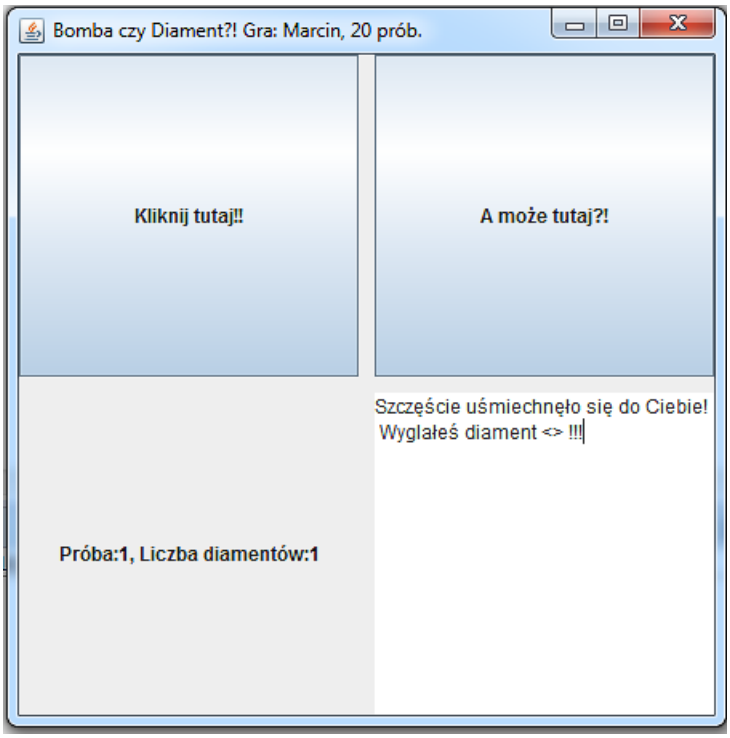
INSTRUKCJE DO ĆWICZEŃ, Programowanie obiektowe (laboratorium)

I. Stworzenie aplikacji „Hello World” na podstawie prostej aplikacji okienkowej z wykorzystaniem pakietu **Swing**. Należy utworzyć okno aplikacji (za pomocą komponentu ***JFrame***) i dodać przycisk „Hello World” (***JButton***). Po kliknięciu przycisku w konsoli powinien pojawić się napis „Hello World!” (można również użyć etykiety ***JLabel***, aby napis pojawił się w okienku aplikacji).

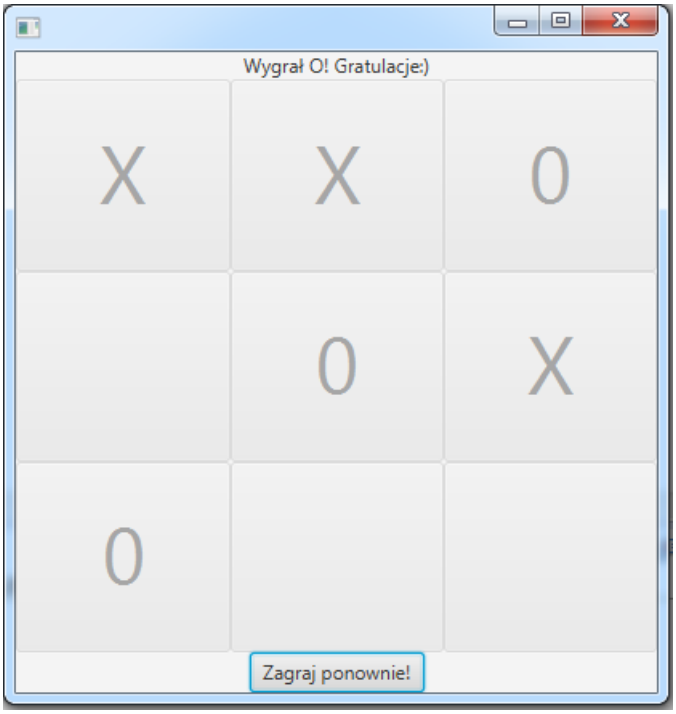
II. Implementacja prostej gry „Bomba czy Diament?”. Stworzyć prosty interfejs graficzny z wykorzystaniem pakietu **Swing** (wykorzystać komponenty ***JButton, JLabel*** oraz ***JTextArea***). Komponenty powinny zostać rozmieszczone za pomocą menedżera rozkładu ***GridLayout*** w okienku aplikacji. Gra ma charakter losowy, dlatego należy wykorzystać klasę ***java.util.Random***. Przy każdej próbie (gra kończy się po 20 próbach) losujemy „Bombę” lub „Diament”, klikając na jeden z dwóch przycisków. Do wprowadzania imienia gracza po uruchomieniu aplikacji należy wykorzystać klasę ***java.util.Scanner*** .



