

Projekt TKOM  
Jakub Strawa 300266

Temat: Łączenie plików emx zawierających diagram klas wykonany w programie Rational Software Architect Designer w jeden spójny diagram.

1. Zakres projektu

Projekt obejmuje scalanie 2 plików z rozszerzeniem emx w 1 spójny plik. Użytkownik wybiera dwa pliki oraz sposób ich scalania. Można utworzyć nowy plik lub wynikiem działania programu podmienić wybrany plik emx. Użytkownik może zdecydować czy chce rozwiązywać problemy w trakcie działania programu czy woli aby program utworzył plik tekstowy opisujący wszystkie kolizje oraz napotkane błędy. Program skupia się na scalaniu modelu. Warstwa graficzna diagramu klas, jeśli istnieje, jest kopiowana bez zmian w celu uniknięcia potencjalnych konfliktów z programem RSAD. Z tego powodu opis gramatyki nie uwzględnia szczegółów dotyczących sposobu opisywania warstwy graficznej w plikach emx.

2. Sposób realizacji

Aplikacje: lekser, parser oraz interpreter łączący pliki emx zostały napisane w pythonie. Diagramy klas zapisane w plikach emx są tworzone ręcznie przez użytkownika w programie RSAD.

Użytkownik, przy pomocy aplikacji okienkowej wykonanej z użyciem biblioteki tkinter, wybiera 2 pliki, które mają zostać scalone oraz ustawia dodatkowe parametry programu. Następnie po uruchomieniu programu tworzony jest interpreter, który tokenizuje (przy pomocy leksera) i parsuje (przy pomocy parsera) oba pliki wejściowe, a następnie próbuje je scalić oraz tworzy zapis swojego działania. Jeśli zaznaczona została przez użytkownika chęć ręcznego rozwiązywania konfliktów, program pytał będzie użytkownika co powinien w danej sytuacji zrobić. Scalanie plików odbywa się na podstawie porównania obu drzew AST, które utworzone zostały przez parser.

3. Wymagania funkcjonalne

- Program tworzy 1 spójny pliku emx z 2 plików.
- Program rozróżnia klasy, atrybuty i metody.
- Program zapamiętuje stereotypy klas
- Program rozróżnia 5 typów relacji pomiędzy klasami
- Program rozróżnia 4 typy parametrów metod: in, out, return oraz inout.
- Program rozróżnia fragmenty odpowiedzialne za model i grafikę.

4. Wymaganie niefunkcjonalne

- Program tworzy plik emx, który jest bez błędów wczytywany przez RSAD.
- Program jest w stanie poradzić sobie z sytuacjami konfliktowymi/błędami niekrytycznymi.

5. Zakładane funkcjonalności

We wszystkich poniższych przykładach xmi:id reprezentuje unikalne id konkretnego elementu modelu.

