|  |
| --- |
|  |
| 여행 서비스 XML GUI |
|  |

|  |
| --- |
| 2022-12-11 |

# 목차

1. 구현정도

2. 구현 방법 및 메뉴얼

3. GUI 설계 내역과 구조 등의 설명

4. GUI 실행 소스 및 설명

5. GUI 작동 결과 및 화면 캡쳐

1. 구현정도

1~9번까지의 기능이 구현되도록 하였습니다. 하지만 update의 경우 제 xml파일 내에 같은 element 명이 많아, 부모 element에 모두 id attribute를 만들어 이를 통해 구분하도록 하였습니다. 중복된 element가 없어 id attribute가 없어도 진행되도록 해 놓았습니다. 하지만 중복된 이름의 element들(user, guide travel\_product, place 등)은 id로 구분하여 업데이트할 수 있도록 했습니다. 또한 user\_name과 같이 중복된 element들은 부모의 id(여기서는 user)를 통해 찾을 수 있도록 하였습니다. Validation의 경우에는 load를 수행하여 파일을 두개 고를 경우에는 XSD Validation, 한 개를 고를 경우에는 DTD Validation이 수행되도록 하였습니다.

2. 구현 방법 및 매뉴얼

개발 환경은 Eclipse 컴파일러를 활용하여 했습니다. 따라서 Eclipse로 돌릴 경우 Ctrl+F11을 눌러 컴파일 할 경우 GUI가 뜨는 것을 볼 수 있습니다. 그 외에 CMD 창으로 할 경우에는 저의 프로잭트 파일로 가서 ‘javac 파일명.java’를 먼저 해야됩니다. 하지만 저의 개발 환경에서는 버전 차이의 문제로 위처럼 치지 않고 ‘javac 파일명.java –release 8’과 같이 쳐 버전 차이를 극복합니다. 이후 파일 내에 class가 생성되면 ‘java 파일명’을 치면 GUI가 똑같이 뜨는 것을 확인할 수 있습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림1 버전 차이로 인한 오류 화면

그림텍스트이(가) 표시된 사진

자동 생성된 설명 2 버전 문제 해결 후 자바 컴파일 완료된 화면

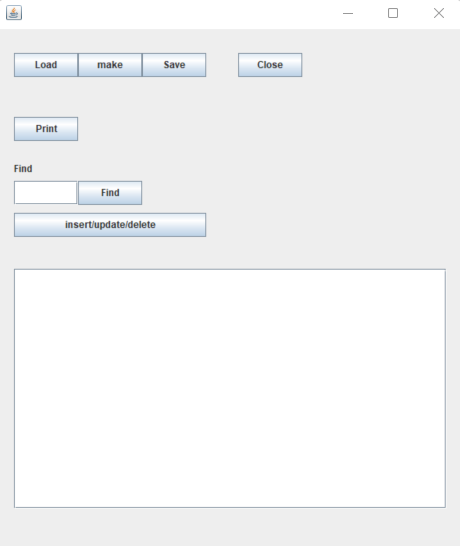
Load 할 경우에

XMl\_FinalProject\_18011881\_XSD.xml 을 고르고 뒤에 XMl\_FinalProject\_18011881\_main.dtd 파일을 골라야 오류가 뜨지 않고 validation이 되는 것을 확인할 수 있습니다.

또는 XMl\_FinalProject\_18011881\_DTD.xml을 고르고 뒤에 load창은 취소를 할 경우에도 DTD Validation으로 오류 없이 뜨는 것을 확인할 수 있으니 참고바랍니다.

3. GUI 설계 내역과 구조 등의 설명

GUI 화면과 같이 설명하겠습니다. 그림3 참고

 Load: XML 파일을 Load 하는 버튼입니다.

Make: 새로운 XML을 만드는 버튼입니다.

Save: XML을 저장하는 버튼입니다.

Close: 프로그램 종료 버튼입니다.

Print: XML 파일 전체 출력 버튼입니다.

검색 창: Find를 할 때 필요한 검색어 입력 부분입니다.

