Konwerter plików w bibliotece Swing

z poradnikiem narzędzia Mockito do testów jednostkowych

w ramach kursu komputerowego w języku Java

Michał Błaszczyk

Uniwersytet Mikołaja Kopernika w Toruniu

29.01.2025

Wstęp

W dobie cyfryzacji i rosnącej ilości danych, umiejętność efektywnego zarządzania plikami staje się kluczowa. W kontekście kursu komputerowego w języku Java, stworzenie konwertera plików przy użyciu biblioteki Swing nie tylko rozwija umiejętności programistyczne, ale także wprowadza nas w świat tworzenia aplikacji graficznych. Dodatkowo, wprowadzenie narzędzia Mockito do testów jednostkowych pozwala na zrozumienie znaczenia testowania w procesie tworzenia oprogramowania.

Problematyka konwertera plików

Złożoność formatu plików: Różnorodność formatów plików (np. TXT, CSV, DOCX,PDF, JSON) oraz ich specyfika mogą stanowić wyzwanie przy implementacji konwertera. Każdy format wymaga innego podejścia do odczytu i zapisu danych.

1. **Interfejs użytkownika**: Stworzenie intuicyjnego i przyjaznego interfejsu użytkownika w bibliotece Swing jest kluczowe dla zapewnienia pozytywnego doświadczenia użytkownika. Problemy mogą pojawić się w kontekście responsywności oraz estetyki aplikacji.
2. **Obsługa błędów**: W trakcie konwersji plików mogą wystąpić różne błędy, takie jak błędny format pliku, brak uprawnień do odczytu/zapisu, czy problemy z pamięcią. Odpowiednia obsługa tych błędów jest niezbędna dla stabilności aplikacji.
3. **Wydajność**: Przy dużych plikach lub dużej liczbie plików, wydajność konwertera staje się kluczowym zagadnieniem. Należy rozważyć optymalizację algorytmów konwersji oraz zarządzanie pamięcią.

Problematyka testowania jednostkowego z użyciem Mockito

1. **Złożoność interakcji między komponentami**

   W aplikacjach z interfejsem graficznym, takich jak konwerter plików, często występują złożone interakcje między różnymi komponentami (np. GUI, logika

konwersji, zarządzanie plikami). Testowanie tych interakcji bez odpowiednich narzędzi może być trudne i czasochłonne.

2. Izolacja testów

   Aby testy były wiarygodne, muszą być izolowane od zewnętrznych zależności, takich jak system plików czy bazy danych. Mockito umożliwia tworzenie atrap (mocków) dla tych zależności, co pozwala na testowanie logiki aplikacji w oderwaniu od rzeczywistych komponentów.

3. Symulacja zachowań

   W przypadku konwertera plików, mogą wystąpić różne scenariusze, które należy przetestować, takie jak poprawna konwersja plików, obsługa błędów (np. brak pliku, błędny format) oraz różne stany aplikacji. Mockito pozwala na łatwe symulowanie tych zachowań, co ułatwia tworzenie testów.

4. Weryfikacja interakcji

   Testy jednostkowe powinny nie tylko sprawdzać, czy wyniki są poprawne, ale także weryfikować, czy odpowiednie metody zostały wywołane w odpowiednich momentach. Mockito umożliwia weryfikację interakcji między obiektami, co jest istotne w kontekście testowania logiki konwersji plików.

Raport dotyczący konwertera plików w bibliotece Swing oraz testowania jednostkowego z użyciem Mockito ma na celu nie tylko przedstawienie technicznych aspektów implementacji, ale także podkreślenie znaczenia dobrych praktyk programistycznych. Pozwoli to zdobyć praktyczne umiejętności, które są niezbędne w dzisiejszym świecie programowania, a także zrozumieć, jak ważne jest testowanie w procesie tworzenia oprogramowania.

Technologie użyte do zrealizowania projektu

1. **Java**

   - **Opis**: Java to obiektowy język programowania, który jest szeroko stosowany do tworzenia aplikacji desktopowych, webowych oraz mobilnych. Jego platforma niezależna oraz bogaty zestaw bibliotek czynią go idealnym wyborem do budowy aplikacji.
   - **Zastosowanie w projekcie**: w projekcie konwertera plików wykorzystano język Java do implementacji logiki aplikacji oraz interfejsu użytkownika.

