

Programowanie obiektowe

Wykład 7

Marcin Młotkowski

6 kwietnia 2017

Plan wykładu

- 1 Z życia programisty, część 1
- 2 Swing
 - Okna i kontrolki
 - Obsługa zdarzeń
- 3 Pułapki i rozwiązania
 - Z życia programisty, część 2
 - Model–View–Controller
 - MVC w Swingu

Etap 1

Specyfikacja

Zaprogramować system do obsługi biblioteki.

Fragment implementacji

```
public class Ksiazka
{
    private String tytuł;
    private String autor;
    private int wydanie;
    public Ksiazka(String t, String a, int w)
    {
        this.tytuł = t; this.autor = a; this.wydanie = w;
    }
    public String toString()
    {
        return "Książka " + this.tytuł + " " + this.autor;
    }
}
```

Etap 2

Rozszerzenie specyfikacji

Ale żeby jeszcze dało się edytować dane w okienku!

Implementacja w Javie

```
public class Ksiazka
{
    private String tytuł;
    private String autor;
    private int wydanie;
    public Ksiazka(String t, String a, int w) { ... }
    public String toString() { ... }
    public void Edycja()
    {
        ...
    }
}
```

Plan wykładu

- 1 Z życia programisty, część 1
- 2 **Swing**
 - Okna i kontrolki
 - Obsługa zdarzeń
- 3 Pułapki i rozwiązania
 - Z życia programisty, część 2
 - Model–View–Controller
 - MVC w Swingu

Wprowadzenie do biblioteki Swing

Najważniejsze informacje

- istotnie ulepszona wersja AWT (Abstract Window Toolkit);
- niezależność wyglądu i innych mechanizmów od systemu operacyjnego;
- wsparcie dla internacjonalizacji.

Intensywnie korzysta z mechanizmów obiektowych: klas, obiektów i dziedziczenia.

Z czego składa się Swing

Część widoczna

Okna, kontrolki i menu

Obsługa kliknięć

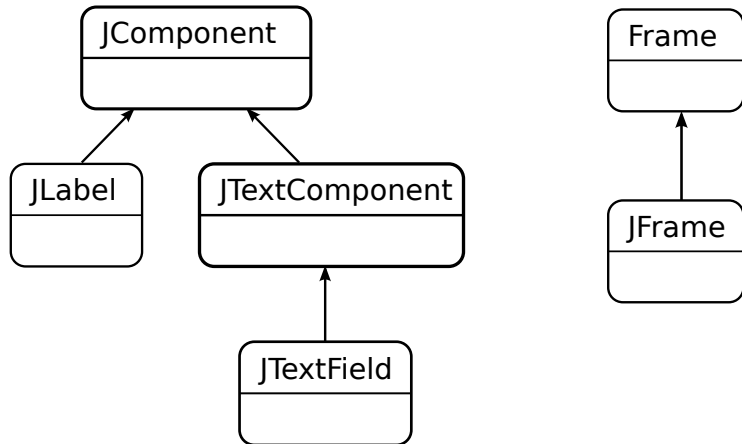
Zdarzenia i słuchacze.

Gdzie jest Swing?

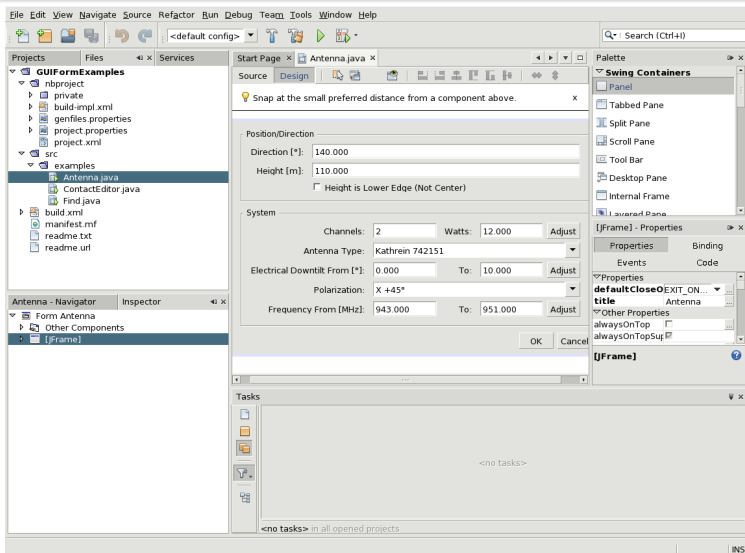
```
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.*;
```

