

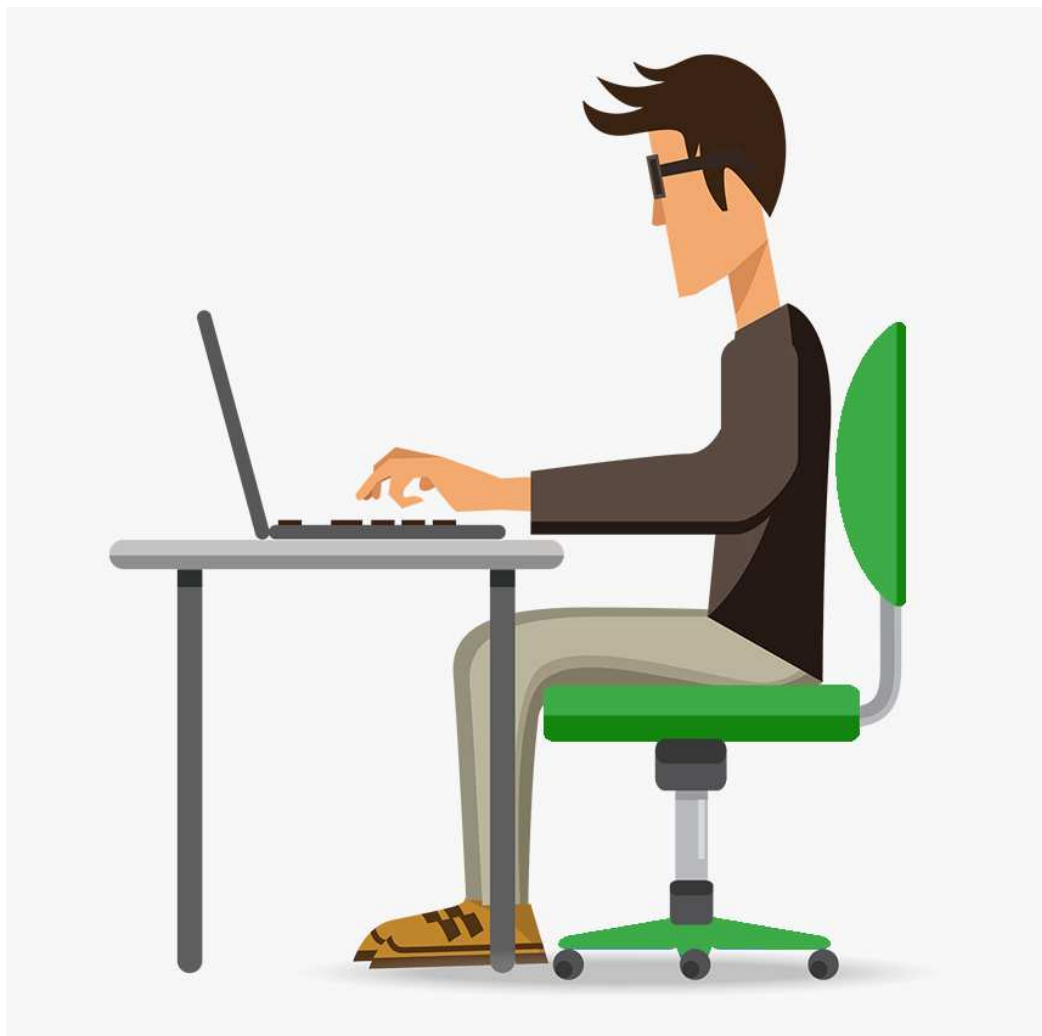
Stefan Sokołowski

PROGRAMOWANIE OBIEKTOWE II

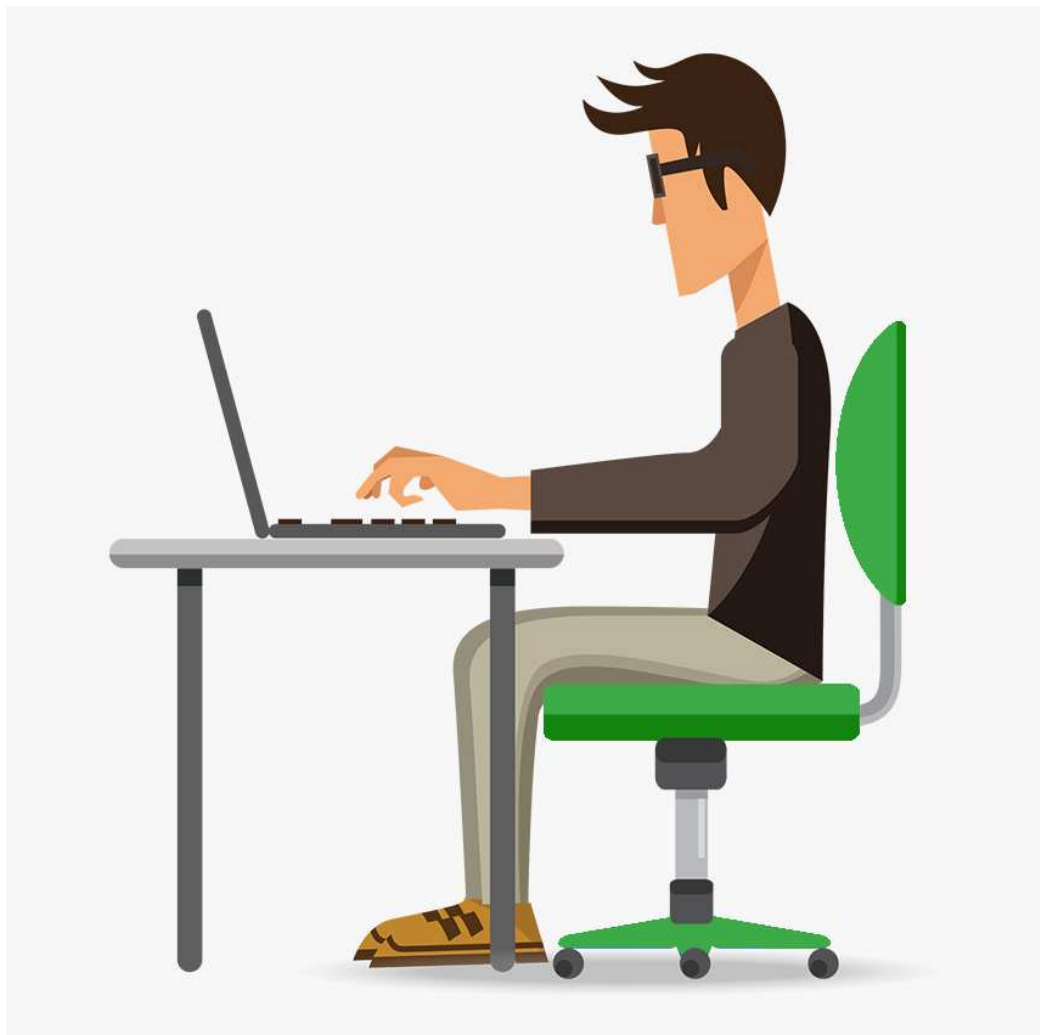
wykład 9

**Inst. Informatyki Stosowanej, ANS
Elbląg, 2024/2025**

Zdarzenia

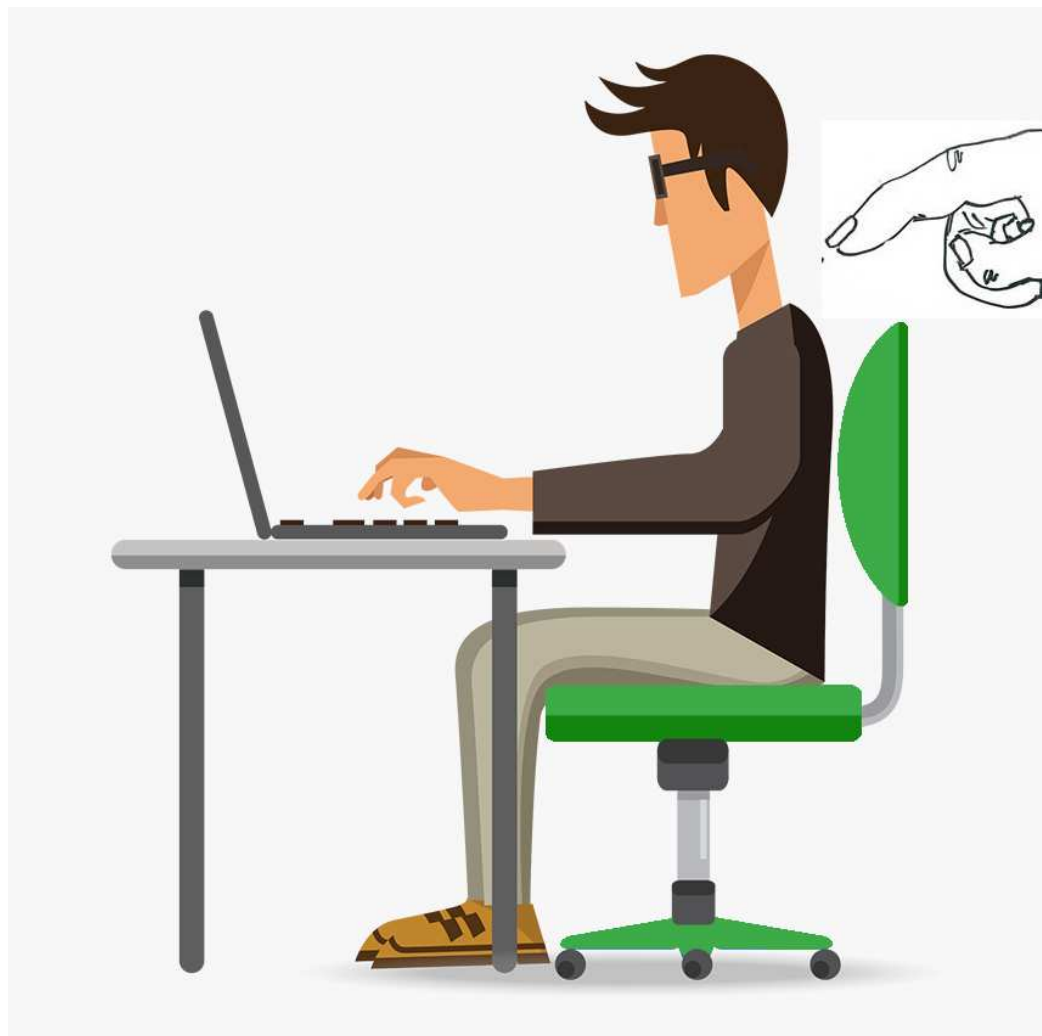


Zdarzenia



I NAGLE...

Zdarzenia



I NAGLE...

Reakcja na zdarzenia

normalne działanie
programu:



Reakcja na zdarzenia

normalne działanie
programu:

odczytać komendę

```
graph TD; A[normalne działanie programu:] --> B[odczytać komendę];
```

Reakcja na zdarzenia

normalne działanie
programu:

odczytać komendę

wykonać

Reakcja na zdarzenia

normalne działanie
programu:

odczytać komendę

wykonać

odczytać komendę



Reakcja na zdarzenia

normalne działanie
programu:

odczytać komendę

wykonać

odczytać komendę

wykonać

Reakcja na zdarzenia

normalne działanie programu:

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

wykonać

Reakcja na zdarzenia

normalne działanie programu:

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

wykonać

Reakcja na zdarzenia

normalne działanie programu:

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

Reakcja na zdarzenia

normalne działanie programu:

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

ew. zdarzenie:

klawiatura

Reakcja na zdarzenia

normalne działanie programu:

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

ew. zdarzenie:

klawiatura

pożądana reakcja:

zapisać
klawisz

Reakcja na zdarzenia

normalne działanie programu:

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

wykonać

odczytać komendę

ew. zdarzenie:

klawiatura

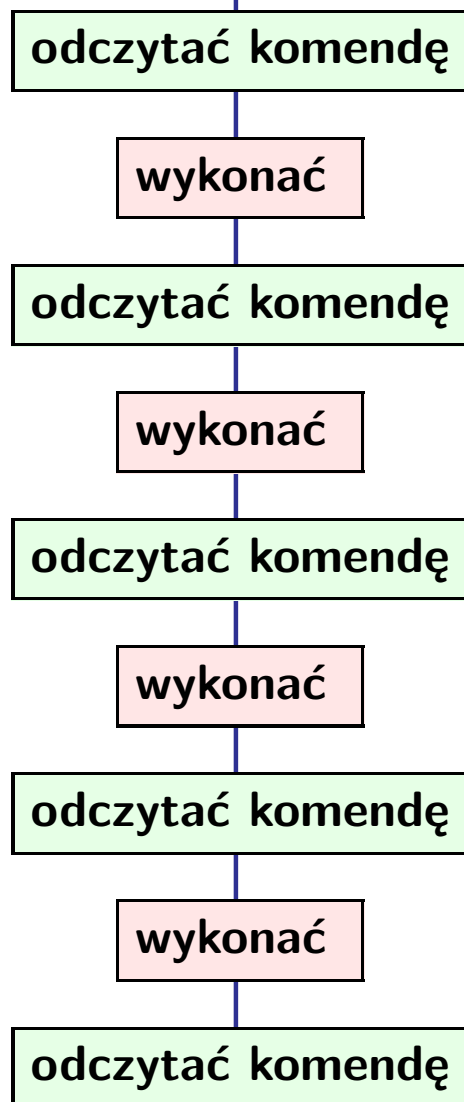
czasomierz

pożądana reakcja:

zapisać
klawisz

Reakcja na zdarzenia

normalne działanie programu:



ew. zdarzenie:

klawiatura

czasomierz

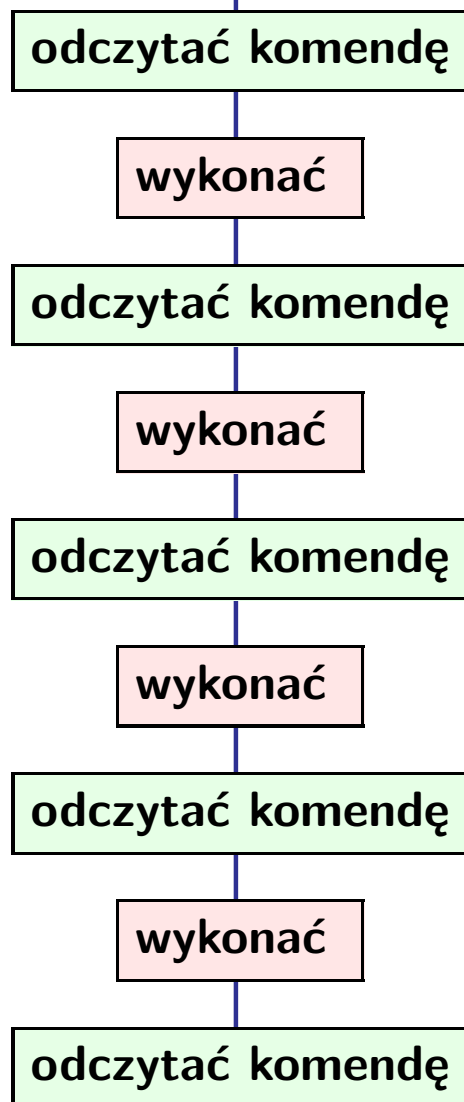
pożądana reakcja:

zapisać
klawisz

zanotować
kolejną
milisekundę

Reakcja na zdarzenia

normalne działanie programu:



ew. zdarzenie:

klawiatura

czasomierz

klik myszą

pożądana reakcja:

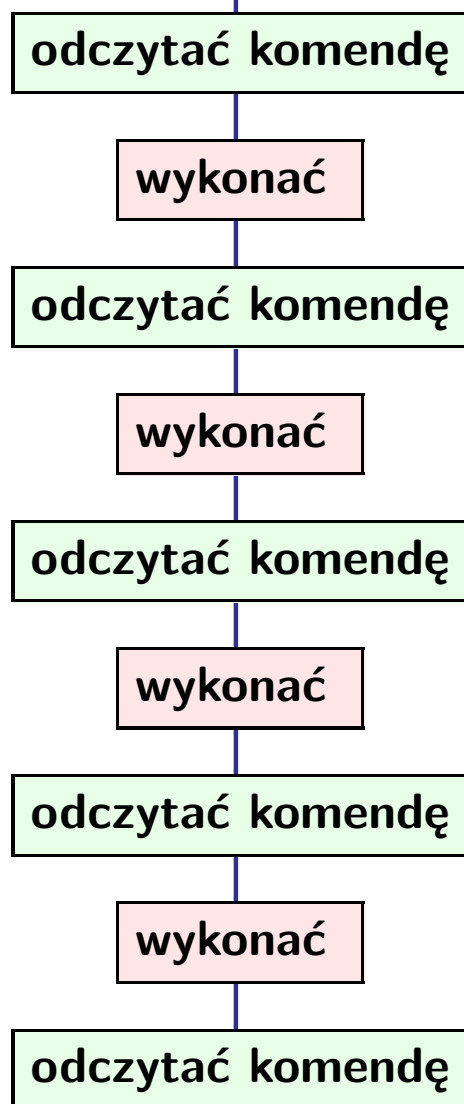
zapisać
klawisz

zanotować
kolejną
milisekundę

zależnie
od
sytuacji

Reakcja na zdarzenia

normalne działanie programu:



ew. zdarzenie:

klawiatura

czasomierz

klik myszą

internet

pożądana reakcja:

zapisać
klawisz

zanotować
kolejną
milisekundę

zależnie
od
sytuacji

...