5.1 Dopuszczalne konstrukcje językowe z przykładami

- Klasa  
Przykładowa klasa Auto:  

```
<packagedElement xmi:type="uml:Class"
xmi:id="_uk1mEJCaEeuG94qDkPAb9w" name="Auto">
</packagedElement>
```
- Stereotyp  
Przykładowe stereotypy pojazd i koła:  

```
<eAnnotations xmi:id="_bmHsIJrIEeuG7NKZ46hTYw"
source="http://www.eclipse.org/uml2/2.0.0/UML">
  <details xmi:id="_d1H6UJrIEeuG7NKZ46hTYw" key="pojazd"/>
  <details xmi:id="_d1hYJrIEeuG7NKZ46hTYw" key="koła"/>
</eAnnotations>
```
- Atrybut  
Przykładowy publiczny atrybut „nowy” z wielokrotnością 1 oraz domyślną wartością 10:  

```
<ownedAttribute xmi:id="_0_xpMJCaEeuG94qDkPAb9w" name="nowy"
visibility="public">
  <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/JavaPrimitiveTypes.library.uml#int"/>
  <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_P6o3sJCcEeu5Cc4IUnM0CA" value="1"/>
  <lowerValue xmi:type="uml:LiteralInteger"
xmi:id="_P6oQoJCcEeu5Cc4IUnM0CA" value="1"/>
  <defaultValue xmi:type="uml:LiteralInteger"
xmi:id="_97w3UJCbEeu5Cc4IUnM0CA" value="10"/>
</ownedAttribute>
```
- Metoda  
Przykładowa metoda edytuj:  

```
<ownedOperation xmi:id="_BL8OoJCcEeu5Cc4IUnM0CA"
name="edytuj"/>
```
- Parametr metody  
Przykładowy parametr osoba o wielokrotności 1:  

```
<ownedParameter xmi:id="_KYHqUJCcEeu5Cc4IUnM0CA" name="osoba"
type="_QtWKMJCgEeuacZWeASWGOQ">
  <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_NemTEJCcEeu5Cc4IUnM0CA" value="1"/>
  <lowerValue xmi:type="uml:LiteralInteger"
xmi:id="_NelE8JCcEeu5Cc4IUnM0CA" value="1"/>
</ownedParameter>
```
- Asocjacja

```
<ownedAttribute xmi:id="_IARdoJCmEeuacZWeASWGOQ" name="osoba"
visibility="private" type="_QtWKMJCgEeuacZWeASWGOQ"
association="_IAPBYJCmEeuacZWeASWGOQ"/>
```

- Generalizacja  

```
<generalization xmi:id="_x0RyAJCkEeuacZWeASWGOQ"
general="_jnVLIJCKEeuacZWeASWGOQ"/>
```
- Agregacja  

```
<ownedAttribute xmi:id="_W3llQJCoEeuICJsfN6k0VQ"
name="preferencja" visibility="private"
type="_JW2klJCoEeuICJsfN6k0VQ" aggregation="shared"
association="_W3k-MJCoEeuICJsfN6k0VQ"/>
```
- Kompozycja  

```
<ownedAttribute xmi:id="_IKfj0JCgEeuacZWeASWGOQ"
name="właściciel" visibility="private"
type="_QtWKMJCgEeuacZWeASWGOQ" aggregation="composite"
association="_IKbSYJCgEeuacZWeASWGOQ">
```

