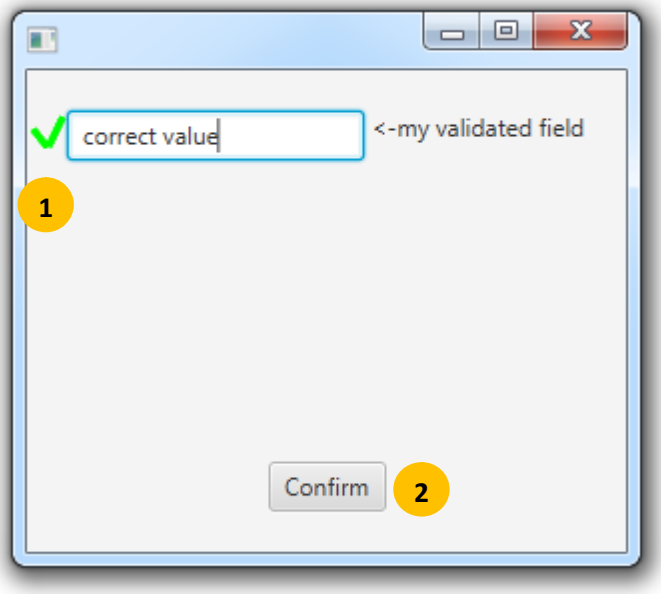
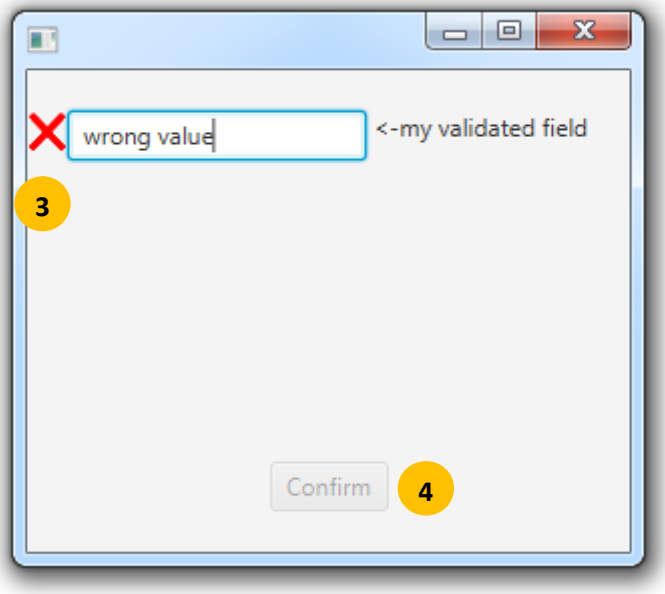
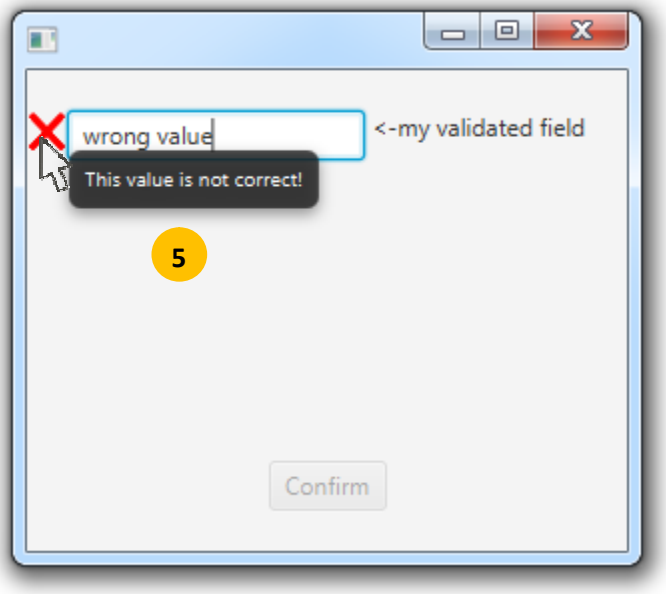
ĆWICZENIE: Walidacja z wykorzystaniem własnej adnotacji

(3 godziny zajęciowe)

Napisać aplikację okienkową (z wykorzystaniem bilbioteki Swing lub JavaFX), która umożliwi walidację wybranego pola dowolnej klasy w konwencji JavaBean, oznaczonego własną adnotacją. Zadania do wykonania i wymagania:





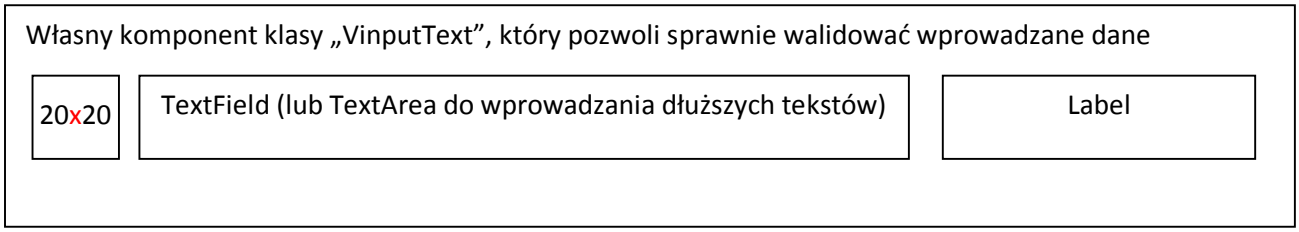


- ✓ Utworzyć nową adnotację do walidacji oznaczonych pól klasowych. Adnotacja powinna bazować na wyrażeniach regularnych, które posłużą do sprawdzania poprawności wprowadzanych danych.
- ✓ Dla dowolnej klasy utworzonej w konwencji JavaBean wybrać i oznaczyć za pomocą stworzonej adnotacji pole, które podlegać będzie walidacji, np.

```
…
@MyPattern(regex="correct value", message = "This value is not correct!")
String myValidatedField="";
…
```

- ✓ Stworzyć dwa proste obrazki, np. „0.png” oraz „1.png” o rozdzielczości 20x20 pikseli, które reprezentować będą wynik walidacji (1, 3).

- ✓ Skorzystać z klasy HBox (JavaFX) lub `BoxLayout` (Swing) w celu utworzenia klasy o nazwie „VinputText”, wykorzystującej kontrolkę `TextInputControl` (JavaFX) lub komponent `JTextComponent` (Swing). Takie rozwiązanie pozwoli zastosować pole lub obszar tekstowy w zależności od potrzeby.



- ✓ Klasa `VinputText` powinna zawierać metodę `registerValidator(Validator v)`, aby umożliwić zarejestrowanie obiektu walidatora. Po jego zarejestrowaniu, komponent `VinputText` będzie wykorzystywał ten obiekt do sprawdzania wprowadzanych danych (w przypadku zadania dodatkowego poniżej będzie to wiele walidatorów rejestrowanych i przechowywanych w stosownej kolekcji).
- ✓ Wykorzystać mechanizm refleksji w celu wykrycia i zastosowania stworzonej adnotacji.
- ✓ Utworzyć odpowiednią klasę walidatora. Przyjmijmy następującą konwencję: nazwa klasy walidatora tworzona jest przez dodanie do nazwy adnotacji słowa „Validator”, np. dla adnotacji `@MyPattern` klasa walidująca będzie nosić nazwę `MyPatternValidator.`
- ✓ Klasa walidatora powinna pozwalać na walidację zgodnie z parametrami przekazanymi w adnotacji (np. wyrażenie regularne przekazane jako parametr **regex**). Natomiast parametr **message** ma umożliwiać przekazywanie wiadomości, która wyświetlana będzie jako Tooltip po najechaniu kursorem na **x** (5).
- ✓ Klasa walidatora powinna implementować interfejs Validator. Metoda `validate(String value)` na bazie parametrów adnotacji dokonuje walidacji wartości wprowadzonej za pomocą komponentu `VinputText.` Wynik walidacji przechowywany jest w prywatnym polu `valid` typu `boolean`, do którego dostęp mamy za pomocą publicznej metody `isValid()`, zwracającej wartość `true` jeśli walidacja powiedzie się albo `false` w przeciwnym przypadku. Metoda `getMessage()` zwraca wiadomość zawartą w parametrze `message` utworzonej adnotacji.

```
public interface Validator {
   void validate(String value);
   boolean isValid();
   String getMessage();
}
```

- ✓ Przycisk Confirm powinien być aktywny tylko wtedy, gdy wprowadzona wartość jest poprawna (2). Dzięki temu, nie będzie możliwości wprowadzenia niepoprawnych danych (4).
- ✓ Walidacja powinna odbywać się automatycznie w czasie rzeczywistym w trakcie wprowadzania danych.

Zadanie dodatkowe (dla chętnych)

- ✓ Usprawnić zaimplementowane rozwiązanie, aby można było umieszczać wiele adnotacji do walidacji nad wybranym polem, jak również umieszczać je nad wieloma polami klasowymi.
- ✓ Usprawnione rozwiązanie zastosować w aplikacji z poprzedniego ćwiczenia.