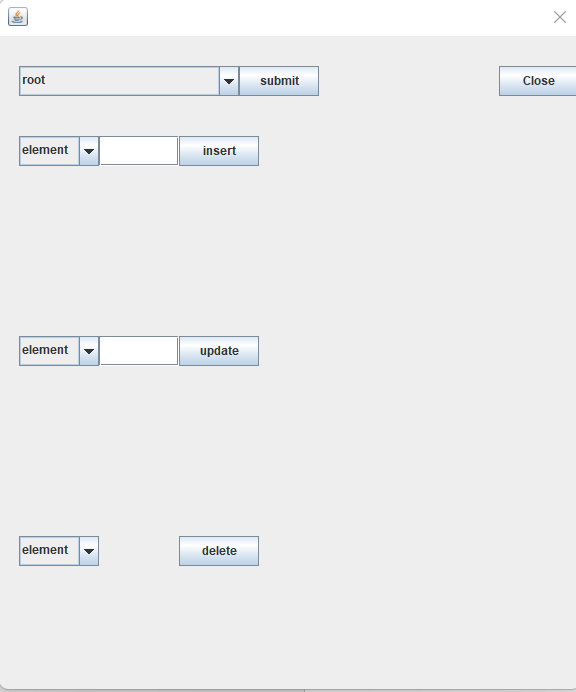
Find: Find 실행 버튼입니다.

Insert/update/delete: XML에 각각 입력, 수정 삭제를 할 수 있는 새 창으로 갑니다.

텍스트 창: Print 또는 Find 실행 시, Data 출력을 나타내는 부분입니다.

그림3-1 GUI 설계

다음은 insert/update/delete 버튼을 누를 시 뜨는 창입니다.

그림3-2 GUI 설계

상단 콤보 박스: element를 선택하는 부분입니다.

Submit: 선택 완료시 submit 버튼을 누릅니다.

Close: 해당 창을 닫습니다.

콤보 박스1: 어떤 node type을 넣을지 고릅니다.

입력 창 1: insert 할 node 이름을 입력합니다.

Insert: insert 실행 버튼입니다.

콤보 박스2: 어떤 node type을 수정할지 고릅니다.

입력 창 2: update 할 새 node 이름을 입력합니다.

update: update 실행 버튼입니다.

콤보 박스3: 어떤 node type을 삭제할지 고릅니다.

Delete: delete 실행 버튼입니다.

4. GUI 실행 소스 및 설명

프로잭트 파일 내에는 6개의 자바 파일이 있습니다. 그림4 참고

텍스트이(가) 표시된 사진

자동 생성된 설명

그림4 프로잭트 파일

data\_window 파일은 insert/update/delete 기능을 수행하는 창이 있습니다.

DOMNodeFind 파일은 find에 사용되는 클래스가 있는 파일입니다.

DOMTraverse 파일은 print에 사용되는 클래스가 있는 파일입니다.

gui\_project221126 파일은 메인 창입니다. Gui 구현 및 클래스를 불러오는 메인 파일입니다.

ValidationCheck\_DTD 파일은 DTD검증에 사용되는 클래스가 있는 파일입니다.

ValidationCheck\_XSD 파일은 XSD검증에 사용되는 클래스가 있는 파일입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 5 gui\_project221126 파일(이하 메인 파일)의 코드 일부

다음 그림 5에는 각각의 Gui 버튼의 기능 구현 코드가 나와 있습니다. btnOpen이란 버튼을 누르면 FileChooser를 불러와 파일을 고르도록 합니다. btnPrint를 누르면 DomTraverse를 새로 선언하며 클래스 파일을 동작하도록 합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명이 파일에는 context라는 클래스 변수를 추가하였습니다. 그래서 각각의 XML NODE 값을 여기에 저장하도록 하였습니다. 이후에는 getData라는 메소드를 만들어 메인으로 값을 불러드렸습니다. 특히 ELEMENT 노드에는 ATTRIBUTE가 있을 수 있어 따로 처리한 모습을 볼 수 있습니다.

그림6 Domtravese 파일 일부

텍스트이(가) 표시된 사진

자동 생성된 설명 다시 그림 5로 돌아가면 node\_submit이라는 버튼이 있습니다. 여기에서는 DOMNodeFind 클래스에 xml 경로와 검색어를 넣습니다. 이후에는 Find 기능을 수행합니다.

DOMNodeFind 안에는 traverse라는 이름으로 이전 print와 비슷한 코드가 있습니다. 다른 점은 각각의 node마다 if문이 있다는 것입니다. Node.getNodeName.equals(eleName) 이라는 구문으로 검색어와 같은지에 대한 Boolean 값을 찾습니다. 이를 통해 if문을 돌리어 만약에 값이 같을 경우 context라는 클래스 변수에 넣습니다. 이후에는 getData를 통해 메인 파일로 넘깁니다. ATTRIBUTE, TEXT, COMMENT는 getNodeName이 아닌 getNodeValue도 참고했습니다.

그림7 DOMNodeFind 일부

텍스트이(가) 표시된 사진

자동 생성된 설명다음은 메인 파일에 save 파트입니다. 역시 JFileChooser를 사용하여 경로를 가져옵니다. 이후 수정 사항이 있었던 xml file tree를 doc 변수를 통해 저장하여 경로에다가 저장합니다.