Hierarchia klas

`javax.swing.*`



Szybkie rozwiązanie



Okno główne

```
JFrame frame = new JFrame("Edycja książki");  
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Okno główne

```
JFrame frame = new JFrame("Edycja książki");  
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
Container kontener = frame.getContentPane();  
GridLayout layout = new GridLayout(4, 2);  
kontener.setLayout(layout);
```

Edycja autora

```
JLabel autor_etykieta = new JLabel("Autor");  
kontener.add(autor_etykieta);  
JTextField autor = new JTextField(this.autor, 40);  
kontener.add(autor);
```

Edycja tytułu

```
JLabel tytuł_etykieta = new JLabel("Tytuł");  
kontener.add(tytuł_etykieta);  
JTextField tytuł = new JTextField(this.tytuł, 40);  
kontener.add(tytuł);
```


Edycja wydania

```
JLabel wydanie_etykieta = new JLabel("Wydanie");  
kontener.add(wydanie_etykieta);  
JTextField wydanie =  
    new JTextField(Integer.toString(this.wydanie), 40);  
kontener.add(wydanie);
```

Przycisk zapisu danych

```
 JButton b = new JButton("Zapisz");  
 b.addActionListener(this);  
 kontener.add(b);
```

Zakończenie budowania okna edycji

```
frame.pack();  
frame.setVisible(true);
```

Reakcja na kliknięcie

```
public class Ksiazka implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        // np. pobranie z kontrolki wpisanych danych: Jtextfield.getText();
    }
    public void Edycja()
    {
        b.addActionListener(this);
    }
}
```

Wynik

Autor	Adam Mickiewicz
Tytuł	Pan Tadeusz
Wydanie	7
Zapisz	

Obsługa zdarzeń

Zdarzenie

Kliknięcie myszką, naciśnięcie klawisza, upływanie czasu, zakończenie jakiejś operacji.

Kontrolka

Kontrolka związana ze zdarzeniem, np. kliknięty przycisk.

Obsługa zdarzenia

Akcja (metoda) wykonywana po zaistnieniu zdarzenia.

Subskrypcja zdarzeń w Swingu

Kontrolka "przyjmująca zdarzenia"

Kontrolka ma listę klas, które implementują interfejs `ActionListener`. Po każdym zdarzeniu wszystkie elementy listy są informowane o zaistnieniu zdarzenia.

Słuchacz

Implementuje metodę `actionPerformed(ActionEvent)`, która jest wywoływana w momencie wystąpienia zdarzenia.

Implementacja nasłuchu

```
public class Ksiazka implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        ...
    }

    public void Edycja()
    {
        ...
        JButton b = new JButton("Zapisz");
        b.addActionListener(this);
        ...
    }
}
```


Słuchacze

- Obiekty mogą rejestrować Słuchaczy (dowolną ich liczbę);
- informacja o zmianach jest rozsyłana przez obiekt do wszystkich słuchaczy;
- w Swingu Słuchacz implementuje interfejs `EventListener` i pochodnych, w tym `ActionListener`.

Implementacja słuchacza, 1. podejście

```
public class Ksiazka implements ActionListener
{
    b.addActionListener(this);
}
```

Implementacja słuchacza, 2. podejście

```
public class Ksiazka
{
    class MyListener implements ActionListener
    {
        public void actionPerformed(ActionEvent e) { ... }
    }

    ...
    b.addActionListener(new MyListener());
    ...
}
```

Implementacja słuchaczy, 3. podejście

Klasa anonimowa

```
b.ActionListener(  
    new ActionListener()  
    {  
        public void actionPerformed(ActionEvent evt) { ... }  
    }  
);
```

Własności klasy anonimowej

- można definiować "w locie" i nie trzeba wymyślać nazwy;
- klasa anonimowa ma dostęp do zmiennych obiektu macierzystego.

Plan wykładu

- 1 Z życia programisty, część 1
- 2 Swing
 - Okna i kontrolki
 - Obsługa zdarzeń
- 3 Pułapki i rozwiązania
 - Z życia programisty, część 2
 - Model-View-Controller
 - MVC w Swingu

Rozwój oprogramowania

Ale żeby działało na komórce/palmtopie/....

Rozwój oprogramowania

Ale żeby działało na komórce/palmtopie/....

Implementacja

