Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №6

Технології розроблення програмного забезпечення

«Шаблони «Abstract Factory», «Factory Method»,

«Memento», «Observer», «Decorator»»

Варіант 24

Виконав
студент групи ІА–33:
Вовчок М.М.

Перевірив
Мягкий М. Ю.

Київ — 2025

Тема

Реалізація шаблону Factory Method для створення дій системи Flexible Automation Tool.

Короткі теоретичні відомості

Шаблон Factory Method забезпечує створення об'єктів через фабричний метод, який повертає абстрактний тип, тоді як конкретний тип об'єкта визначається у підкласах. Таким чином, клієнтський код не залежить від конкретних реалізацій і працює лише з абстракціями, що підвищує гнучкість та розширюваність програмної системи.

Завдання

• Ознайомитись з короткими теоретичними відомостями.
• Реалізувати частину функціоналу гнучкого інструменту автоматизації у вигляді класів.
• Реалізувати Factory Method відповідно до обраної теми.
• Реалізувати не менше трьох класів.
• Підготувати звіт з діаграмою класів та фрагментами коду.

Реалізація шаблону Factory Method

У системі Flexible Automation Tool правила автоматизації складаються з набору дій. Кожна дія має власний тип і параметри. Щоб не прив'язувати код до конкретних реалізацій дій, застосовано шаблон Factory Method.

Фрагменти коду реалізації

Інтерфейс IAutomationAction:

```
public interface IAutomationAction
{
   string Name { get; }
   void Execute();
}
```

Клас DownloadFileAction:

```
public class DownloadFileAction : IAutomationAction
{
   public string Name => "DownloadFile";
   ...
   public void Execute()
   {
      Console.WriteLine($"[ACTION] Downloading '{_url}'...");
   }
}
```

Клас SetStatusAction:

```
public class SetStatusAction : IAutomationAction
{
   public string Name => "SetStatus";
   ...
}
```

Клас SendNotificationAction:

```
public class SendNotificationAction : IAutomationAction
{
   public string Name => "SendNotification";
   ...
}
```

Фабрики створення дій

```
public interface IActionCreator
{
   IAutomationAction CreateAction(string parameters);
}
```

Клас DownloadFileActionCreator:

```
public class DownloadFileActionCreator : IActionCreator
{
```

```
    public IAutomationAction CreateAction(string parameters)
    {
        return new DownloadFileAction(url, dest);
    }
}
```

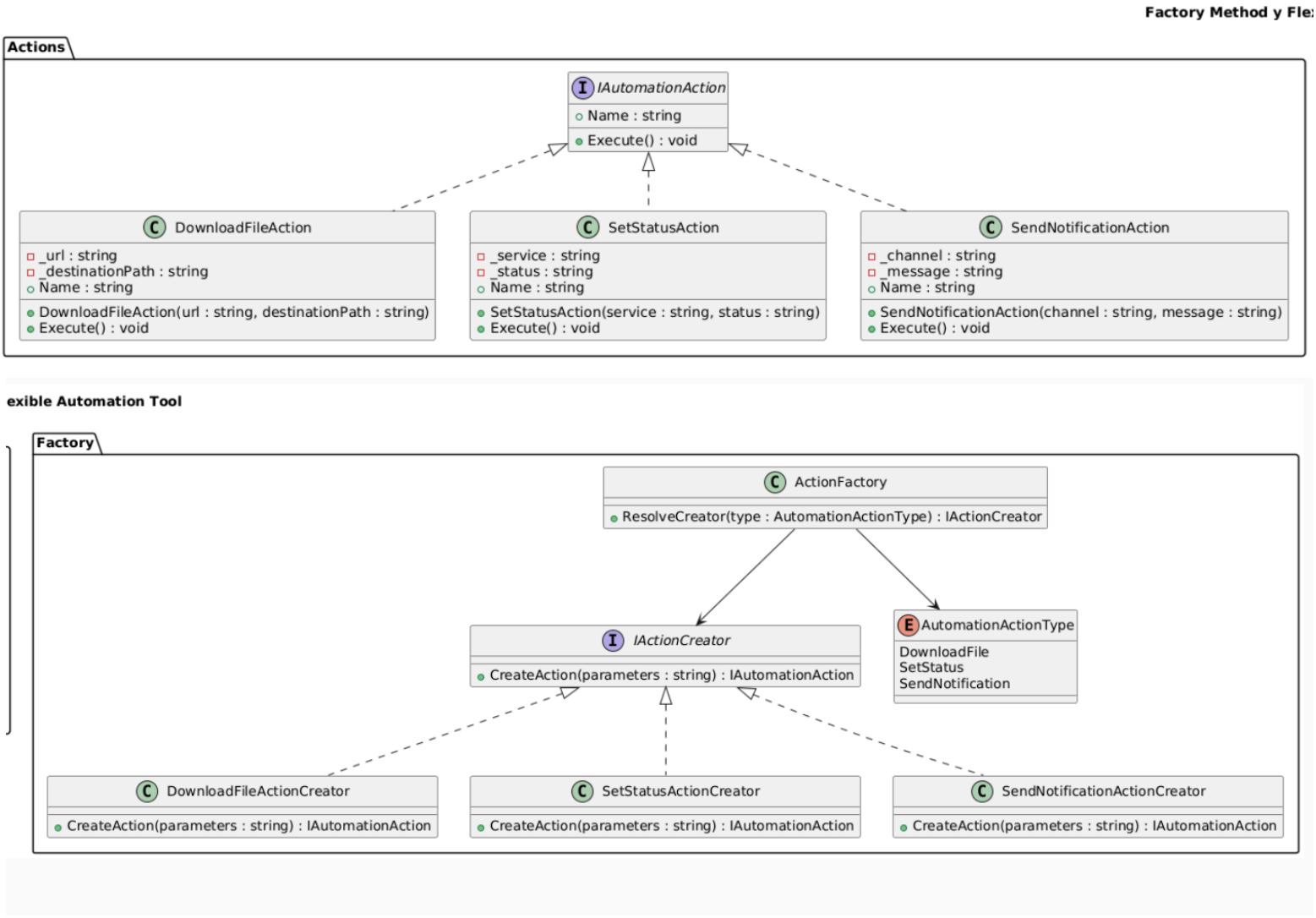
ActionFactory

```
public static class ActionFactory
{
    public static IActionCreator ResolveCreator(AutomationActionType type)
    {
        return type switch
        {
            AutomationActionType.DownloadFile => new DownloadFileActionCreator(),
            AutomationActionType.SetStatus => new SetStatusActionCreator(),
            AutomationActionType.SendNotification => new SendNotificationActionCreator(),
            _ => throw new ArgumentOutOfRangeException()
        };
    }
}
```

Результат виконання програми

```
=== FACTORY METHOD DEMO ===
Executing action: DownloadFile
[ACTION] Downloading 'https://example.com/file.zip' to '/tmp/file.zip'...
Executing action: SetStatus
[ACTION] Setting status 'Away' for 'Skype'...
Executing action: SendNotification
[ACTION] Sending notification via 'email': Automation completed
```

Діаграма класів



На діаграмі відображені інтерфейси, конкретні дії та фабрики, що реалізують шаблон Factory Method у системі Flexible Automation Tool.

Висновок

У лабораторній роботі реалізовано шаблон Factory Method для динамічного створення допустимих дій у системі автоматизації. Дано програмний код та продемонстровано його роботу.