그림 7 메인 파일 일부

다음은 insert/update/delete 기능을 수행하는 data\_window 파일입니다.

그림 8 data\_window 파일 일부

텍스트이(가) 표시된 사진

자동 생성된 설명

맨 처음 노드를 고르는 경우입니다. 126-129줄에서 id 속성이 있는지 판단합니다.

만약 있는 경우 134-136에서 id값을 입력 받습니다. 이후 150줄에서 이름과 id 값으로 해당 node를 찾습니다. Id가 없는 경우 146줄에서 이름으로만 찾습니다. 저의 xml의 파일에는 동일한 node 명이 많아, 모두 id 속성 처리를 해놓았습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명그림 9 data\_window 파일 insert 함수 일부

Insert 함수 내부입니다. 그 중에서도 element 입력 부입니다. If 문안에 있는 문구로 입력시에는 자동으로 id값이 추가되도록 하기 위해 다음과 같이 짰습니다. 만약 안에 자식이 있는 경우에는 insertBefore, 없는 경우에는 appendChild를 이용하여 insert했습니다.

그림 9-1 data\_window 파일 insert 함수 일부

텍스트이(가) 표시된 사진

자동 생성된 설명

Attribute의 경우 attribute를 입력 받고, attribute value 값도 입력을 받을 필요가 있습니다. 따라서 showInputDialog를 통해 value 값을 입력 받아서 insert를 한 코드입니다.

다음은 update 부분입니다. Update 부분은 특이한 것이 node 들의 종류 중 attribute를 고를 때 att name과 att value 두가지로 고를 수 있게 한 것입니다. Att name은 attribute node 명을 바꾸는 것이고, att value는 node 값을 바꾸는 것입니다. 그림 10 참고

그림텍스트이(가) 표시된 사진

자동 생성된 설명 10 data\_window update 함수 일부

284는 element type이 입력된 text값으로 바로 업데이트 되는 파트입니다.

287부터는 속성 이름을 바꾸는 파트입니다. Node 내부에 있는 모든 속성을 이름과 비교하고 만약 text와 이름이 같은 경우에 바꿀 이름을 value로 입력 받고 이를 수정합니다. 만약 없는 경우에는 297에서 메시지를 출력합니다.

304에서는 같은 방법으로 속성 value를 수정합니다. 속성 이름과 비교 후 같은 text가 있으면 바꿀 이름이 아닌 value값을 입력받습니다. 이후 이를 update합니다.

그림텍스트이(가) 표시된 사진

자동 생성된 설명 11 data\_window 파일 delete 함수 일부

텍스트이(가) 표시된 사진

자동 생성된 설명

Delete의 경우 element는 이미 선택된 node를 지웁니다. 하지만 attribute는 어떤 attribute를 지울지를 이름으로 입력받습니다. 텍스트의 경우에는 선택된 node의 자식 중 알파벳, 숫자로 구성된 text가 있는 경우에 지우도록 했습니다. 해당 node에 조건을 만족하는 text가 없다면 메시지를 띄웁니다. Comment도 같은 방식으로 했습니다.

5. GUI 작동 결과 및 화면 캡쳐

작동 결과는 버튼을 설명했던 순서대로 보이도록 하겠습니다.

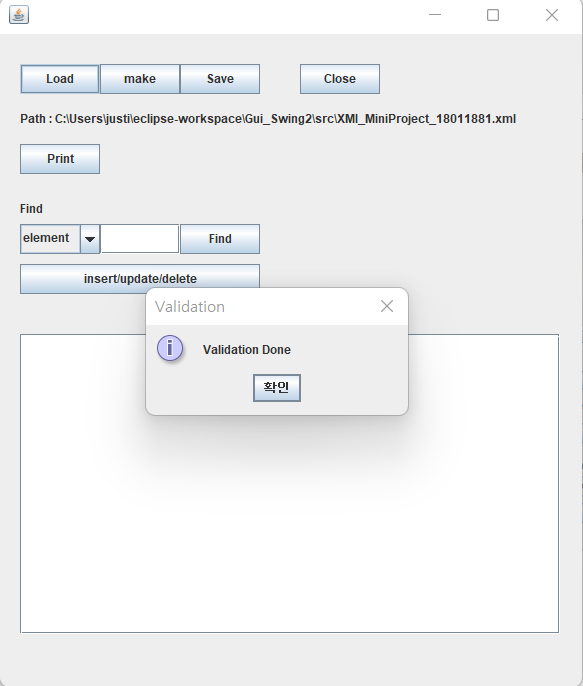
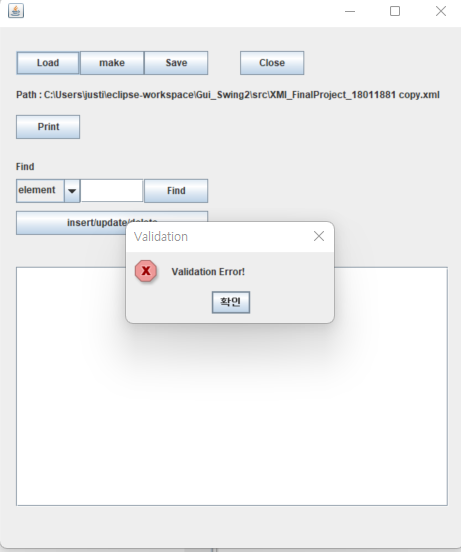
텍스트이(가) 표시된 사진