```
public class Ksiazka
{
    public void Edycja() { ... }
    public void EdycjaWersjaKomórkowa() { ... }
    public void EdycjaWersjaPalmtop() { ... }
}
```


Wynik

- implementacja "właściwej" klasy Książka: 15 wierszy;
- implementacja edycji (tylko Swing): 40 wierszy

Refleksja

Czy klasa Książka ma dobrą nazwę?

Co jest w środku klasy Książka?

- implementacja funkcjonalności związanej z modelowaniem książki
- kod odpowiedzialny za interakcję ze środowiskiem okienkowym

Reguła programowania obiektowego

Zasada pojedynczej odpowiedzialności

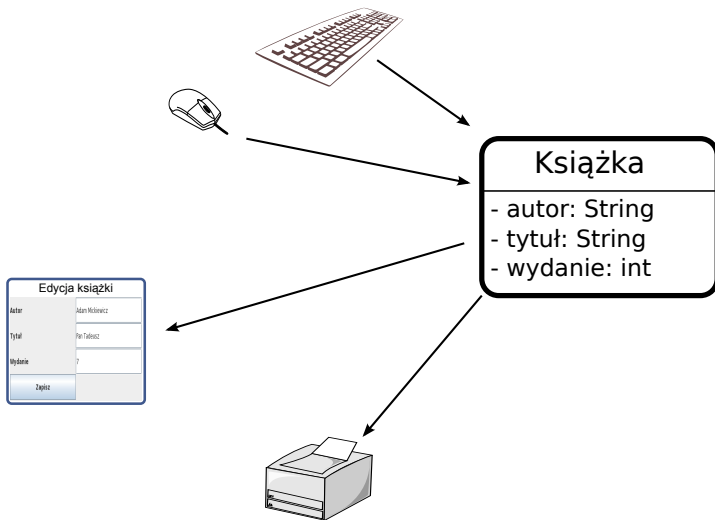
Klasa powinna mieć wyłącznie jeden obszar odpowiedzialności.

Wprowadzenie

Wzorzec projektowy MVC: Model-View-Controller

- jeden z pierwszych wzorców projektowych
- zaprojektowany dla biblioteki graficznej

Interakcja z otoczeniem



MVC

Model

Dane oraz zależności między nimi

View

Widok danych (ekran, drukarka, html)

Controller

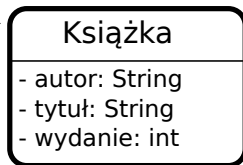
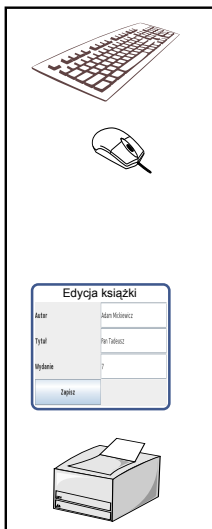
"Coś" co modyfikuje stan Modelu (np. przyciski na ekranie).

Współdziałanie elementów

Kontroler ma referencję do Modelu i na podstawie akcji użytkownika (naciśnięcie klawisza) wywołuje odpowiednie metody modyfikujące stan Modelu.

Widok (lub Widoki) nasłuchują zmian w Modelu i aktualizują wygląd w przypadku modyfikacji.

MVC w Swingu



Ulepszanie kodu

Rozdzielenie klasy na dwie klasy:

```
public class Ksiazka  
{  
    ...  
}
```

```
public class KsiazkaWidokSwing  
{  
    public KsiazkaWidok(Ksiazka k) { ... }  
}
```

Podsumowanie

- Rozdzielenie klas względem odpowiedzialności daje naturalny podział kodu;
- zastosowanie wzorca MVC lub podobnego daje możliwość niezależnej implementacji klas modelu i skojarzonych z nimi interfejsów graficznych;
- definiowanie podklasy modelu może naturalnie być odzwierciedlone w podklasie interfejsu.

Ocena dotychczasowego rozwiązania

- rozdzielono obiekt od jego "edytora";
- uruchomienie edytora wymaga przesłania danych z obiektu (modelu) do edytora;
- zakończenie pracy edytora wymaga przesłania poprawionych danych z edytora do obiektu.

Marzenie programisty

Powiązanie kontroltek (takich jak JTextField) bezpośrednio z danymi.

Rozwiązanie w Swingu

Koncepcja dokumentu:

- dokument jest kontenerem na tekst; zarówno na prosty tekst jak i na ustrukturalizowane dokumenty (html, xml)
- dokumenty można podłączać jako obserwatory do kontrolki, wtedy transfer danych z kontrolki do dokumentu jest automatyczny;
- dalsze informacje:
`javax.swing.text.AbstractDocument` i podklasy.