Clock;

public ClockWithArrowsAdapter(ClockWithArrows clock)

{

Clock = clock;

}

public override int GetSeconds()

{

return (int)(Clock.TurnSecondArrow / 6);

}

public override int GetMinutes()

{

return (int)(Clock.TurnMinuteArrow / 6);

}

public override int GetHours()

{

return (int)(Clock.TurnHourArrow / 30);

}

}

}