Projekt kalkulatora figur płaskich – sprawozdanie z przedmiotu „Zastosowanie programowania komponentowego”

Sporządził

Michał Postek

U-14847

**Spis treści**

[Cel projektu 3](#__RefHeading___Toc1277_2295379900)

[Wykorzystane technologie 4](#__RefHeading___Toc1279_2295379900)

[Wymagania funkcjonalne 5](#__RefHeading___Toc1281_2295379900)

[Obliczanie pola figury 6](#__RefHeading___Toc1296_2295379900)

[Obliczanie obwodu figury 7](#__RefHeading___Toc1298_2295379900)

[Zapis danych do pliku 8](#__RefHeading___Toc1300_2295379900)

[Odczyt danych z pliku 9](#__RefHeading___Toc1302_2295379900)

[Wymagania niefunkcjonalne 10](#__RefHeading___Toc1283_2295379900)

[Prototypy komponentów 11](#__RefHeading___Toc1285_2295379900)

[Aplikacja testująca 12](#__RefHeading___Toc1287_2295379900)

# Cel projektu

Celem projektu jest stworzenie kalkulatora figur płaskich wykorzystując technologię Java Beans. Projekt ma pomóc w poznaniu owej technologii i pomóc w jej praktycznym wykorzystaniu.

# Wykorzystane technologie

W niniejszym projekcie wykorzystano następujące technologie oraz narzędzia:

* Język programowania Java w tym biblioteki Swing oraz AWT
* Technologię komponentową Java Beans
* Zintegrowane środowisko programistyczne Intellij Idea
* System kontroli wersji git
* Program do tworzenia prototypów interfejsu GUI Pencil

# Wymagania funkcjonalne

Wymagania funkcjonalne dotyczące poszczególnych komponentów oraz aplikacji testującej omówionej w dalszej części sprawozdania zostały przedstawione za pomocą formularzy na dalszych stronach.

## Obliczanie pola figury

**Nazwa funkcji:** Oblicz pole

**Opis:** Oblicza pole figury na podstawie danych podanych przez użytkownika, w wypadku komponentów, których wielkości nie zawsze są jednoznaczne, użytkownik ma możliwość wyboru wielkości, na podstawie której obliczany jest wynik.

**Dane wejściowe:** W zależności od komponentu, wielkość określająca parametry umożliwiające obliczanie pola (np. długość podstawy)

**Źródło danych wejściowych:** Wielkości pochodzą z pól tekstowych zawartych w interfejsie GUI poszczególnych komponentów

**Dane wyjściowe:** Pole figury

**Przeznaczenie:** Obliczenia. Wyniki obliczeń są wyświetlane na ekranie.

**Wymaga:** Podanie liczb, aby program mógł przeprowadzić obliczenia

**Efekty uboczne:** Nie ma

**Uwagi:** Wynik jest zwracany jako wartość zmiennoprzecinkowa, w celu zwiększenia dokładności obliczeń

## Obliczanie obwodu figury

**Nazwa funkcji:** Oblicz obwód

**Opis:** Oblicza obwód figury na podstawie danych podanych przez użytkownika, w wypadku komponentów, których wielkości nie zawsze są jednoznaczne, użytkownik ma możliwość wyboru wielkości, na podstawie której obliczany jest wynik.

**Dane wejściowe:** W zależności od komponentu, wielkość określająca parametry umożliwiające obliczanie obwodu (np. długość podstawy)

**Źródło danych wejściowych:** Wielkości pochodzą z pól tekstowych zawartych w interfejsie GUI poszczególnych komponentów

**Dane wyjściowe:** Obwód figury

**Przeznaczenie:** Obliczenia. Wyniki obliczeń są wyświetlane na ekranie.

**Wymaga:** Podanie liczb, aby program mógł przeprowadzić obliczenia.

**Efekty uboczne:** Nie ma

**Uwagi:** Wynik jest zwracany jako wartość zmiennoprzecinkowa, w celu zwiększenia dokładności obliczeń

## Zapis danych do pliku

**Nazwa funkcji:** Zapisz dane do pliku

**Opis:** Zapisuje dane wejściowe do pliku wskazanego przez użytkownika

**Dane wejściowe:** Wielkości określające daną figurę, ścieżka do pliku, w którym program ma zapisać dane.

**Źródło danych wejściowych:** Wielkości określające figury pobierane są na podstawie danych wprowadzonych w pola tekstowe przez użytkownika. Ścieżka do pliku docelowego może być wpisana do pola tekstowego, bądź wybrana za pomocą wyniku z okna JFileChooser.