Reakcja na zdarzenia

Zdarzenia mogące zajść w czasie działania programu

Reakcja na zdarzenia

Zdarzenia mogące zajść w czasie działania programu:

- wciśnięcie przycisku klawiatury,
- zwolnienie przycisku klawiatury

Reakcja na zdarzenia

Zdarzenia mogące zajść w czasie działania programu:

- wciśnięcie przycisku klawiatury,
- zwolnienie przycisku klawiatury,
- wciśnięcie przycisku myszy,
- zwolnienie przycisku myszy,
- przesunięcie myszy,
- przekręcenie kółeczka myszy

Reakcja na zdarzenia

Zdarzenia mogące zajść w czasie działania programu:

- wciśnięcie przycisku klawiatury,
- zwolnienie przycisku klawiatury,
- wciśnięcie przycisku myszy,
- zwolnienie przycisku myszy,
- przesunięcie myszy,
- przekręcenie kółeczka myszy,
- upływ czasu („przerwanie czasomierza”)

Reakcja na zdarzenia

Zdarzenia mogące zajść w czasie działania programu:

- wciśnięcie przycisku klawiatury,
- zwolnienie przycisku klawiatury,
- wciśnięcie przycisku myszy,
- zwolnienie przycisku myszy,
- przesunięcie myszy,
- przekręcenie kółeczka myszy,
- upływ czasu („przerwanie czasomierza”),
- komunikat z urządzenia zewnętrznego (np. drukarki, lub internetu),
- itd.

Reakcja na zdarzenia

Obsługa zdarzenia

Reakcja na zdarzenia

Obsługa zdarzenia:

- *hardware* zatrzymuje działanie procesora i zapisuje stan jego rejestrów i adres powrotu

Reakcja na zdarzenia

Obsługa zdarzenia:

- *hardware* zatrzymuje działanie procesora i zapisuje stan jego rejestrów i adres powrotu;
- zostaje uruchomiony program *obsługi* zdarzenia (np. uaktualnienia bufora klawiatury)

Reakcja na zdarzenia

Obsługa zdarzenia:

- *hardware* zatrzymuje działanie procesora i zapisuje stan jego rejestrów i adres powrotu;
- zostaje uruchomiony program *obsługi* zdarzenia (np. uaktualnienia bufora klawiatury) — najczęściej ten program należy do systemu operacyjnego, ale **możemy podstawić własny callback**

Reakcja na zdarzenia

Obsługa zdarzenia:

- *hardware* zatrzymuje działanie procesora i zapisuje stan jego rejestrów i adres powrotu;
- zostaje uruchomiony program *obsługi* zdarzenia (np. uaktualnienia bufora klawiatury) — najczęściej ten program należy do systemu operacyjnego, ale **możemy podstawić własny callback**;
- ten program odtwarza poprzedni stan procesora i podejmuje przerwane działania.

Reakcja na zdarzenia

Obsługa zdarzenia:

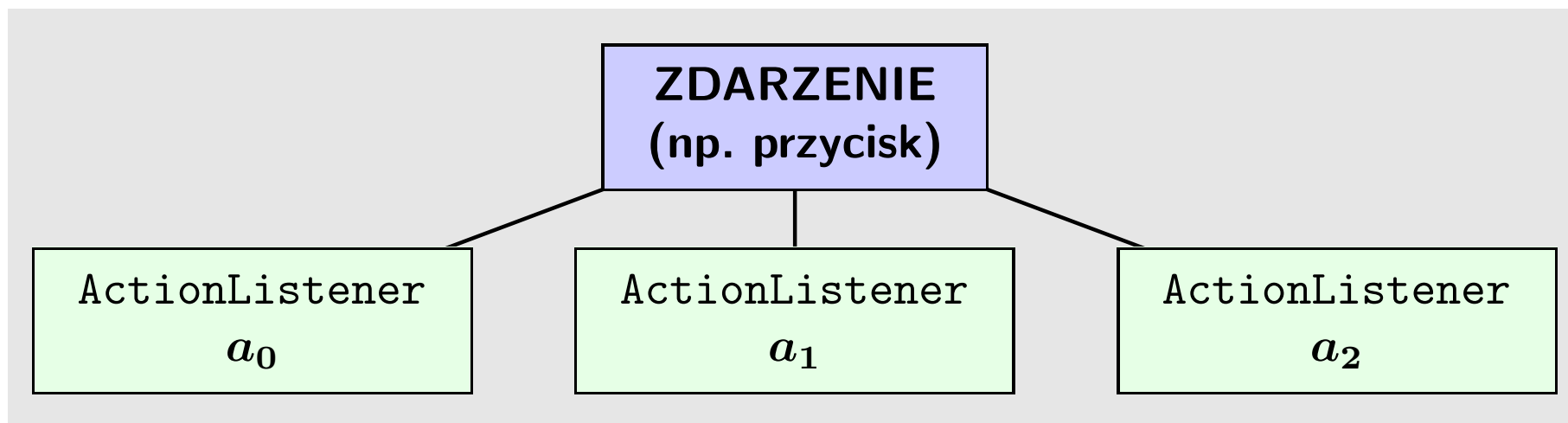
- *hardware* zatrzymuje działanie procesora i zapisuje stan jego rejestrów i adres powrotu;
- zostaje uruchomiony program *obsługi* zdarzenia (np. uaktualnienia bufora klawiatury) — najczęściej ten program należy do systemu operacyjnego, ale **możemy podstawić własny callback**;
- ten program odtwarza poprzedni stan procesora i podejmuje przerwane działania.