## 6. Składnia

### 6.1. Rozpoznawane tokeny:

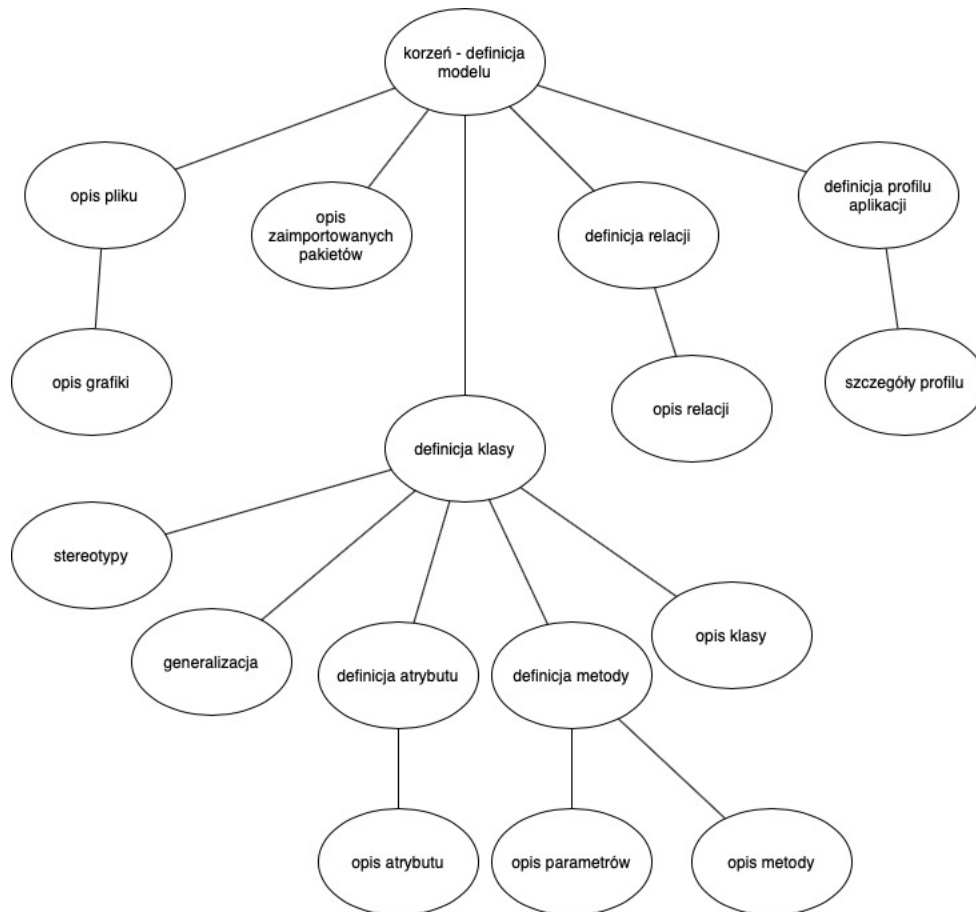
- <
- >
- /
- =
- uml:Model
- xmi:id
- name
- eAnnotations
- source
- contents
- xmi:type
- element
- xsi:nil
- packageImport
- importedPackage
- href
- packagedElement
- generalization
- general
- ownedAttribute
- visibility
- type
- upperValue

- value
- lowerValue
- defaultValue
- aggregation
- association
- ownedOperation
- ownedParameter
- isStatic
- memberEnd
- ownedEnd
- profileApplication
- references
- appliedProfile
- xmlns:notation
- children
- element
- target
- details
- key
- isLeaf
- isOrdered
- isReadOnly
- isDerived
- isDerivedUnion
- isQuery
- direction
- isUnique
- isAbstract
- EOF
- xmi:version
- xmlns:xmi
- xmlns:xsi
- xmlns:ecore
- xmlns:uml
- xmlns:umlnotation
- string value – odpowiada string value z 3.3
- double string value – odpowiada double string value z 3.3

## 6.2 Budowa drzewa

Drzewo jest zbudowane z głównych elementów pliku emx wyszczególnionych w podpunkcie 10. Drzewo skupia się głównie na opisie klas oraz wszystkich podległych jej elementów. Korzeniem drzewa jest definicja modelu UML, zawiera ona id oraz nazwę modelu, opis grafiki i plików, zaimportowane pakiety, elementy i profile. Definicje klas i relacji znajdują się w jednej liście, ponieważ ich kolejność w pliku może być dowolna. W klasie znajdziemy atrybuty odpowiadające: stereotypom, liście

generalizacji, liście atrybutów, liście operacji, widoczności klasy oraz jej parametrom. W definicji relacji znajdziemy id relacji, dwa id określające członków relacji oraz obiekt OwnedEnd, który szczegółowo opisuje typ relacji.



### 6.3 EBNF:

```

number = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0";
letter = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" |
"l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" |
"x" | "y" | "z" | "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |
"J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" |
"V" | "W" | "X" | "Y" | "Z" ;
symbol = number | letter | ":" | "/" | "." | "_" | "#" | "-" | "?";
boolean = "true" | "false";
string value = '' , ( symbol ) , { symbol } , '';
double string value = '' , ( symbol ) , { symbol } , " " , ( symbol ) , {
symbol } , '';

```

```

emx = emx header , model;
model = "<uml:Model" , model description , ">" , model contents ,
"</uml:Model>";

```

```

model description = 'xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" ,
['xmlns:notation="http://www.eclipse.org/gmf/runtime/1.0.2/notation"' ] ,