2. **Swing**

   - **Opis**: Swing to biblioteka GUI w Javie, która umożliwia tworzenie graficznych interfejsów użytkownika. Oferuje zestaw komponentów, takich jak przyciski, pola tekstowe, listy rozwijane, które można łatwo dostosować.
   - **Zastosowanie w projekcie**: w projekcie wykorzystano Swing do stworzenia interfejsu użytkownika konwertera plików. Klasa *ConverterGuiSwing* odpowiada za zarządzanie komponentami GUI oraz interakcjami użytkownika.

3. **Apache POI**

   - **Opis**: Apache POI to biblioteka Java, która umożliwia odczyt i zapis plików Microsoft Office, w tym plików DOCX i XLSX. Jest to przydatne narzędzie do pracy z dokumentami biurowymi.
   - **Zastosowanie w projekcie**: w projekcie konwertera plików użyto Apache POI do odczytu zawartości plików DOCX w metodzie *convert()* w klasie *ConvertionLogicClass*.

4. **iText (Lowagie)**

- **Opis**: iText to biblioteka do tworzenia i manipulacji dokumentami PDF w Javie. Umożliwia generowanie PDF-ów z różnych źródeł danych.
- **Zastosowanie w projekcie**: w projekcie wykorzystano iText (Lowagie) do generowania plików PDF z zawartości plików DOCX w metodzie *convertDocxToPdf()* w klasie *ConvertionLogicClass*.

5. **Mockito**

- **Opis**: Mockito to framework do tworzenia atrap (mocków) w testach jednostkowych w Javie. Umożliwia symulowanie zachowań obiektów oraz weryfikację interakcji między nimi.
- **Zastosowanie w projekcie**: w projekcie zastosowano Mockito do testowania jednostkowego komponentów konwertera plików. Przykłady testów znajdują się w klasach *ConverterGuiSwingMockitoTest* oraz *ConvertionLogicMockitoTest*, gdzie mockowano klasy i weryfikowano interakcje.

6. **JUnit**

Opis: JUnit to framework do testowania jednostkowego/integracyjnego w Javie, który umożliwia pisanie i uruchamianie testów.

Zastosowanie w projekcie: w projekcie użyto JUnit do organizacji testów integracyjnych i jednostkowych. Miał na celu pokazać słabe jak i mocne strony Mockito.

Struktura projektu

1. Logika konwersji

Plik C*onversionLogicClass* implementuje główną funkcjonalność konwersji plików DOCX na PDF:

- Metoda *convertDocxToPdf()* otwiera plik wejściowy i zapisuje wynik w podanej lokalizacji.
- Metoda *convert()* obsługuje właściwą konwersję, wykorzystując:

- **Apache POI** do odczytu DOCX.
- **iText** do generowania pliku PDF.

2. Obsługa plików

   Plik *FileToConvertClass.java* zarządza plikami wejściowymi i wyjściowymi:

   - Przechowuje ścieżki plików
   - Tworzy katalog *./converted_files/* dla wyników.
   - Metoda *convertAndSave(File file)* zapewnia, że plik zostanie zapisany.

   Plik *TypeFileClass.java* definiuje obsługiwane typy plików jako enum: *PDF, DOCX, TXT, CSV*

3. Interfejs użytkownika

   Plik ConverterGuiSwing.java implementuje GUI w Swing:

   - Obsługuje przyciski:
     - addFileFunctionButton – dodawanie pliku
     - removeFileFunctionButton – usuwanie pliku
     - convertFunctionButton – uruchamianie konwersji
   - JcomboBox pozwala użytkownikowi wybrać format pliku
   - Przechowuje referencję do klasy FileToConvertClass i integruje się z logiką konwersji

4. Testy jednostkowe

   - **Testy GUI:**
     - *ConverterGuiSwingIntegrationTest.java* sprawdza interakcje w interfejsie, np. czy przycisk konwersji aktywuje się po wpisaniu nazwy pliku.
     - *ConverterGuiSwingMockitoTest.java* używa **Mockito**, by testować interakcje GUI z mockowaną logiką konwersji.