III. Implementacja popularnej gry kółko/krzyżyk z wykorzystaniem biblioteki **JavaFX**.

W ramach wprowadzenia do ćwiczenia prowadzący objaśnia i pokazuje jak korzystać z narzędzia **Scene Builder** do szybkiej budowy interfejsów **JavaFX**. Wspólnie ze studentami buduje prosty interfejs graficzny z komponentem ***javafx.scene.control.Button*** (zdefiniowany w pliku **fxml**), pokazuje jak obsługiwać zdarzenia i uzyskać dostęp do komponentów w kontrolerze poprzez wstrzykiwanie zależności.

Wykorzystując wiedzę i umiejętności zdobyte podczas wprowadzenia, należy zaimplementować grę kółko/krzyżyk o interfejsie graficznym zbliżonym do poniższego. Niezbędne jest również użycie odpowiednich zmiennych sterujących aplikacją oraz zaimplementowanie logiki aplikacji.



IV. **Polimorfizm**, **rzutowanie**, **klasa abstrakcyjna, dziedziczenie, interfejs oraz implementacja interfejsu.** Na wybranym przez siebie przykładzie należy (każdy student powinien indywidualnie zamodelować obiektowo wybrany fragment rzeczywistości):

A) Klasa abstrakcyjna

- Stworzyć klasę abstrakcyjną z wybraną metodą abstrakcyjną.
- Stworzyć 2-3 klasy, które dziedziczą po klasie abstrakcyjnej i implementują wybraną metodę.
- Utworzyć instancje klas potomnych.
- Dokonać rzutowania w górę, np. umieszczając obiekty w tablicy typu klasy bazowej.
- Dokonać wywołań polimorficznych zaimplementowanej metody (np. iterując po obiektach rzutowanych do klasy bazowej, znajdujących się w tablicy).

B) Interfejs

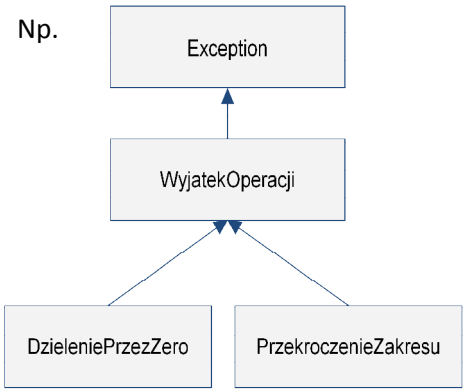
- Stworzyć interfejs z wybraną metodą.
- Stworzyć 2-3 klasy, które implementują interfejs.
- Utworzyć instancje utworzonych klas.

- Dokonać rzutowania na typ interfejsowy, np. przypisując obiekty do tablicy elementów typu interfejsowego.
- Dokonać wywołań polimorficznych zaimplementowanej metody (np. iterując po obiektach rzutowanych na typ interfejsowy, znajdujących się w tablicy).

V. Tworzenie własnej **rodziny wyjątków** w aplikacjach Java. Na wybranym przykładzie (każdy student powinien wymyśleć i zaimplementować swój własny przykład, prezentowane tutaj rysunki są przykładowe) stworzyć **rodzinę wyjątków, które mogą być zgłaszane przez aplikację**.

Prosty interfejs graficzny pozwoli na wprowadzanie danych i reakcję aplikacji w przypadku podania niepoprawnych danych. Wyskakujące okienko dialogowe, np. ***javafx.scene.control.Alert (AlertType.ERROR)***, ma informować użytkownika o zgłoszonym wyjątku i prezentować informację pobraną z obiektu wyjątku za pomocą metody ***getMessage()*.**

