**현장실습 주간보고서[2주차]**

**21.1.11 ~ 21.1.15**

**아주대 미디어학부 황지훈**

**펜타큐브-개발**

**1. coding convention 피드백**

git을 이용하여 피드백을 받았다. 피드백의 주요 내용은 코드는 길이가 약간 길더라도 가독성이 좋게 구조를 설정하고 컴포넌트들을 명명해야 한다는 것이다. 따라서 많은 부분에서 수정사항이 발생했다. 우선 interface는 독립적인 cs파일에 위치해야 한다. 또한 변수, property, 함수의 이름은 다른 사람이 봤을 때 이해하기 쉬워야 한다. 특히 함수에 반환 값이 있다면 반환을 유추할 수 있는 형태로 명명해야 한다. Class와 변수, property의 경우 명사형태로, 함수의 경우는 동사형태로 이름을 지어야 한다. 또한 Collection은 복수형으로 명명하는 것이 좋다.

코드에 들여쓰기가 너무 많이 포함되면 가독성이 낮아져 좋지 않다. 따라서 if문이 포함된 들여쓰기라면 continue키워드를 사용하거나 세부적인 로직 부분은 새로운 함수로 빼 둠으로써 들여쓰기를 줄여 가독성을 높였다. 추가적으로, 파일을 읽어 처리하는 과정에서 반복문과 조건문이 복잡한 구조를 이루고 있어서 C#의 LINQ를 사용하여 해당 부분을 간결하게 표현했다.

**2. LDRCNetClassification 로직 부분 구현**

지난주에 정규표현식과 NetClassification pair로 구분된 csv파일을 입력 받고 Dictionary에 저장하는 과정을 수행했다. 이번주에는 해당 dictionary를 기반으로 Net name 들을 입력 받아 실제로 분류하는 과정을 구현했다. Net name 파일은 각 Net이름 들이 개행문자로 구분된 txt파일형식이다. 각 Net name을 반복문을 통해 순회하면서 정규표현식이 저장된 dictionary에 대응되는 정규표현식이 존재할 때 Dictionary의 value가 의미하는 Net class를 생성했다. 이 과정은 switch문으로 구현했다. 각 Net이름과 Net class의 mapping정보(결과)는 새로운 dictionary<string, INet>를 생성하고 저장했다.