tym się teraz zajmiemy



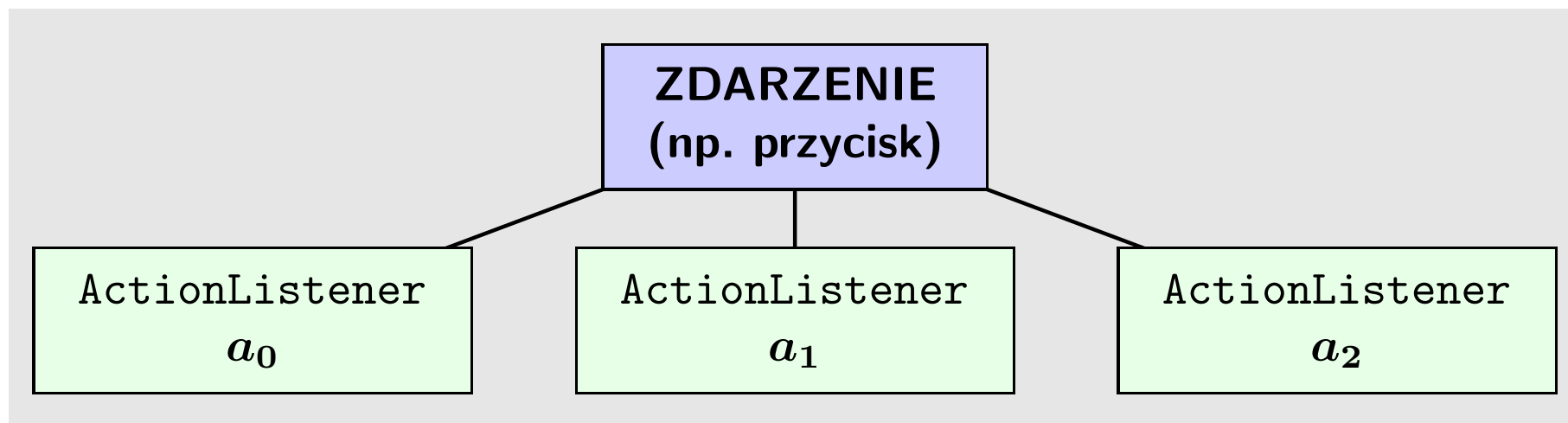
Reakcja na zdarzenia

Z każdym możliwym zdarzeniem jest związana *lista „nasłuchiwczy”*, czy się zdarzyło:



Reakcja na zdarzenia

Z każdym możliwym zdarzeniem jest związana *lista „nasłuchiwczy”*, czy się zdarzyło:

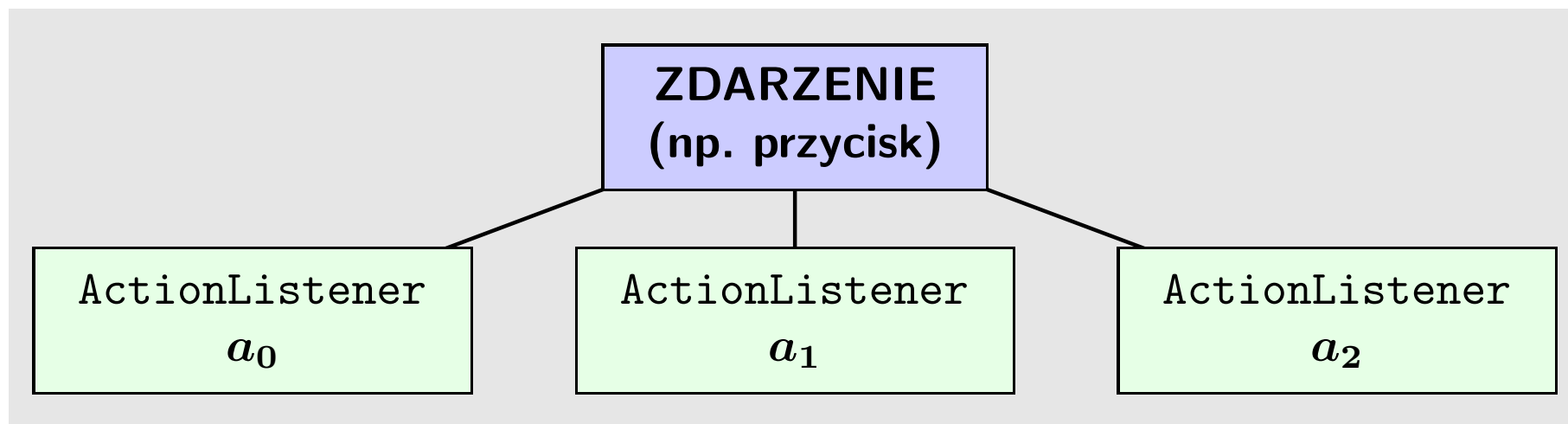


Programowanie reakcji na **zdarzenie**:

- utworzyć obiekt **reakcja** klasy implementującej `ActionListener`

Reakcja na zdarzenia

Z każdym możliwym zdarzeniem jest związana *lista „nasłuchiwczy”*, czy się zdarzyło:



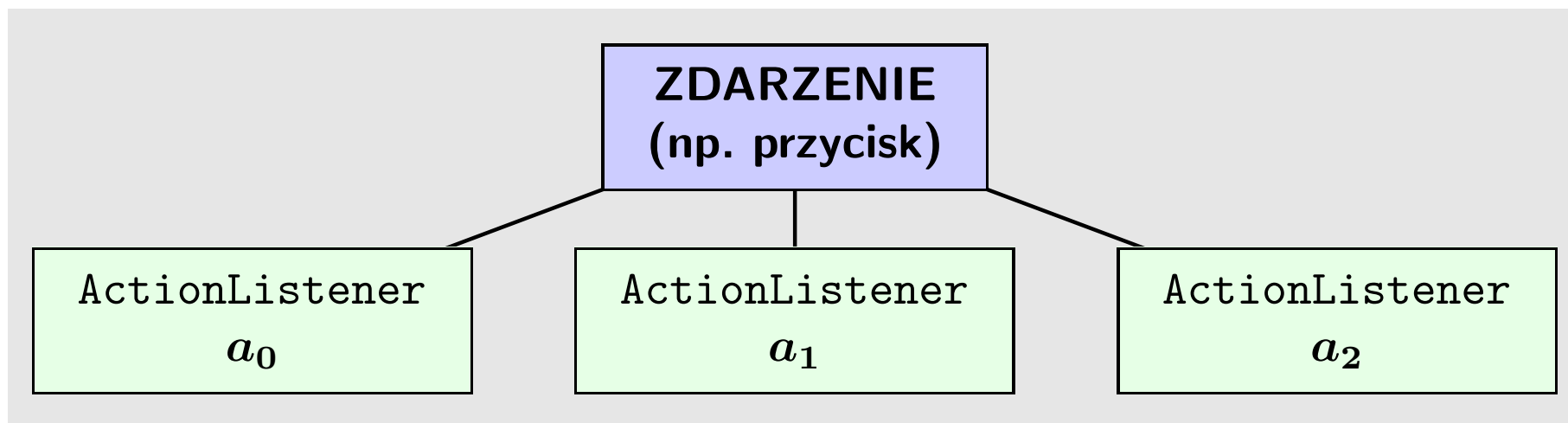
Programowanie reakcji na **zdarzenie**:

- utworzyć obiekt **reakcja** klasy implementującej `ActionListener`
- wpisać go na *listę nasłuchiwczy*:

`zdarzenie.addListener(reakcja)`

Reakcja na zdarzenia

Z każdym możliwym zdarzeniem jest związana *lista „nasłuchiaczy”*, czy się zdarzyło:

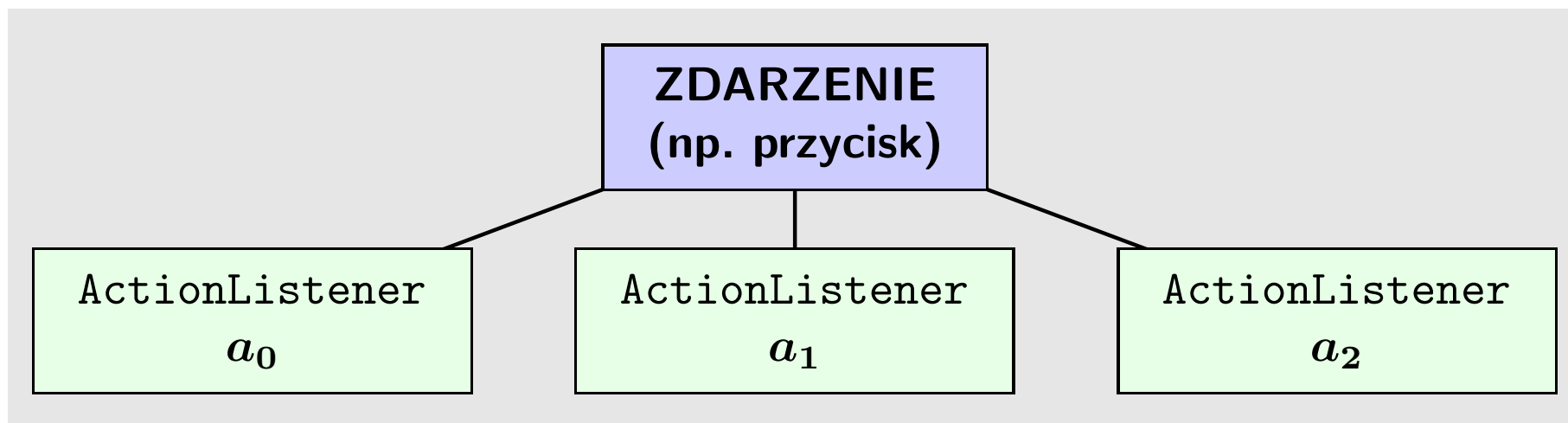


Programowanie reakcji na **zdarzenie**:

- utworzyć obiekt **reakcja** klasy implementującej **ActionListener**
- wpisać go na *listę nasłuchiaczy*:
`zdarzenie.addActionListener(reakcja)`
- wewnątrz klasy implementującej dostarczyć metodę, która ma być wywołana, kiedy zdarzenie zajdzie

Reakcja na zdarzenia

Z każdym możliwym zdarzeniem jest związana *lista „nasłuchiwczy”*, czy się zdarzyło:



Programowanie reakcji na **zdarzenie**:

- utworzyć obiekt **reakcja** klasy implementującej `ActionListener`
- wpisać go na *listę nasłuchiwczy*:
`zdarzenie.addActionListener(reakcja)`
- wewnątrz klasy implementującej dostarczyć metodę, która ma być wywołana, kiedy zdarzenie zajdzie:

```
public void actionPerformed(ActionEvent e) {  
    . . . . .  
}
```

Reakcja na zdarzenia

Przykład:

Definicja akcji:

```
class Reakcja implements ActionListener {  
    public void actionPerformed(ActionEvent klik) {  
        System.out.println("\n\nZDARZENIE klik\n");  
    }  
};
```

Reakcja na zdarzenia

Przykład:

Definicja akcji:

```
class Reakcja implements ActionListener {  
    public void actionPerformed(ActionEvent klik) {  
        System.out.println("\n\nZDARZENIE klik\n");  
    }  
};
```

Utworzenie przycisku i rejestracja akcji na liście jego nasłuchiowaczy:

```
JButton klik = new JButton("KLIKNIJ");  
klik.addActionListener(new Reakcja());
```

Reakcja na zdarzenia

Przykład:

Definicja akcji:

```
class Reakcja implements ActionListener {  
    public void actionPerformed(ActionEvent klik) {  
        System.out.println("\n\nZDARZENIE klik\n");  
    }  
};
```

Utworzenie przycisku i rejestracja akcji na liście jego nasłuchiowaczy:

```
JButton klik = new JButton("KLIKNIJ");  
klik.addActionListener(new Reakcja());
```

Dołączenie przycisku do ramki:

```
rama.add(klik);
```



Reakcja na zdarzenia

Przykład:

Definicja akcji:

```
class Reakcja implements ActionListener {  
    public void actionPerformed(ActionEvent klik) {  
        System.out.println("\n\nZDARZENIE klik\n");  
    }  
};
```

Utworzenie przycisku i rejestracja akcji na liście jego nasłuchiowaczy:

```
JButton klik = new JButton("KLIKNIJ");  
klik.addActionListener(new Reakcja());
```

Dołączenie przycisku do ramki:

```
rama.add(klik);
```



Reakcja na zdarzenia

Prościej — bez jawnego definiowania klasy Reakcja

Reakcja na zdarzenia

Prościej — bez jawnego definiowania klasy Reakcja:

Utworzenie przycisku i rejestracja akcji na liście jego nasłuchiowaczy:

```
JButton klik = new JButton("KLIKNIJ");  
klik.addActionListener(  
    new ActionListener() {  
        public void actionPerformed(ActionEvent klik) {  
            System.out.println("\n\nZDARZENIE klik\n");  
        }  
    }  
);
```

Reakcja na zdarzenia

Prościej — bez jawnego definiowania klasy Reakcja:

Utworzenie przycisku i rejestracja akcji na liście jego nasłuchiowaczy:

```
JButton klik = new JButton("KLIKNIJ");  
klik.addActionListener(  
    new ActionListener() {  
        public void actionPerformed(ActionEvent klik) {  
            System.out.println("\n\nZDARZENIE klik\n");  
        }  
    }  
);
```

Dołączenie przycisku do ramki:

```
rama.add(klik);
```



Reakcja na zdarzenia

Jeszcze prościej — przez λ

Reakcja na zdarzenia

Jeszcze prościej — przez lambdę:

Utworzenie przycisku i rejestracja akcji na liście jego nasłuchiowaczy:

```
JButton klik = new JButton("KLIKNIJ");  
klik.addActionListener(  
    e -> System.out.println("\n\nZDARZENIE klik\n")  
);
```

Reakcja na zdarzenia

Jeszcze prościej — przez lambdę:

Utworzenie przycisku i rejestracja akcji na liście jego nasłuchiowaczy:

```
JButton klik = new JButton("KLIKNIJ");  
klik.addActionListener(  
    e -> System.out.println("\n\nZDARZENIE klik\n")  
);
```

Dołączenie przycisku do ramki:

```
rama.add(klik);
```



iis.ans-elblag.pl/~stefan/Dydaktyka/Prog0biII/Prog09/P3_Przycisk.java

Reakcja na zdarzenia

Dwa różne nasłuchiwanie tego samego zdarzenia

Reakcja na zdarzenia

Dwa różne nasłuchiwanie tego samego zdarzenia:

Utworzenie przycisku i rejestracja reakcji na liście jego nasłuchiwczy:

```
JButton klik = new JButton("KLIKNIJ");  
  
klik.addActionListener(  
    e -> System.out.println("\n\nZDARZENIE 1\n")  
);
```

Reakcja na zdarzenia

Dwa różne nasłuchiwanie tego samego zdarzenia:

Utworzenie przycisku i rejestracja **2** reakcji na liście jego nasłuchiwanie:

```
JButton klik = new JButton("KLIKNIJ");

klik.addActionListener(
    e -> System.out.println("\n\nZDARZENIE 1\n")
);

klik.addActionListener(
    e -> System.out.println("\n\nZDARZENIE 2\n")
);
```

Reakcja na zdarzenia

Dwa różne nasłuchiwanie tego samego zdarzenia:

Utworzenie przycisku i rejestracja **2** reakcji na liście jego nasłuchiwczy:

```
JButton klik = new JButton("KLIKNIJ");

klik.addActionListener(
    e -> System.out.println("\n\nZDARZENIE 1\n")
);

klik.addActionListener(
    e -> System.out.println("\n\nZDARZENIE 2\n")
);
```

Dołączenie przycisku do ramki:

```
rama.add(klik);
```



Reakcja na zdarzenia

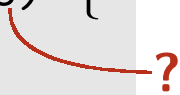
Ale do czego służy argument metody actionPerformed?

```
public void actionPerformed(ActionEvent e) {  
    . . . . .  
}
```

Reakcja na zdarzenia

Ale do czego służy argument metody actionPerformed?

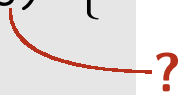
```
public void actionPerformed(ActionEvent e) {  
    . . . . .  
}
```



Reakcja na zdarzenia

Ale do czego służy argument metody `actionPerformed`?

```
public void actionPerformed(ActionEvent e) {  
    . . . . .  
}
```

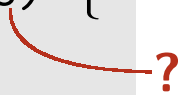


Do *callbacku*, który mamy zdefiniować, przekazywane są informacje o okolicznościach zajścia zdarzenia.

Reakcja na zdarzenia

Ale do czego służy argument metody `actionPerformed`?

```
public void actionPerformed(ActionEvent e) {  
    . . . . .  
}
```



Do *callbacku*, który mamy zdefiniować, przekazywane są informacje o okolicznościach zajścia zdarzenia.

Klasa `ActionEvent` zawiera metody, którymi możemy te informacje wydobyc.

Reakcja na zdarzenia

Ale do czego służy argument metody `actionPerformed`?

```
public void actionPerformed(ActionEvent e) {
    . . . . .
}
```



Do *callbacku*, który mamy zdefiniować, przekazywane są informacje o okolicznościach zajścia zdarzenia.

Klasa `ActionEvent` zawiera metody, którymi możemy te informacje wydobyc. Np.

`int getModifiers()` — klawisze naciśnięte przy klikaniu

...100 01100101 10010010

bit CTRL-u

bit SHIFT-u



Reakcja na zdarzenia

Ale do czego służy argument metody `actionPerformed`?

```
public void actionPerformed(ActionEvent e) {
    . . . . .
}
```



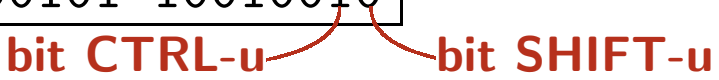
Do *callbacku*, który mamy zdefiniować, przekazywane są informacje o okolicznościach zajścia zdarzenia.

Klasa `ActionEvent` zawiera metody, którymi możemy te informacje wydobyć. Np.