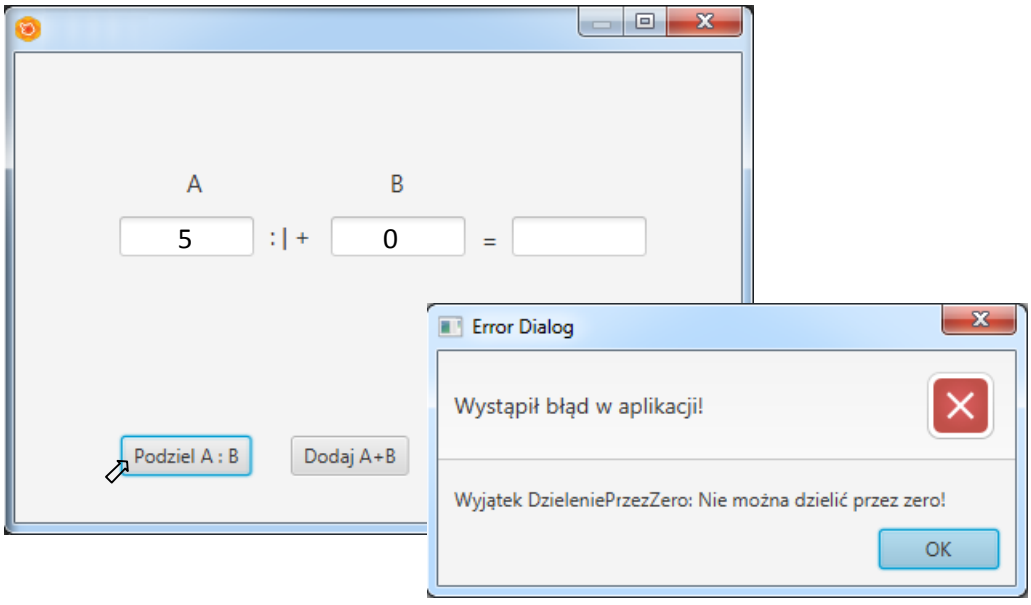
Np.



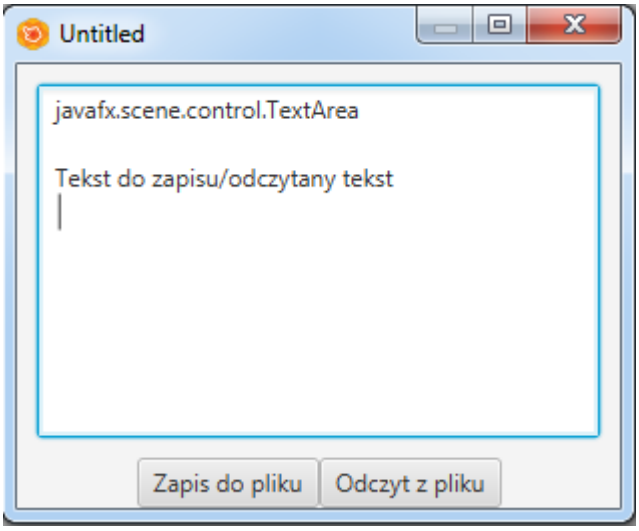

```
class WyjatekOperacji extends Exception { ...}


class DzieleniePrzezZero extends WyjatekOperacji {
...
}
class PrzekroczenieZakresu extends WyjatekOperacji {
...
}
```

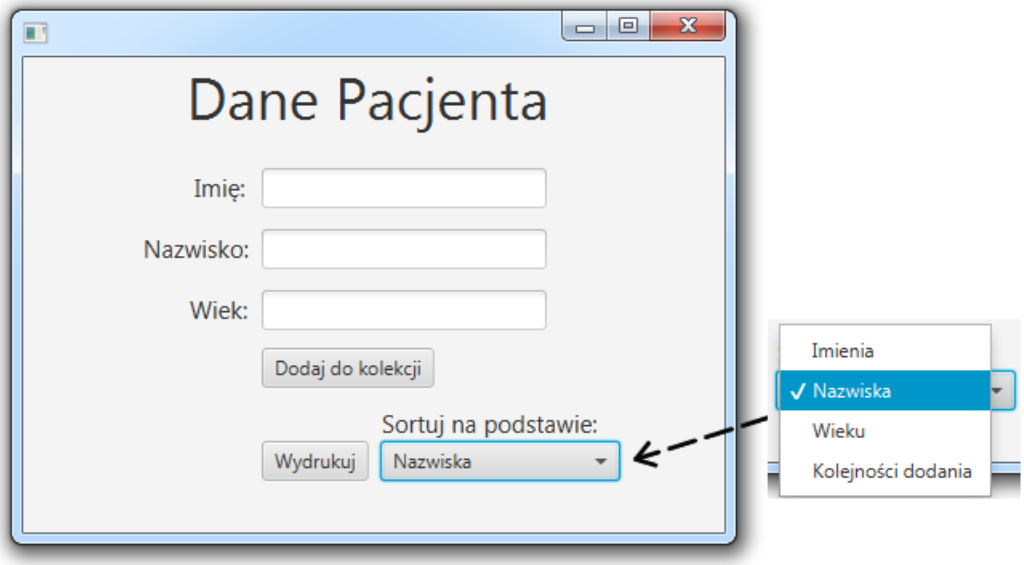
Np.



VI. Zaimplementować możliwość zapisu/odczytu do/z pliku tekstowego. W tym celu stworzyć w katalogu domyślnym plik *.txt* o wybranej nazwie oraz prosty interfejs graficzny widoczny poniżej. Przycisk "Odczyt z pliku" ma powodować wczytanie tekstu ze stworzonego pliku do obszaru tekstowego. Dowolny tekst wpisany w obszarze tekstowym po wciśnięciu przycisku "Zapis do pliku" powinien zostać zapisany w tymże pliku.



