


Programowanie w JAVA

Lab. 3 – Swing

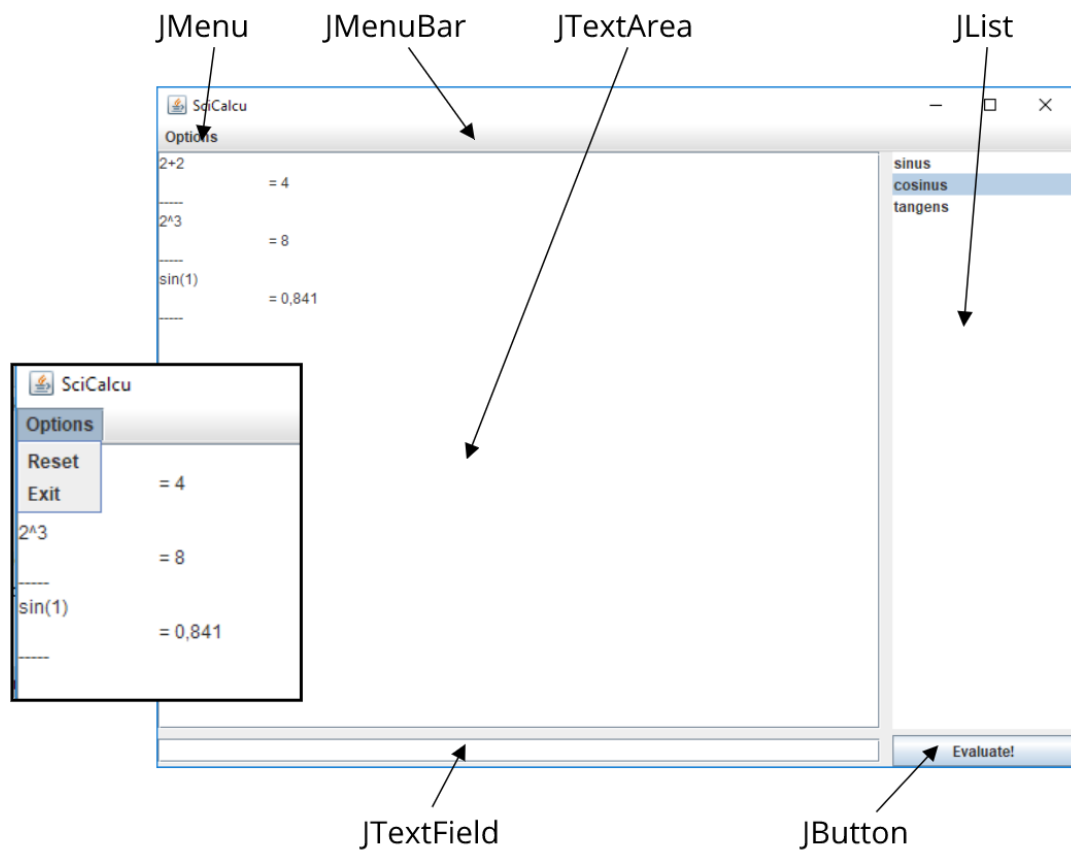
1. **Cel zadania:** poczuj się jak pracownik NASA i zaimplementuj kalkulator naukowy w oparciu o bibliotekę mXparser!

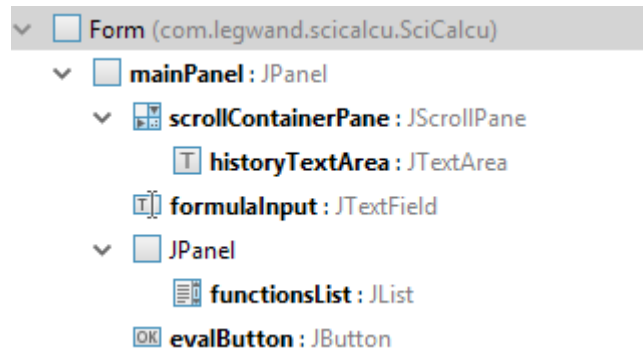
 **nasa/CCDD** – CcddMacroHandler.java Java