```

```

xmlns:uml="http://www.eclipse.org/uml2/3.0.0/UML"
xmlns:umlnotation="http://www.ibm.com/xtools/1.5.3/Umlnotation" xmi:id=',
string value, ' name=', string value;
model contents = file description,{ package import },{ packaged element
},{ profile application };

file description = "<eAnnotations xmi:id=", string value, '
source="uml2.diagrams"', ['references=', string value] ,'>', file name,
"</eAnnotations>";
file name = "<contents xmi:type='umlnotation:UMLDiagram' xmi:id=", string
value, ' type="Class" name=', string value, ">", graphic description,
"</contents>";
graphic description = children description, '<element xsi:nil="true"/>',
edges description;
children description = ...; //nie interesuje nas
edges description = ...; //nie interesuje nas

package import = "<packageImport xmi:id=", string value, ">", package,
"</packageImport>";
package = '<importedPackage xmi:type="uml:Model" href=', string value,
"/>";

packaged element = "<packagedElement xmi:type=", ( class | association ),
"</packagedElement>";
class = '"uml:Class" xmi:id=', string value, " name=", string value,
visibility, ['isLeaf="true"'], ['isAbstract="true"'], ">", [ stereotype ],
[ generalization ], {attribute}, {operation};
stereotype = "<eAnnotations xmi:id=", string value, " source=", string
value, ">", stereotype description, {stereotype description},
"</eAnnotations>";
stereotype description = "<details xmi:id=", string value, " key=", string
value, ">";
generalization = "<generalization xmi:id=", string value, " general=",
string value, ">";
attribute = "<ownedAttribute xmi:id=", string value, " name=", string
value, attribute parameters, ( ">" | attribute description );
attribute description = ">", [type], [limit], [default value],
"</ownedAttribute>";
attribute parameters = visibility, ['isLeaf="true"'], ['isStatic="true"'],
['isOrdered="true"'], ['isReadOnly="true"'], ['isDerived="true"'],
['isDerivedUnion="true"'], [short type], [association type];
visibility = " visibility=", visibility type;
visibility type = "public" | "private" | "protected" | "package";
short type = " type=", string value;

association type = [ 'aggregation="composite"' | 'aggregation="shared"' ],
"association=", string value;
type = "<type xmi:type=", string value, "href=", string value,"/>";
limit = upper limit, lower limit;
upper limit = "<upperValue xmi:type=", string value, " xmi:id=", string
value, [" value=", ("1" | "*")] ,"/>";
lower limit = "<lowerValue xmi:type=", string value, " xmi:id=", string
value, [" value=", ("1" | "*")] ,"/>";
default value = "<defaultValue xmi:type=", string value, " xmi:id=",

```

```

string value, " value=", string value, ( default value type | ">" );
default value type = type, "</defaultValue>";

operation = "<ownedOperation xmi:id=", string value, " name=", string
value, [ operation parameters ], (">" | parameter );
operation parameters = visibility, ['isLeaf="true"'], ['isStatic="true"'],
['isQuery="true"'];
parameter = owned parameter, {owned parameter}, "</ownedOperation>";
owned parameter = "<ownedParameter xmi:id=", string value, " name=",
string value, [owned parameter parameters], (">" | owned parameter
description);
owned parameter parameters = short type, ['isOrdered="true"'],
['isUnique="false"'], [parameter direction];
parameter direction = " direction=", direction type;
direction type = "return" | "out" | "inout";
owned parameter description = ">", [type], [upper limit], [lower limit],
[default value], "</ownedParameter>";

association = "uml:Association", " xmi:id=", string value, "
memberEnd=", double string value, ">", owned end;
owned end = "<ownedEnd xmi:id=", string value, " name=", string value,
visibility, short type, "association=", string value, ">", upper limit,
lower limit, "</ownedEnd>";

profile application = "<profileApplication xmi:id=", id, ">", eannotation,
applied profile, "</profileApplication>";
eannotation = "<eAnnotations xmi:id=", string value, " source=", string
value, ">", references, "</eAnnotations>";
references = '<references xmi:type="ecore:EPackage" href=', string value,
"/>';
applied profile = "<appliedProfile href=", path, "/>";

emx header =
'<?xml version="1.0" encoding="UTF-8"?>
<!--xtools2_universal_type_manager-->
<!--Rational Software Architect Designer 9.7.0-->
<?com.ibm.xtools.emf.core.signature <signature
id="com.ibm.xtools.mmi.ui.signatures.diagram" version="7.0.0"><feature
description="" name="Rational Modeling Platform (com.ibm.xtools.rmp)"
url="" version="7.0.0"/></signature>?>
<?com.ibm.xtools.emf.core.signature <signature
id="com.ibm.xtools.uml.msl.model" version="7.0.0"><feature description=""
name="com.ibm.xtools.ruml.feature" url=""
version="7.0.0"/></signature>?>';