VII. Wykorzystać implementację struktury danych opartą na liście z wykorzystaniem ***klasy java.util.ArrayList<E>***. Kolekcja ta powinna przechowywać **obiekty klasy Pacjent** (parametr E), która zawiera prywatne pola Imię, Nazwisko oraz Wiek, gettery i settery, jak również konstruktor umożliwiający wprowadzenie danych pacjenta w momencie tworzenia obiektu.



Program powinien umożliwiać wprowadzanie danych pacjentów i dodawanie ich do wykorzystywanej kolekcji. Funkcja wydruku pozwalać ma na wyprowadzenie danych wszystkich wprowadzonych pacjentów na konsolę w porządku alfabetycznych ze

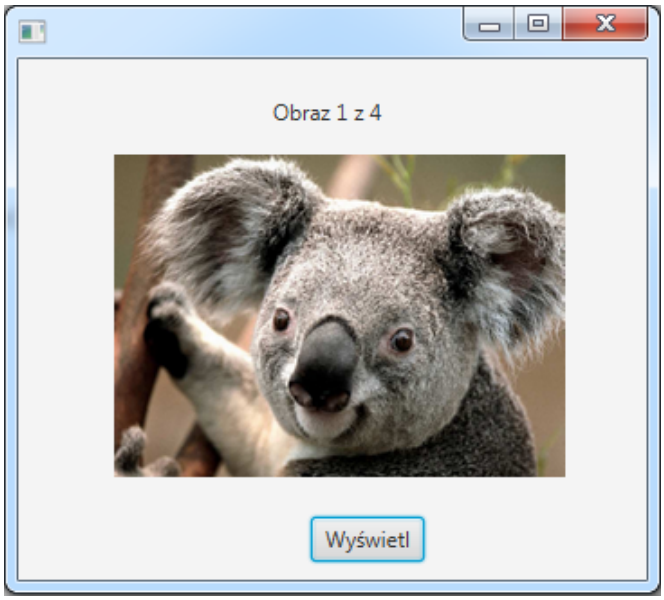
względu na imię lub nazwisko, jak również zgodnie w wiekiem pacjenta. Wybór sposobu sortowania za pomocą komponentu ***javafx.scene.control.ChoiceBox<T>***.
Dane należy wyprowadzać na konsolę w formacie umożliwiającym wymianę danych **JSON** (ang. *JavaScript Object Notation*).

Np. {
"imie": "Jan",
"nazwisko": "Kowalski",
"wiek": 64
}

Zadanie dodatkowe (**dla chętnych**). Walidacja wprowadzanych danych za pomocą wyrażeń regularnych z wykorzystaniem pakietu **java.util.regex**.

VIII. Napisać aplikację, której zadaniem będzie synchronizacja dwóch katalogów o nazwie A i B. Katalog A ma być katalogiem wzorcowym, natomiast katalog B ma być dokładną kopią zawartości katalogu A. Dla ułatwienia zakładamy, że porównywanie plików w katalogach A i B odbywa się tylko na podstawie nazwy plików. Funkcjonalność należy zaimplementować w osobnym wątku, który co sekundę będzie porównywał zawartość katalogów.

IX. Prosta przeglądarka plików graficznych. Należy przygotować 4 pliki graficzne (np. JPG o rozdzielczości 300x400) przed wykonaniem ćwiczenia. Napisać aplikację z wykorzystaniem biblioteki **JavaFX**, która po wciśnięciu przycisku "Wyświetl" spowoduje wczytanie i wyświetlenie pierwszego z przygotowanych obrazów. Dalsze klikanie przycisku ma spowodować cykliczne wyświetlanie kolejnych plików (po wyświetleniu 4-tego obrazu powinien zastać wczytany ponownie obraz pierwszy).



Przydatne linki (tutoriale oraz dokumentacja):

- ✓ https://docs.oracle.com/javase/tutorial/
- ✓ https://docs.oracle.com/javase/8/docs/api/

Warunkiem zaliczenia jest samodzielne wykonanie powyższych ćwiczeń i oddanie sprawozdań z zaimplementowanych aplikacji. Sprawozdania są podstawą oceny i powinny zawierać :

- ✓ napisany przez studenta kod aplikacji;
- ✓ wydruk interfejsu graficznego (jeśli występuje);
- ✓ wnioski i spostrzeżenia.

Na końcową ocenę ma również wpływ aktywność studenta na laboratorium, jego przygotowanie do zajęć i wynikające z tego widoczne postępy podczas wykonywania ćwiczenia.