**Dane wyjściowe:** Plik zawierający parametry wejściowe figury, umożliwiające odtworzenie wyników obliczeń.

**Przenaczenie:** Historia obliczeń.

**Wymaga:** Podstawą zapisu do pliku jest podanie ścieżki docelowej oraz parametrów umożliwiających zapis.

**Efekty uboczne:** Brak

**Uwagi:** W systemach Unixowych należy posiadać uprawnienia, aby zapisać plik w niektórych katalogach. Program zapisuje dane w formacie XML.

## Odczyt danych z pliku

**Nazwa funkcji:** Odczytaj dane z pliku

**Opis:** Odczytuje dane wejściowe z pliku wskazanego przez użytkownika, a następnie wypełnia pola z parametrami opisującymi figurę

**Dane wejściowe:** Ścieżka do pliku XML, z którego program ma odczytać dane.

**Źródło danych wejściowych:** Ścieżka do pliku docelowego może być wpisana do pola tekstowego, bądź wybrana za pomocą wyniku z okna JFileChooser.

**Dane wyjściowe:** Parametry wejściowe figur umożliwiające odtworzenie obliczeń.

**Przenaczenie:** Historia obliczeń. Przeprowadzanie obliczeń z pliku.

**Wymaga:** Podstawą odczytu z pliku jest podanie ścieżki docelowej.

**Efekty uboczne:** Brak

**Uwagi:** W systemach Unixowych należy posiadać uprawnienia, aby odczytać plik w niektórych katalogach. Plik wejściowy musi być zapisany w formacie obsługiwanym przez program.

# Wymagania niefunkcjonalne

* Interfejs każdego z komponentów powinien być czytelny i nieskomplikowany
* Aplikacja testowa powinna być dostępna na różnych systemach operacyjnych
* Aplikacja powinna mieć możliwość przechowywania danych obliczeń
* Każda operacja powinna być w oddzielnym widoku
* Aplikacja powinna być uniwersalna (Posługiwać się znanymi większości osób wielkościami, umożliwiającymi wykonywanie operacji)
* Uruchomienie aplikacji powinno się odbywać w taki sam sposób niezależnie od środowiska, w którym jest uruchamiana.

# Prototypy komponentów

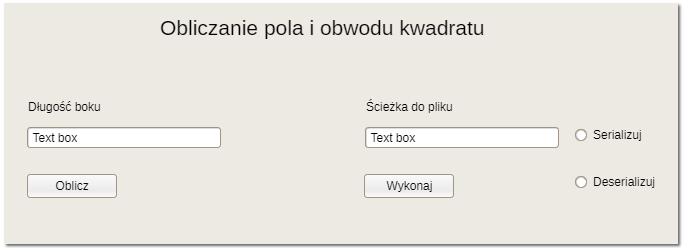
Prototypy komponentów zostały zaprojektowane przy pomocy programu Pencil. Poniżej zrzuty ekranu interfejsów GUI. Każdy z komponentów posiada klasę (model danych), która umożliwia łatwą implementację funkcjonalności dla aplikacji.

## SquareBean

Komponent umożliwiający obliczanie pola i obwodu kwadratu.

Skład pakietu zawierającego komponent:

* **Square** – Klasa kwadratu, umożliwiająca przeprowadzanie obliczeń, serializację oraz posiadająca parametry figury. Implementuje interfejs Serializable.
* **SquareBeanEvent** – Klasa dziedzicząca po EventObject. Umożliwia komunikację między klasami Square oraz SquareBeanPanel, za pomocą zdarzeń.
* **SquareBeanEventListener** – Interfejs umożliwający implementacje obsługi zdarzeń.
* **SquareBeanPanel** – Klasa dziedzicząca po JPanel zawierająca implementację interfejsu GUI komponentu.