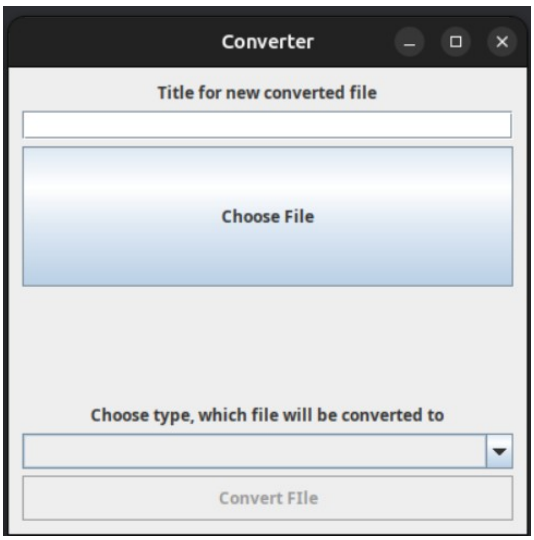
- **Testy logiki konwersji:**
  - *ConvertionLogicMockitoTest.java* testuje *convertDocxToPdf()*, tworząc tymczasowe pliki i sprawdzając, czy wynikowy *PDF* istnieje.
  - *ConvertionLogicTestClass.java* weryfikuje, czy plik *PDF* zostaje poprawnie zapisany.
- **Testy narzędziowe:**
  - *TestUtils.java* pomaga w debugowaniu GUI, np. wyszukuje komponenty po nazwie.

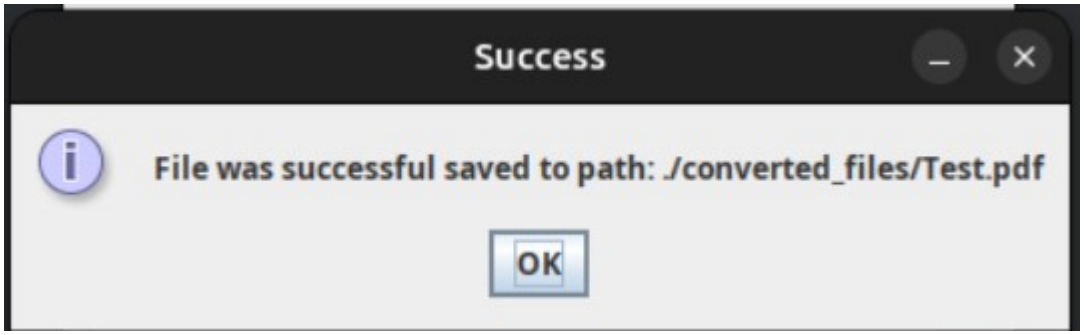
Zwięzłe przedstawienie i rezultaty działania aplikacji konwertera plików

Aplikacja umożliwia konwersję plików DOCX do PDF poprzez prosty interfejs graficzny (Swing). Użytkownik wybiera plik, podaje nazwę wyjściową i określa format docelowy. Po kliknięciu przycisku "Convert", aplikacja:

1) **Odczytuje plik DOCX** za pomocą biblioteki **Apache POI**.
2) **Konwertuje go do PDF**, wykorzystując **iText**.
3) **Zapisuje wynik w katalogu** ./converted_files/.

System sprawdza poprawność wejścia i informuje użytkownika o sukcesie lub błędach. Testy jednostkowe i integracyjne zapewniają stabilność działania.





Testy jednostkowe (Mockito i JUnit) i testy integracyjne (Junit)

1. **Testy jednostkowe sprawdzają pojedyńcze komponenty (klasy, metody) w izolacji, bez zależności od innych części systemu.**

   Przykład z projektu:

   - *ConverterGuiSwingMockitoTest.java*
     - **Mockuje** klasę FIleToConvertClass, aby sprawdzić, czy GUI prawidłowo ustawia nazwę pliku i format.
     - **Nie wykonuje rzeczywistej konwersji**, testuje tylko interakcje między komponentami.
   - *Przykład z projektu:ConvertionLogicMockitoTest.java*
     - Testuje metodę convertDocxToPdf(), ale **nie zależy od rzeczywistych plików** – używa **Mockito** do symulowania operacji.

   Charakterystyka testów jednostkowych w Mockito:

   - Testują pojedyńcze funkcje/metody
   - Są szybkie, bo działają w izolacji
   - Używają mocków do eliminacji zależności

2. **Testy integracyjne sprawdzają współdziałanie wielu komponentów w rzeczywistych warunkach.**

   Przykład z projektu:

   - ConverterGuiSwingIntegrationTest.java
     - Testuje całe GUI w rzeczywistym środowisku

- Symuluje działania użytkownika: wpisuje nazwę pliku, wybiera format i sprawdza, czy przycisk konwersji działa.

- ConvertionLogicTestClass.java
  - Sprawdza, czy konwersja DOCX → PDF faktycznie działa.
  - Tworzy rzeczywisty plik DOCX, przekazuje go do metody convertDocxToPdf() i weryfikuje, czy PDF został zapisany.

Charakterystyka testów integracyjnych:

- Sprawdzają współprace kilku modułów
- Nie ilozują komponentów – używają prawdziwych plików, interfejsu itp.
- Są wolniejsze, bo wymagają pełnej konfiguracji systemu.