Showing the top four matches Last indexed 3 days ago

```
43 import javax.swing.text.JTextComponent;
44
45 import org.mariuszgromada.math.mxparser.Expression;
46
47 import CCDD.CcddClasses.PaddedComboBox;
```

2. Twoim zadaniem jest stworzenie kalkulatora z zaawansowanym interfejsem graficznym:





3. Dodawanie biblioteki mXparser do projektu:

- Stwórz projekt oparty o maven
- Do pliku pom.xml pod tagiem <version> dodaj:

```

<dependencies>
  <dependency>
    <groupId>org.mariuszgromada.math</groupId>
    <artifactId>MathParser.org-mXparser</artifactId>
    <version>4.1.1</version>
  </dependency>
</dependencies>

```

4. Opis poszczególnych funkcjonalności:

- Reset (menu):** Czyści zawartość JTextField i JTextArea
- Exit (menu):** Zamyka aplikację
- JTextArea:** Pole tekstowe, w którym wyświetlana jest historia operacji w formacie podanym na ilustracji.
 - Do wyświetlenia pojedynczego wpisu zaprojektuj szablon i wykorzystaj klasę MessageFormat
 - komponent JTextArea powinien być read-only
 - jeśli będzie taka potrzeba powinien pojawić się suwak do przesuwania historii
- JTextField:** Służy do wpisywania działań matematycznych:
 - Jeśli użytkownik wciśnie klawisz ↑ (na klawiaturze) to w polu tekstowy powinno się pojawić ostatnie wpisane działanie
- JList:** Zawiera listę podstawowych funkcji matematycznych. Zaimplementuj obsługę 3 funkcji jednoargumentowych (<http://mathparser.org/mxparser-math-collection/unary-functions/>), 3 funkcji dwuargumentowych (<http://mathparser.org/mxparser-math-collection/binary-functions/>) i wyświetlanie 3 wybranych stałych (<http://mathparser.org/mxparser-math-collection/constants/>) + opcja „last result”
 - List powinna zostać zbudowana przy użyciu DefaultListModel (<http://www.codejava.net/java-se/swing/jlist-custom-renderer-example>), który będzie zawierał pełne nazwy funkcji i ich odpowiedniki akceptowalne dla parsera
 - Lista musi wyświetlać pełne nazwy funkcji

- iii. Po podwójnym kliknięciu wybrana funkcja powinna się pojawić w polu JTextField w formie akceptowalnej dla parsera (np. po kliknięciu sinus w polu tekstowym powinno pojawić sin())
 - iv. Jeśli funkcja zawiera nawiasy, kursor powinien automatycznie ustawić się między nawiasami
 - v. Funkcja last result powinna wklejać do pola JTextField ostatni wynik
 - vi. Po wciśnięciu klawisza Enter działanie powinno zniknąć z JTextField i pojawiać się razem z wynikiem w historii
 - vii. Jeśli składnia działania jest niepoprawna powinien się pojawić popup z opisem błędu
- f. **Evaluate:** działa analogicznie jak wciśnięcie Enter w JTextField

Korzystanie z biblioteki mXparser (<http://mathparser.org/>)

```
Expression expression = new Expression("2+2");

if (expression.checkSyntax()) {
    Double result = expression.calculate();
}
else {
    String errorMessage = expression.getErrorMessage()
    //należy rzucić wyjątek
}
```

Wyświetlanie okienka popup

```
JOptionPane.showMessageDialog(null, "Message", "Title",
JOptionPane.ERROR_MESSAGE);
```

Korzystanie z MessageFormat

```
String result = MessageFormat.format(
    "At {1,time} on {1,date}, there was {2} on planet
{0,number,integer}.",
    planet, new Date(), event);
```

<https://docs.oracle.com/javase/7/docs/api/java/text/MessageFormat.html>

Po uzyskaniu zaliczenia, prześlij źródła w archiwum **zgodnie z konwencją nazewnictwa** do chmury na adres: <https://cloud.kisim.eu.org/index.php/s/Nfbpg3HGZfdPPeK>