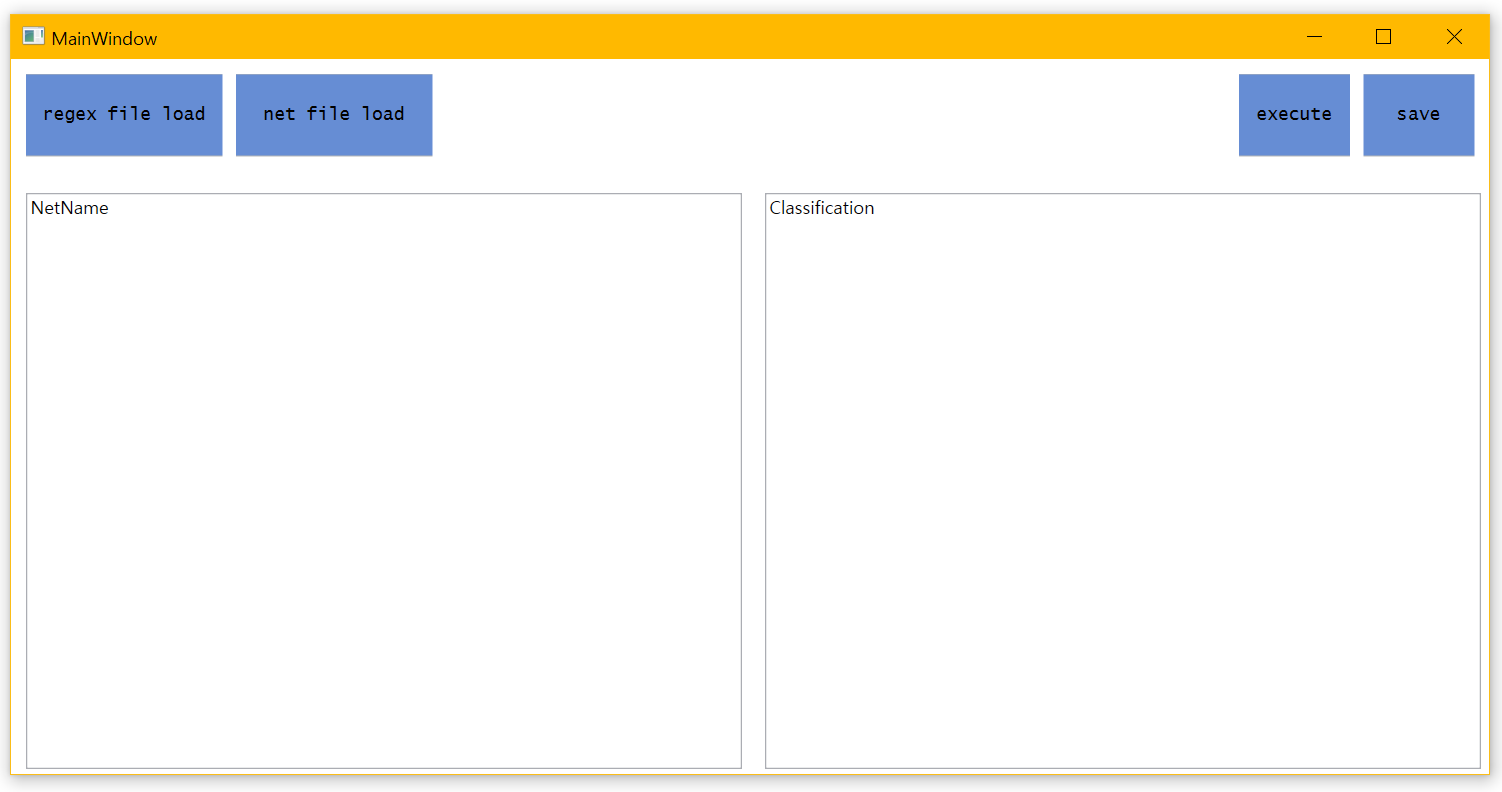
Main함수는 전체 프로세스를 control하는 역할을 하도록 수정했다. 우선 Net name과 classification을 mapping할 responsibility를 갖는 ‘NetNameMapping’ class를 생성했다. Main함수에서 NetNameMapping class의 instance를 생성하고 reference를 통해 “정규표현식 파일 등록, mapping 과정 실행” 순서대로 진행했다(함수호출). 결과를 출력하는 과정은 NetNameMapping class의 responsibility와 어울리지 않다. 따라서 결과를 출력하는 과정은 Main함수에서 담당하도록 구현했다.