Podsumowanie testów – kluczowe różnice

Cecha	Test jednostkowy	Test integracyjny
Cel	Sprawdzanie pojedyńczej metody lub klasy	Sprawdzenie interakcji między modułami
Środowisko	Działa w izolacji	Korzysta z rzeczywistych zasobów
Przykład w projekcie	ConverterGuiSwingMockitoTest	ConverterGuiSwingIntegrationTest
Szybkość	Szybkie	Wolniejsze
Użycie mocków	Tak (Mockito)	Nie
Zastosowanie	Weryfikacja logiki metod	Testowanie działania całej aplikacji

Jak dołączyć Swing i Mockito do projektu

1. **Dołączenie biblioteki Swing:**

   Swing jest częścią standardowej biblioteki Javy, więc nie wymaga dodatkowego pobierania. Aby użyć w swoim projekcie, wystarczy zaimportować odpowiednie klasy w kodzie. Zachęcam również do pobrania wtyczki Swing UI Designer (pomaga w tworzeniu interface'u użytkownika)

2. **Dołączenie biblioteki Mockito**

   Pobranie Mockito: w przypadku korzystania z systemu zarządzania zależnościami, takiego jak Maven należy dodać przykładowy fragment do pliku pom.xml

```xml
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>5.15.2</version>
</dependency>
```

   Importowanie Mockito: należy w kodzie testowym zaimportować odpowiednie klasy Mockito

3. **Tworzenie testów z Mockito – tworzenie przykładowych mock'ów i ich stubowanie**

```java
// Tworzenie mocka dla FIleToConvertClass
FIleToConvertClass mockedFileToConvertClass =
        mock(FIleToConvertClass.class);

when(mockedFileToConvertClass.
        getFilename()).
        thenReturn( t: "MockedFile");
```

Wnioski

Znaczenie umiejętności programistycznych

Projekt konwertera plików w bibliotece Swing podkreśla znaczenie umiejętności programistycznych w kontekście rosnącej cyfryzacji.

Wyzwania związane z konwersją plików

Złożoność różnych formatów plików oraz potrzeba efektywnej obsługi błędów i wydajności stanowią istotne wyzwania. Projekt pokazał, jak ważne jest zrozumienie specyfiki każdego formatu oraz odpowiednie zarządzanie pamięcią i zasobami.

Rola testowania w procesie tworzenia oprogramowania

Wprowadzenie narzędzi takich jak Mockito i JUnit do testowania jednostkowego i integracyjnego podkreśla znaczenie testowania w zapewnieniu stabilności i jakości aplikacji. Testy jednostkowe pozwoliły na izolację komponentów, co przyczyniło się do szybszego wykrywania błędów.

Praktyczne umiejętności w testowaniu:

Miałem okazję nauczyć się, jak tworzyć mocki i symulować zachowania obiektów, co jest kluczowe w testowaniu aplikacji z interfejsem graficznym. Umiejętność ta jest nie tylko przydatna w kontekście tego projektu, ale również w przyszłych przedsięwzięciach programistycznych.

Zastosowanie nowoczesnych technologii:

Wykorzystanie bibliotek takich jak Apache POI i iText w projekcie pokazuje, jak nowoczesne technologie mogą ułatwić pracę z różnymi formatami plików.

Podsumowanie

Projekt konwertera plików nie tylko dostarczył mi praktycznych umiejętności programistycznych, ale także uwypuklił znaczenie testowania i dobrych praktyk w tworzeniu oprogramowania.

NA ZAKOŃCZENIE ZACHĘCAM DO DOKŁADNEJ WERYFIKACJI KODU PROJEKTU, ORAZ SUMIENNEGO PRZECZYTANIA KOMENTARZY I URUCHOMIENIU POSZCZEGÓLNYCH TESTÓW. POZWOLI TO LEPIEJ ZROZUMIEĆ IDEE PROJEKTU.

Referencje do źródeł zewnętrznych

Mockito:

https://site.mockito.org/

https://www.javatpoint.com/methods-of-mockito

https://dzone.com/refcardz/mockito

Swing:

https://www.jetbrains.com/help/idea/design-gui-using-swing.html

https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html

https://www.javatpoint.com/java-swing

Apache POI i iText

https://poi.apache.org/

https://itextpdf.com/

Inne referencje godne polecenia:

https://wasp-lang.dev/blog/2023/10/12/on-importance-of-naming-in-programming

https://habr.com/en/articles/567870/