```

## 7. Obsługa błędów

Błąd składniowy w podanych plikach emx jest traktowany jako błąd krytyczny (np. pojawienie się niespodziewanego tokenu w opisie klasy), ponieważ zakładam, że użytkownik nie ingerował w wygenerowane przez RSAD pliki, więc taki błąd przerywa działanie programu.

Błędy leksykalne mogą być niekrytyczne tylko wewnątrz opisu warstwy graficznej, ponieważ jest ona kopiowana bezpośrednio jako lista tokenów. W pozostałych miejscach programu jest to błąd krytyczny, który kończy działanie programu. Konflikt atrybutów/metod/parametrów/relacji itp. pomiędzy klasami jest pozostawiony użytkownikowi do rozstrzygnięcia lub pozostaje w takiej formie w jakiej występuje w pliku głównym (plik numer 1), a informacja o takim zdarzeniu jest zapisywana do pliku tekstowego.

#### 8. Sposób testowania

Testowanie interpretera odbywa się na zasadzie wielu testów jednostkowych skupiających się na różnych poziomach skomplikowania. Zaczynając od 2 prawie identycznych prostych plików, następnie takich różniących się rosnącą liczbą atrybutów i metod poprzez pliki różniące się praktycznie wszystkimi klasami i relacjami. Każdy kolejny test skupia się na sprawdzeniu działania programu z większą ilością konfliktów.

Testowanie leksera polega na porównaniu wyników działania (listy tokenów) dla kilku przykładowych fragmentów prawdziwego pliku emx.

Testowanie parsera polega na porównaniu drzew wykonanych dla prostych przykładowych plików.

#### 9. Krótki opis budowy pliku emx

- Opis środowiska, kodowanie, wersja programu
- Rozpoczęcie opisu modelu
- Opis pliku
  - Opis grafiki modelu
- Opis zaimportowanych pakietów typów danych
- Opis klas
  - Stereotypy
  - Generalizacja
  - Atrybuty
    - Typ oraz rodzaj relacji
    - Górny limit
    - Dolny limit
    - Wartość domyślna
  - Metody
    - Parametry
      - Górny limit
      - Dolny limit
- Opis relacji pomiędzy klasami
- Opis profili aplikacji
- Zakończenie opisu modelu

#### 10. Wartości domyślne w pliku emx

Pliki emx często nie pokazują wszystkich wartości atrybutów. Dzieje się tak dlatego, że wartości domyślne dla parametrów np. klas czy metod nie są zapisywane przez RSAD. W trakcie realizacji projektu zauważyłem, że jawne przypisanie domyślnych



wartości parametrów jest równoznaczne z ich nieprzypisaniem, dlatego często wyjściowe pliki różnią się w nieznaczny sposób od tych wejściowych, ponieważ występuje jawne przypisanie tych wartości. Rozwiązaniem tego problemu jest wczytanie pliku do RSAD i np. niewielka zmiana warstwy graficznej (przykładowo przesunięcie klasy). Sprawia to, że po zapisaniu plik jest zaktualizowany i wszystkie redundantne parametry zostają usunięte. Poniżej znajdują się domyślne wartości parametrów wykorzystywanych w tym projekcie.