자동 생성된 설명그림12 Load 작동 결과

로드 버튼을 누를 경우, FileChooser가 뜨는 것을 볼 수 있습니다. 이를 통해 프로잭트 파일 내의 XML 파일을 골라서 Load 시킬 수 있습니다. Save 실행도 옆과 같습니다.

Xml 파일을 연 경우 연속으로 한 번 더 뜹니다. 이때 xsd 파일을 고르면 xsd validation, 고르지 않으면 dtd validation이 진행됩니다.

Validation 오류가 뜬 경우 에러 메시지가 뜨고,

오류가 없으면 바로 완료 메세지가 뜹니다. 

텍스트이(가) 표시된 사진

자동 생성된 설명

Load 한 경우, 위에는 XML 파일의 경로가 뜨는 것을 볼 수 있습니다. Validation에 에러가 있어도 뜹니다.

이후 Print 버튼을 누를 경우 아래에 XML 파일을 모두 볼 수 있게 출력됩니다. 스크롤을 통해 끝까지 내용을 볼 수 있도록 구현하였습니다.

그림9 Print 작동 결과

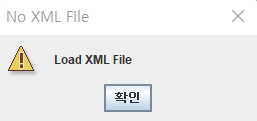
Make, close를 제외한 모든 버튼은 로드를 하기 전까지 작동하지 않습니다. 만약 버튼을 누를 경우 다음과 같은 오류 창이 뜹니다.