**2. WPF를 이용한 LDRCNetClassification의 UI 부분 구현**

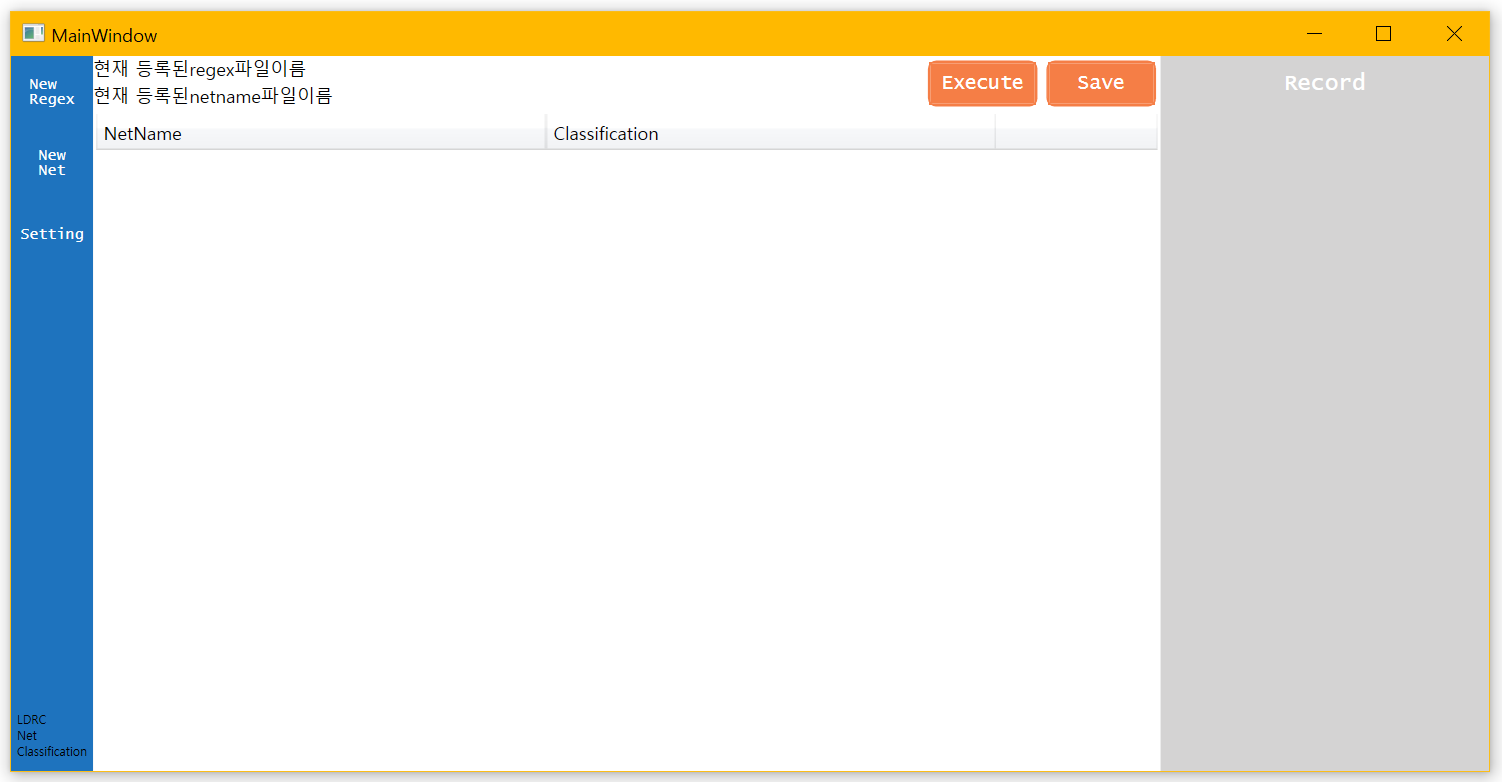
UI부분을 구현하기 위해 기존 솔루션 파일에 새로운 .NET Framework 프로젝트를 생성했다. UI프로젝트는 로직 부분을 구현한 프로젝트와 종속성을 갖도록 했다(UI프로젝트에서 로직 프로젝트의 함수를 호출한다). 단 종속성은 단방향으로 설정하여 loop가 만들어지지 않도록 해야 한다. 생성한 UI프로젝트를 시작 프로젝트로 설정하여 UI프로젝트가 먼저 실행되도록 했다.

WPF에서는 xaml을 이용하여 레이아웃, 컨트롤 등 UI컴포넌트들을 구성하고 해당 event 처리와 관련된 로직은 C#코드를 통해 처리된다. Grid, StackPanel, DockPanel등을 이용하여 레이아웃을 구성했고, 각 레이아웃 위에 Button, TextBlock, Border, DataGrid등을 올려 디자인했다. 중복되는 내용이 많이 포함되는 컨트롤에 대해서는 <Window.Resource> 테그에 <Style>과 <Setter>를 이용하여 정의해 두고 재사용했다.

다음 사진은 21년 1월 13일에 디자인한 UI초안 결과이다.



Grid를 이용해서 레이아웃을 나누고 TextBlock, Button을 올려봤다. 하지만 Button들이 위에만 위치하여 출력결과를 보여줄 공간이 낭비될 것 같아 Button을 옆으로 빼 두고, 등록된 파일 정보를 보여주는 부분과 Save한 기록을 남길 수 있는 공간을 추가로 만들었다.



등록된 파일이름이 보여지는 부분에서는 DockPanel을 이용하여 Grid보다 간단하게 코드를 작성할 수 있었다. 그리고 로직 연산 결과를 보일 부분은 DataGrid를 이용했다. 이 부분은 다음주차에 결과들을 채워 display할 계획이다.

**3. Visual Studio에서 디버깅하기**

Visual studio에서 break point에 대해서는 알고 있었지만, 조사식을 활용한 디버깅 방법을 새로 알게 되어 사용해봤다. 조사식을 활용하면 실행 중에 변수 값 등을 확인할 수 있어 매우 유용했다. 또한 UI의 경우 디버깅을 하면 컨트롤 들의 경계라인을 확인할 수 있었다.

**4. 소감 및 차주계획**

WPF를 처음 다뤄보면서 xaml을 통해 UI를 구성할 수 있었고 UI구현은 매우 어렵고 세밀한 작업임을 알게 되었다. 다음주에는 펜타큐브의 라이브러리를 사용하여 좀 더 의미 있는 데이터를 이용하여 테스트해 볼 것이다.