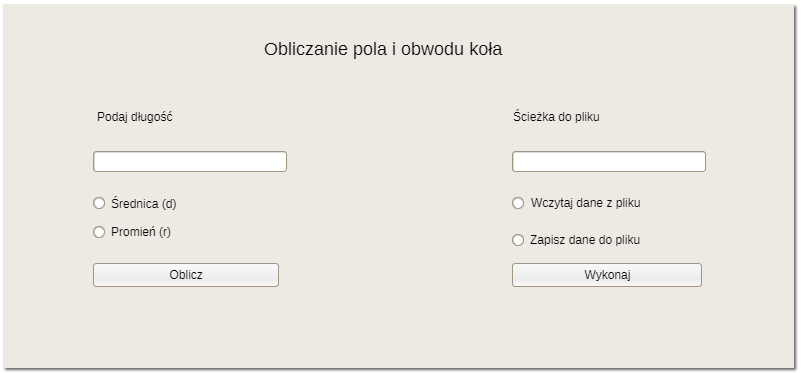
Rys. 1 Zrzut ekranu prototypu komponentu SquareBean.

## CircleBean

Komponent umożliwiający obliczanie pola i obwodu koła i okręgu.

Skład pakietu zawierającego komponent:

* **Circle** – Klasa koła, umożliwiająca przeprowadzanie obliczeń, serializację oraz posiadająca parametry figury. Implementuje interfejs Serializable.
* **CircleBeanEvent** – Klasa dziedzicząca po EventObject. Umożliwia komunikację między klasami Circleoraz CircleBeanPanel, za pomocą zdarzeń.
* **CircleBeanEventListener** – Interfejs umożliwający implementacje obsługi zdarzeń.
* **CircleBeanPanel** – Klasa dziedzicząca po JPanel zawierająca implementację interfejsu GUI komponentu.



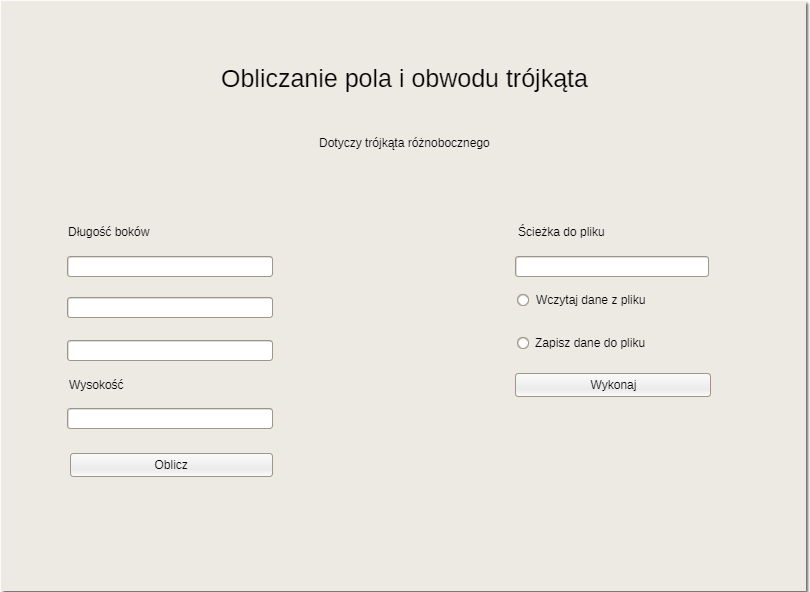
Rys. 2 Zrzut ekranu prototypu komponentu CircleBean.

## TriangleBean

Komponent umożliwiający obliczanie pola i obwodu koła i okręgu.