그림9-1 오류 창

테이블이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

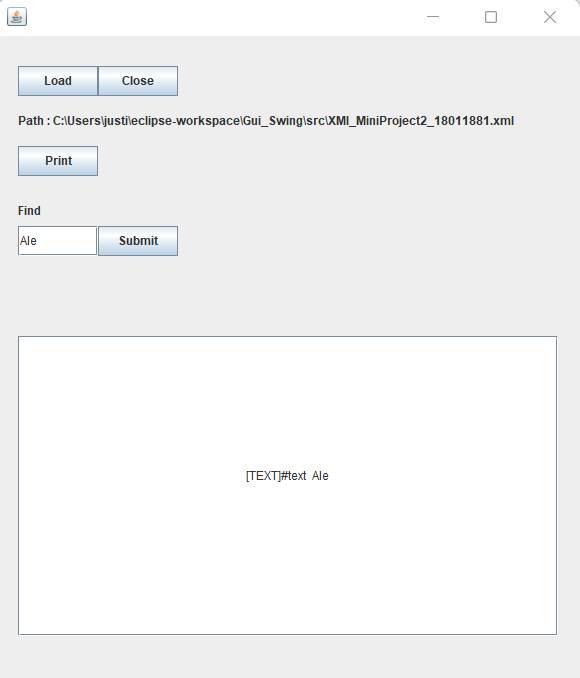
자동 생성된 설명

그림10-12 Find 출력 결과

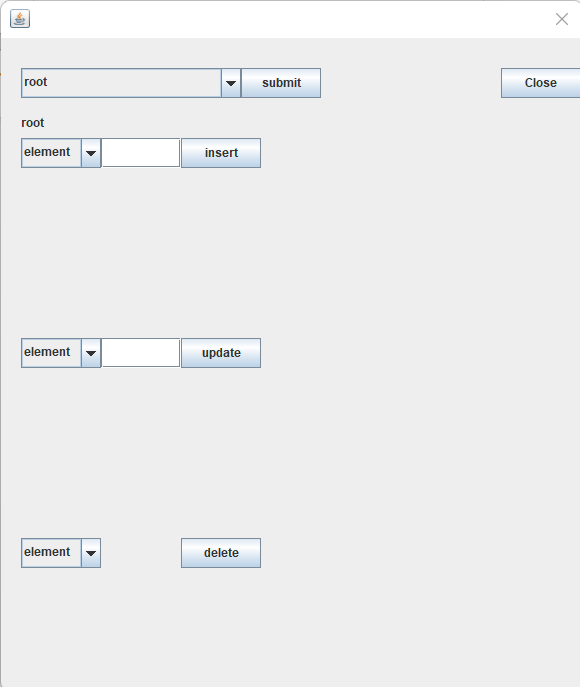
검색창에 특정 텍스트를 입력할 경우, 그 값과 같은 어떤 Node든 다 출력합니다. 그림 10은 ATTRIBUTE 또는 TEXT 값을 숫자로 찾아온 예시입니다. 그림 11은 User의 이름인 Ale를 검색하여 출력한 값입니다. 그림 12는 user 라는 ELEMENT를 모두 출력한 값입니다. 이를 추후에 insert와 같은 기능을 할 때는 Node도 입력을 받을 예정입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명make 버튼을 누를 경우 load했던 xml 파일이 있으면 바로 save 기능이 실행됩니다.

이후 root element 명을 정하도록 합니다. 이름을 입력하고 확인하면 이름에 해당하는 root element가 실행됩니다. 취소를 누를 경우 root라는 이름으로 배정됩니다.

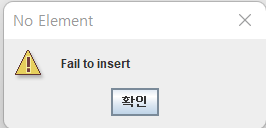
빈칸으로 확인버튼을 누르면 실행이 되지 않습니다.



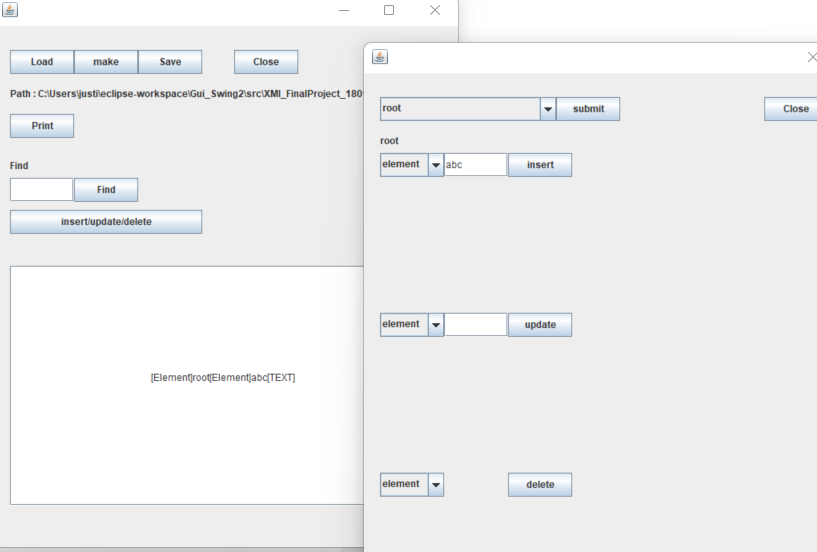
Insert/update/delete

버튼을 누를 경우 다음 창이 뜨며

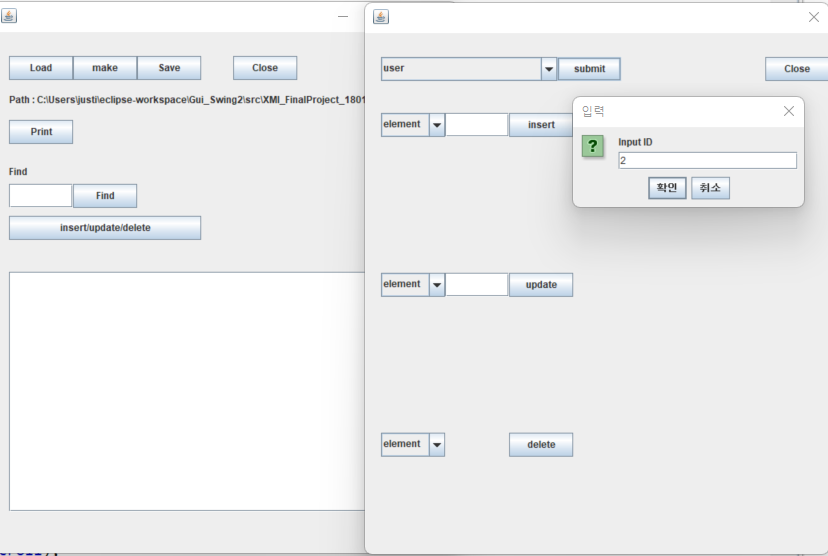
Element 명을 고르고 submit 하면 element 가 선택됩니다. Element를 선택하지 않고 아래의 버튼을 작동할 시에는 오류가 뜹니다.