Class:

- visibility="public"
- isAbstract="false"
- isLeaf="false"

Attribute:

- isUnique="true", reszta na "false"
- wszystkie visibility są jawnie napisane
- aggregation=None
- multiplicity limits: (upper/lower) możliwości: \*/None, 1/None, 1/1, \*/1 None reprezentuje 0

Operation:

- visibility "public"
- isUnique="true", reszta "false"

Operation attribute:

- direction="in"
- visibility="public"
- isOrdered="false", isUnique="true"
- multiplicity limits: (upper/lower) możliwości: \*/None, 1/None, 1/1, \*/1 None reprezentuje 0

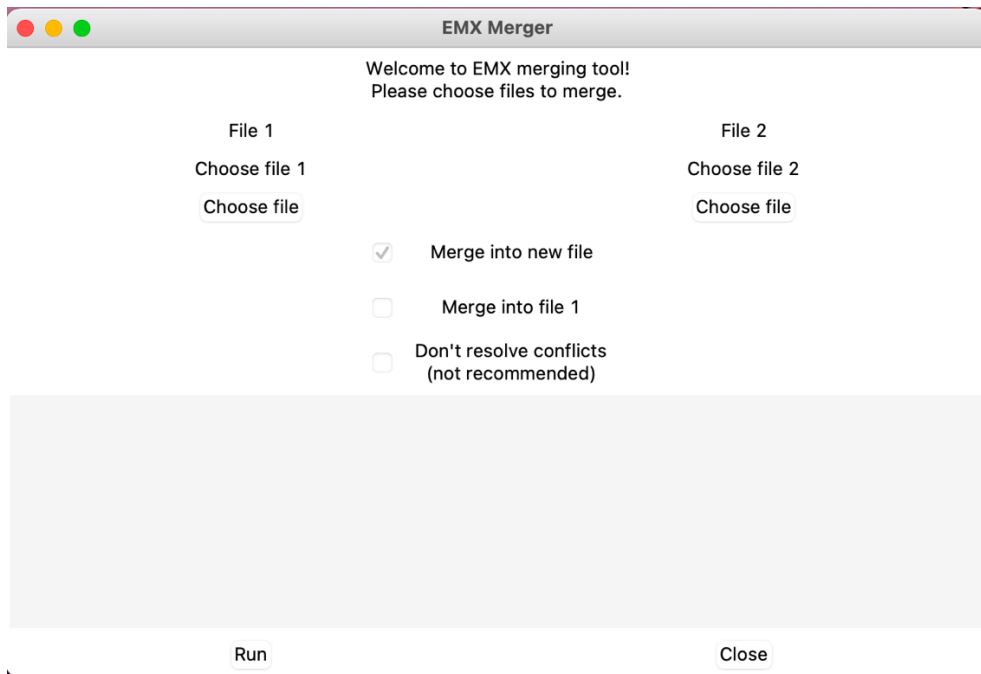
## 11. Sposób uruchomienia

Do uruchomienia aplikacji potrzebujemy:

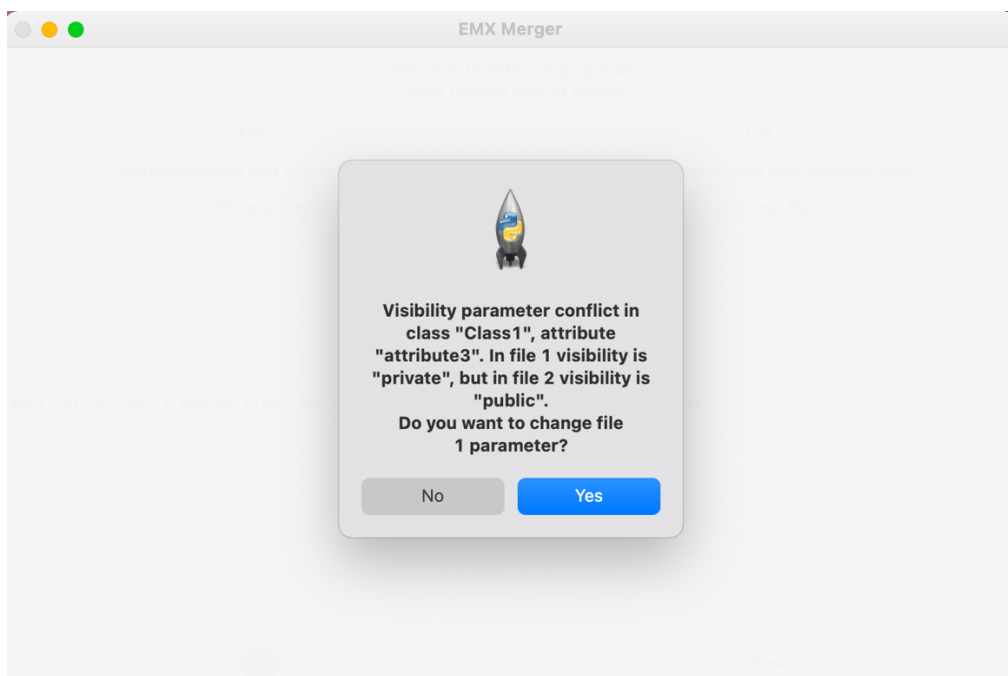
- pythona 3.x
- biblioteki tkinter (***pip install tk***)
- Rational Software Architect Designer (RSAD) – potrzebny do utworzenia plików emx

Program uruchamiamy poleceniem: ***python main.py***

W tym momencie powinno pojawić się okno aplikacji:



Wybieramy 2 pliki, ustawiamy interesujące nas opcje i klikamy przycisk: **Run**, co spowoduje uruchomienia programu. Jeśli zaznaczyliśmy opcję ręcznego rozwiązywania konfliktów, to możemy spodziewać się komunikatów o konfliktach takich jak na poniższym zdjęciu.



Po skończeniu działania logi wypisane w aplikacji oraz zapisane w folderze *output\_files* razem z wynikiem działania programu, jeśli zaznaczyliśmy opcję scalenia plików do nowego pliku. W przeciwnym razie wynik działania programu zastąpi plik numer 1. Poniżej znajduje się wynik działania przykładowego programu.

EMX Merger

Welcome to EMX merging tool!  
Please choose files to merge.

File 1

...generalizations and association.emx

Choose file

File 2

... and association with conflicts.emx

Choose file

☒ Merge into new file

☐ Merge into file 1

☐ Don't resolve conflicts  
(not recommended)

IsOrdered parameter conflict in class "Class1", operation "Operation1", attribute "Parameter1".

IsOrdered parameter conflict resolved in class "Class1", operation "Operation1", attribute "Parameter1", isQuer

y changed to "false".

Upper limit value conflict in class "Class1", operation "Operation1", attribute "Parameter1".

Upper limit value conflict resolved in class "Class1", operation "Operation1", attribute "Parameter1", value ch

anged to "\*".

Direction parameter conflict in class "Class1", operation "Operation1", attribute "Parameter4".

Direction parameter conflict resolved in class "Class1", operation "Operation1", attribute "Parameter4", direct

ion changed to "in".

Appended new generalization to class: "Class1", referencing class: "Class2".

Visibility parameter conflict in class "Class2", attribute "class3".

Visibility parameter conflict resolved in class "Class2", attribute "class3", visibility changed to "package".

Application result was written into new file: output\_files/new\_emx\_merged.emx.

Run

Close