Skład pakietu zawierającego komponent:

* **Triangle** – Klasa trójkąta, umożliwiająca przeprowadzanie obliczeń, serializację oraz posiadająca parametry figury. Implementuje interfejs Serializable.
* **TriangleBeanEvent** – Klasa dziedzicząca po EventObject. Umożliwia komunikację między klasami Triangleoraz TriangleBeanPanel, za pomocą zdarzeń.
* **TriangleBeanEventListener** – Interfejs umożliwający implementacje obsługi zdarzeń.
* **TriangleBeanPanel** – Klasa dziedzicząca po JPanel zawierająca implementację interfejsu GUI komponentu.



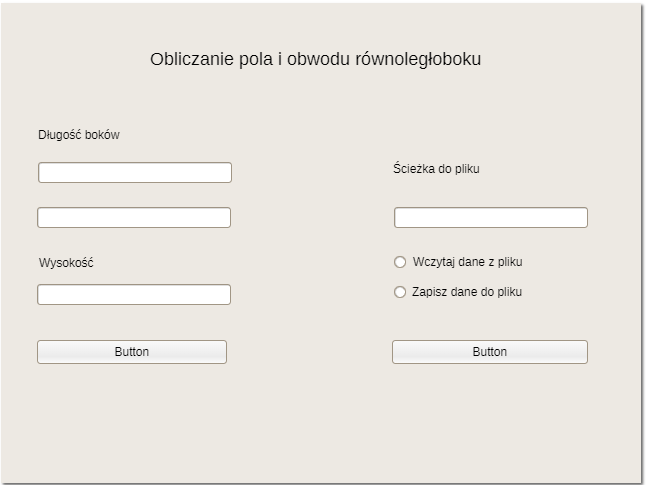
Rys. 3 Zrzut ekranu prototypu komponentu TriangleBean

## ParallelogramBean

Komponent umożliwiający obliczanie pola i obwodu równoległoboku

Skład pakietu zawierającego komponent:

* **Parallelogram** – Klasa trójkąta, umożliwiająca przeprowadzanie obliczeń, serializację oraz posiadająca parametry figury. Implementuje interfejs Serializable.
* **ParallelogramBeanEvent** – Klasa dziedzicząca po EventObject. Umożliwia komunikację między klasami Parallelogramoraz ParallelogramBeanPanel, za pomocą zdarzeń.
* **ParallelogramBeanEventListener** – Interfejs umożliwający implementacje obsługi zdarzeń.
* **ParallelogramBeanPanel** – Klasa dziedzicząca po JPanel zawierająca implementację interfejsu GUI komponentu.



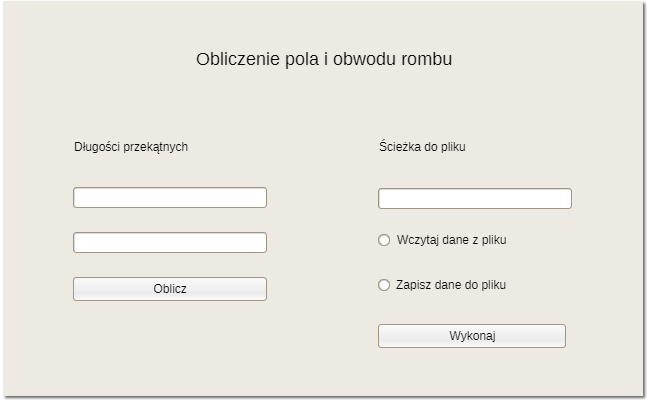
Rys. 4 Zrzut ekranu prototypu komponentu ParallelogramBean

## DiamondBean

Komponent umożliwiający obliczanie pola i obwodu rombu.

Skład pakietu zawierającego komponent:

* **Diamond** – Klasa trójkąta, umożliwiająca przeprowadzanie obliczeń, serializację oraz posiadająca parametry figury. Implementuje interfejs Serializable.
* **DiamondBeanEvent** – Klasa dziedzicząca po EventObject. Umożliwia komunikację między klasami Diamondoraz DiamondBeanPanel, za pomocą zdarzeń.
* **DiamondBeanEventListener** – Interfejs umożliwający implementacje obsługi zdarzeń.
* **DiamondBeanPanel** – Klasa dziedzicząca po JPanel zawierająca implementację interfejsu GUI komponentu.