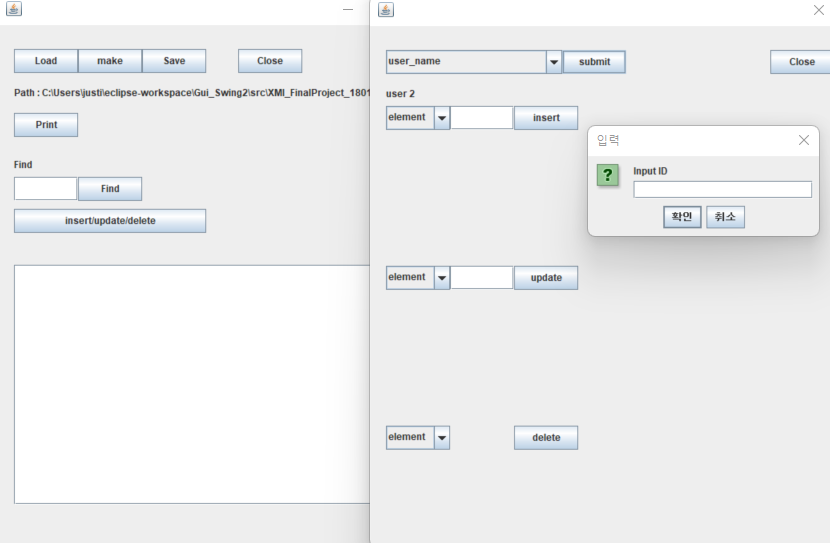
**INSERT**



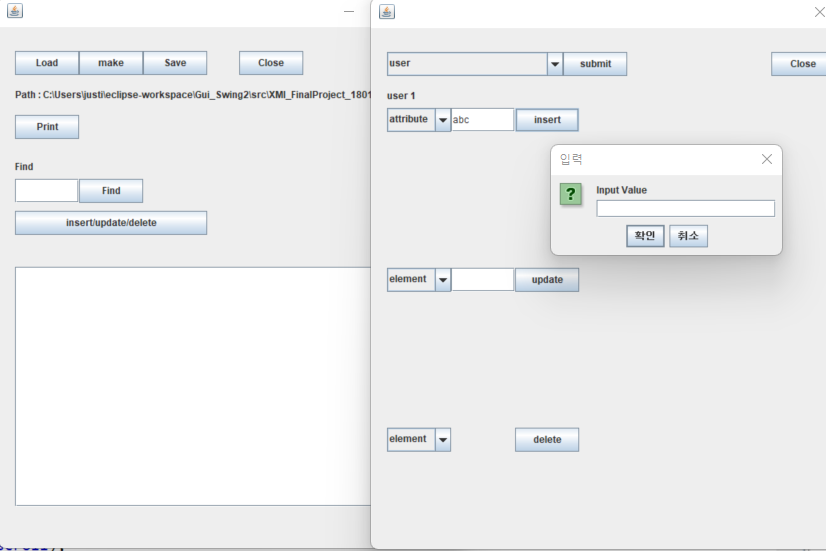
Root에 abc element를 insert한 예입니다. 오류가 뜨지 않으면 메인 창에서 print를 하면 반영되어 있는 것을 볼 수 있습니다.



동일한 user element에 대해 id를 입력받는 예입니다. User 중 속성에 id값이 2인 node를 선택합니다.



User의 이름을 바꾸고 싶어 user\_name을 고른 경우, user\_name의 부모의 id를 확인하기 위해 또 id를 입력받습니다. 모든 node는 선택되는 당시에 부모로 계속 올라가보며 id 속성을 가지고 있는 조상이 있는 경우를 확인합니다. 이후 있으면 무조건 id를 입력 받아 node를 찾을 수 있도록 합니다. 없는 경우에는 바로 node가 선택됩니다.

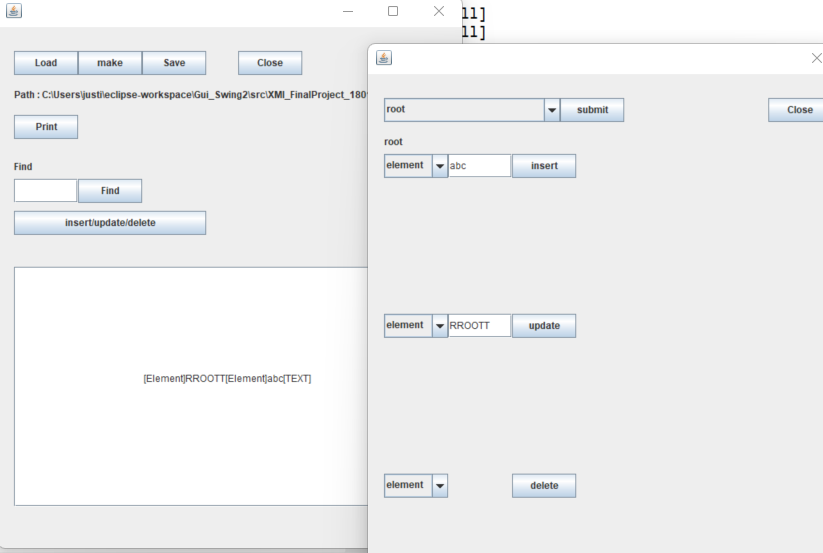


Attribute insert를 하는 경우 node 명을 입력하고 insert를 누릅니다. 그러면 value값을 입력도록 창이 뜹니다.

 이는 insert된 것을 print에서 가져온 것입니다.

Text와 comment는 element와 동일하게 적용됩니다.

**UPDATE**



root의 element 명을 update한 예시입니다. 이름이 RROOTT로 바뀐 것을 볼 수 있습니다.

텍스트, 스크린샷, 실내이(가) 표시된 사진

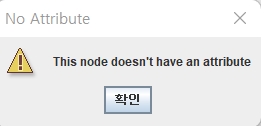
자동 생성된 설명

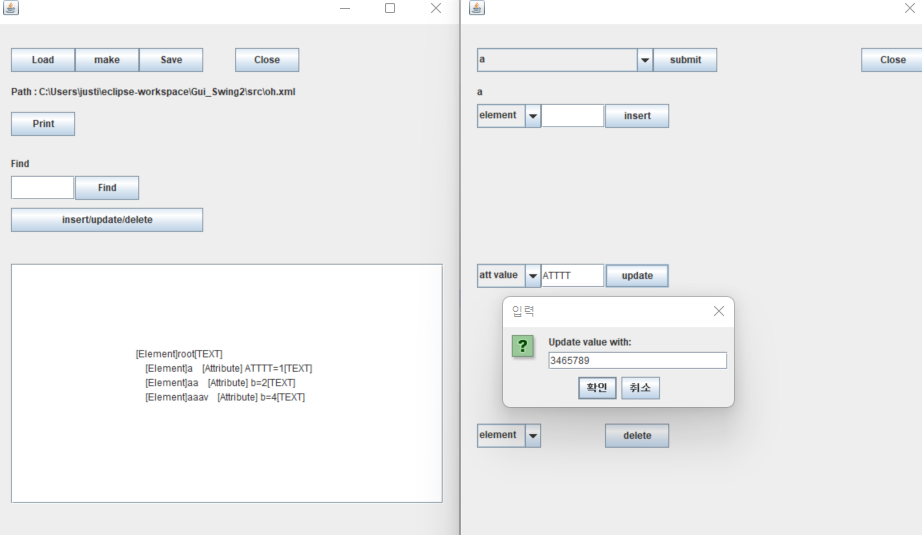
a라는 node를 선택하고 att name으로 콤보 박스를 설정합니다. b라는 속성 이름을 바꾸려고 합니다. 그럼 다음과 같은 창이 뜨며 이름을 적습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명이는 바뀐 결과입니다.

만약 없는 attribute 명을 입력할 시에는 오류가 뜹니다.





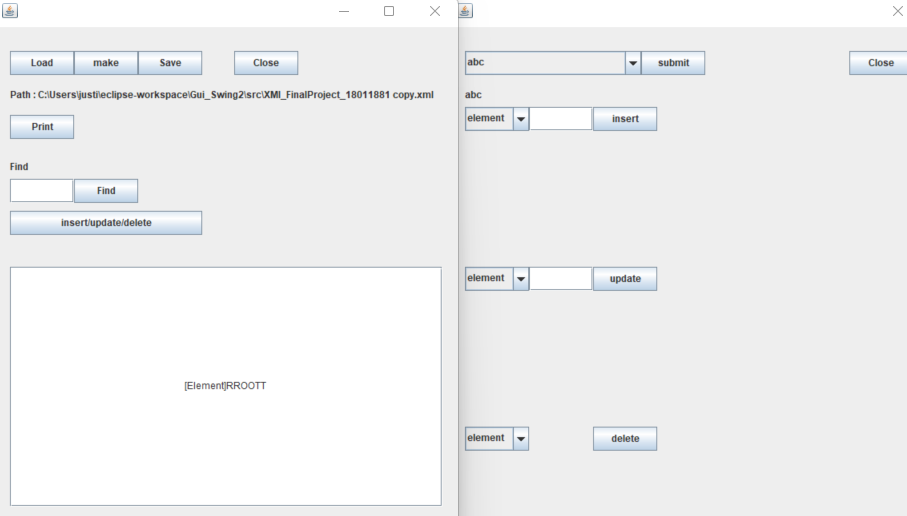
다음은 attribute value 값 수정입니다. Att value로 콤보 박스를 설정하고 해당하는 attribute 명을 입력하면 바꿀 값을 입력하라는 창이 뜹니다.