Rys. 5 Zrzut ekranu prototypu interfejsu komponentu DiamondBean

## TrapezeBean

Komponent umożliwiający obliczanie pola i obwodu trapezu.

Skład pakietu zawierającego komponent:

* **Trapeze** – Klasa trójkąta, umożliwiająca przeprowadzanie obliczeń, serializację oraz posiadająca parametry figury. Implementuje interfejs Serializable.
* **TrapezeBeanEvent** – Klasa dziedzicząca po EventObject. Umożliwia komunikację między klasami Trapeze oraz TrapezeBeanPanel, za pomocą zdarzeń.
* **TrapezeBeanEventListener** – Interfejs umożliwający implementacje obsługi zdarzeń.
* **TrapezeBeanPanel** – Klasa dziedzicząca po JPanel zawierająca implementację interfejsu GUI komponentu.

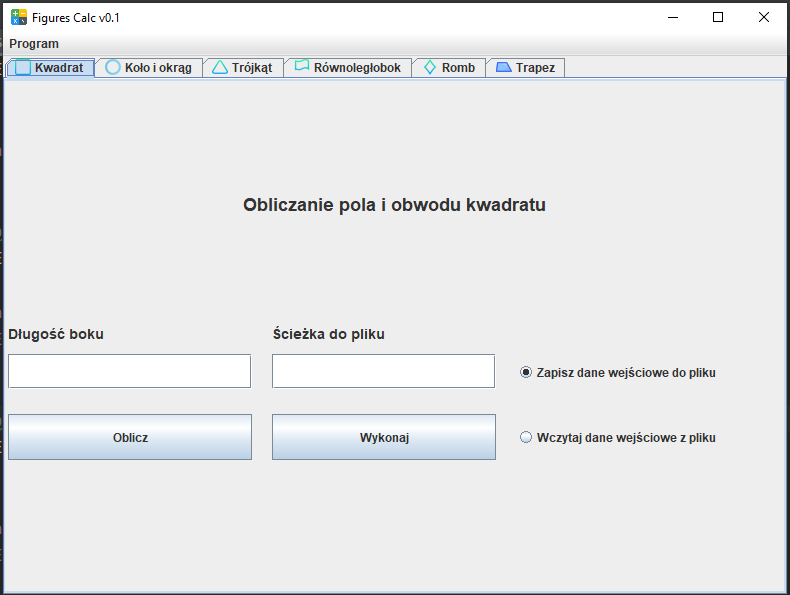
## 

Rys. 6 Zrzut ekranu prototypu komponentu TrapezeBean.

# Aplikacja testująca

Aplikacja testująca została stworzona w celu wykorzystania wcześniej omawianych komponentów w praktyce. Została stworzona w oparciu o wzorzec projektowy MVC. Składa się z okna głównego (klasa MainWindow, która dziedziczy po klasie JFrame), która posiada widok kart, które zawierają interfejsy graficzne oraz funkcjonalność poszczególnych Beanów. Pozwala na szybkie obliczanie pól i obwodów figur płaskich, na podstawie podanych przez użytkownika danych. Wynik obliczeń jest wyświetlany po naciśnięciu przycisku „Oblicz” w oknie w formie podobnej do kontrolki MsgBox popularnej w technologiach firmy Microsoft. Dzięki mechanizmowi serializacji możliwe jest również przechowywanie danych wejściowych w pliku, a także ich przechowywanie na wypadek takiej potrzeby ze strony użytkownika.

Poniżej przykładowe zrzuty ekranu obrazujące działanie aplikacji. Aplikacja dostępna jest pod linkiem do repozytorium na stronie github.com



Rys. 7 Okno główne aplikacji

Jak widać na powyższym zrzucie ekranu interfejs jest dość intuicyjny. Pola tekstowe są duże, a przyciski rozmieszczone równomiernie co pozwala na szybką obsługę. Kolejny zrzut ekranu przedstawia efekt obliczeń.