텍스트이(가) 표시된 사진

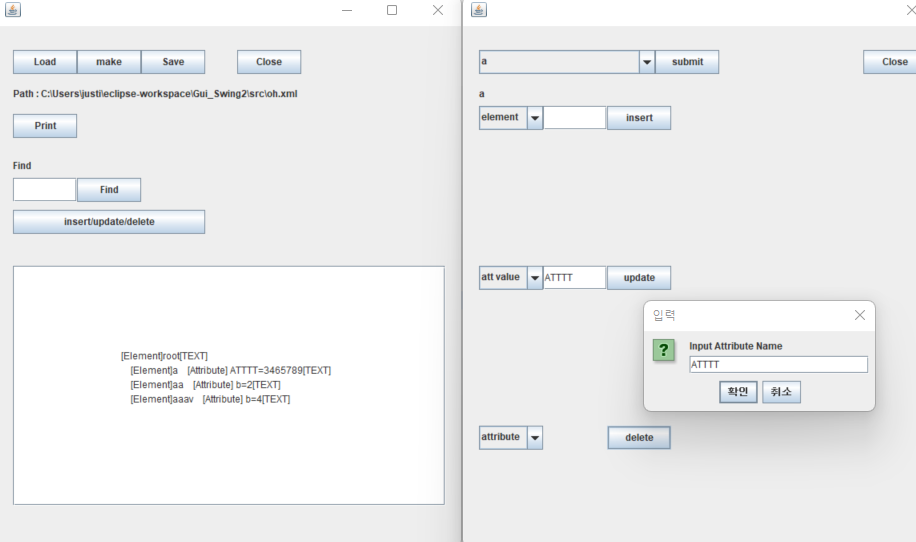
자동 생성된 설명결과값은 다음과 같습니다.

Text와 comment는 element와 작동 원리가 같습니다.

**DELETE**



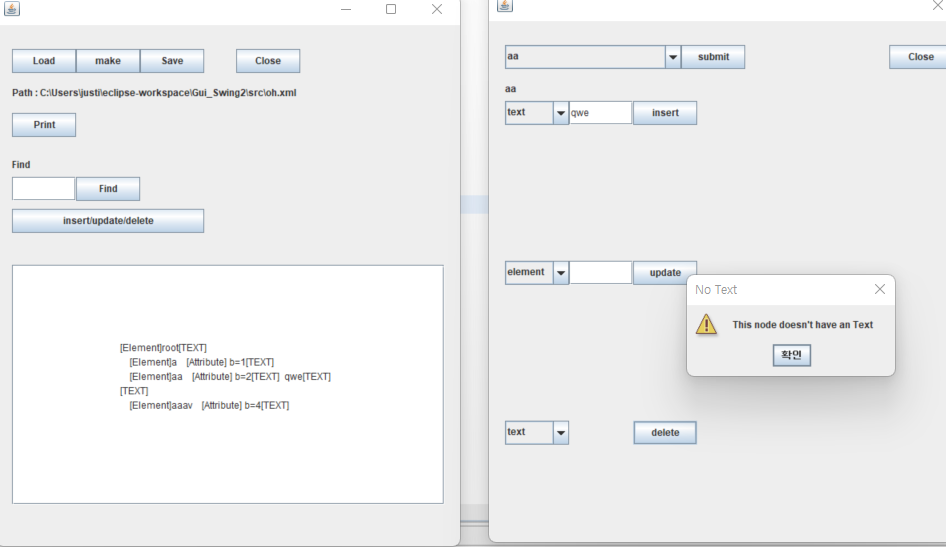
Abc element를 delete 한 경우입니다.



Attribute의 경우 element에 여러 개 있을 수 있으므로 이름을 입력 받습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명다음은 ATTT 속성이 사라진 것을 볼 수 있습니다.



Text를 포함해 attribute, comment가 선택된 node에 없으면 없다는 창이 뜹니다. 단 text와 comment는 node의 자식 중 알파벳 숫자로 구성된 text 노드만을 고릅니다.