`int getModifiers()` — klawisze naciśnięte przy klikaniu

...100 01100101 10010010

bit CTRL-u bit SHIFT-u



`long getWhen()` — kiedy kliknięte
w milisek. od 1 stycznia 1970 0:00:00.000 UTC

iis.ans-elblag.pl/~stefan/Dydaktyka/Prog0biII/Prog09/P5_Przycisk.java

iis.ans-elblag.pl/~stefan/Dydaktyka/Prog0biII/Prog09/P6_Przycisk.java

Reakcja na zdarzenia

Jak programować obsługę zdarzeń?
(dobre rady dziadka)



Reakcja na zdarzenia

Jak programować obsługę zdarzeń? (dobre rady dziadka)

- Przestać myśleć o biegu programu w stylu klasycznym – „najpierw dzieje się to, potem tamto”. Kolejność wykonywania zależy od zdarzeń, i programista nad nią nie panuje.



Reakcja na zdarzenia

Jak programować obsługę zdarzeń? (dobre rady dziadka)

- Przestać myśleć o biegu programu w stylu klasycznym – „najpierw dzieje się to, potem tamto”. Kolejność wykonywania zależy od zdarzeń, i programista nad nią nie panuje.
- Nie wszystko w jednym miejscu — rozbijać na klasy!



Reakcja na zdarzenia

Jak programować obsługę zdarzeń? (dobre rady dziadka)

- Przestać myśleć o biegu programu w stylu klasycznym – „najpierw dzieje się to, potem tamto”. Kolejność wykonywania zależy od zdarzeń, i programista nad nią nie panuje.
- Nie wszystko w jednym miejscu — rozbijać na klasy!
- Utworzyć klasę implementującą `ActionListener`; w jej (obowiązkowej) metodzie `actionPerformed` zaprogramować reakcję na zdarzenie.



Reakcja na zdarzenia

Jak programować obsługę zdarzeń? (dobre rady dziadka)

- Przestać myśleć o biegu programu w stylu klasycznym – „najpierw dzieje się to, potem tamto”. Kolejność wykonywania zależy od zdarzeń, i programista nad nią nie panuje.
- Nie wszystko w jednym miejscu — rozbijać na klasy!
- Utworzyć klasę implementującą `ActionListener`; w jej (obowiązkowej) metodzie `actionPerformed` zaprogramować reakcję na zdarzenie.
- Zarejestrować (przy pomocy metody `addActionListener`) obiekty generujące zdarzenia w nowym obiekcie klasy implementującej `ActionListener`.



Reakcja na zdarzenia

Jak programować obsługę zdarzeń? (dobre rady dziadka)

- Przestać myśleć o biegu programu w stylu klasycznym – „najpierw dzieje się to, potem tamto”. Kolejność wykonywania zależy od zdarzeń, i programista nad nią nie panuje.
- Nie wszystko w jednym miejscu — rozbijać na klasy!
- Utworzyć klasę implementującą `ActionListener`; w jej (obowiązkowej) metodzie `actionPerformed` zaprogramować reakcję na zdarzenie.
- Zarejestrować (przy pomocy metody `addActionListener`) obiekty generujące zdarzenia w nowym obiekcie klasy implementującej `ActionListener`.



Reakcja na zdarzenia

Przykład: (włączanie światła przyciskiem)

Reakcja na zdarzenia

Przykład: (włączanie światła przyciskiem)
Żarówka — jak poprzednio.

Reakcja na zdarzenia

Przykład: (włączanie światła przyciskiem)
Żarówka — jak poprzednio.

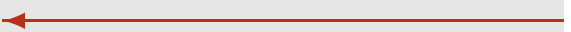
Włącznik trzeba oprogramować:

```
class Wlacznik implements ActionListener {  
    private JFrame rama;  
  
    public Wlacznik (JFrame rama) { this.rama = rama; }  
  
    public void actionPerformed(ActionEvent e) {  
        rama.getContentPane().removeAll();  
        rama.add(new Zarowka()); rama.validate();  
    }  
}
```

Reakcja na zdarzenia

Przykład: (włączanie światła przyciskiem)
Żarówka — jak poprzednio.

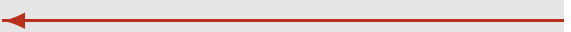
Włącznik trzeba oprogramować:

```
class Wlacznik implements ActionListener {  
    private JFrame rama;  w której jest  
żarówka  
  
    public Wlacznik (JFrame rama) { this.rama = rama; }  
  
    public void actionPerformed(ActionEvent e) {  
        rama.getContentPane().removeAll();  
        rama.add(new Zarowka()); rama.validate();  
    }  
}
```

Reakcja na zdarzenia

Przykład: (włączanie światła przyciskiem)
Żarówka — jak poprzednio.

Włącznik trzeba oprogramować:

```
class Wlacznik implements ActionListener {  
    private JFrame rama;  w której jest  
żarówka  
  
    public Wlacznik (JFrame rama) { this.rama = rama; }  
  
    public void actionPerformed(ActionEvent e) {  
        rama.getContentPane().removeAll();  
        rama.add(new Zarowka()); rama.validate();  
    }  
}
```

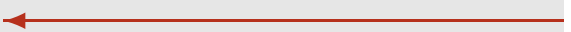
Rejestracja nasłuchiwanego zdarzenia przy przycisku:

```
JButton przycisk = new JButton("wlacznik");  
przycisk.addActionListener(new Wlacznik(rama));
```

Reakcja na zdarzenia

Przykład: (włączanie światła przyciskiem)
Żarówka — jak poprzednio.

Włącznik trzeba oprogramować:

```
class Wlacznik implements ActionListener {  
    private JFrame rama;  w której jest  
żarówka  
  
    public Wlacznik (JFrame rama) { this.rama = rama; }  
  
    public void actionPerformed(ActionEvent e) {  
        rama.getContentPane().removeAll();  
        rama.add(new Zarowka()); rama.validate();  
    }  
}
```

Rejestracja nasłuchiwanego zdarzenia przy przycisku:

```
JButton przycisk = new JButton("wlacznik");  
przycisk.addActionListener(new Wlacznik(rama));
```

Reakcja na zdarzenia

Przykład: kalkulator

Po każdym naciśnięciu klawisza uaktualnia dwie liczby:

- liczbę nowo pisaną,
- sumę dotychczas zsumowanych liczb.

9 7	10 8	11 9
6 4	7 5	8 6
3 1	4 2	5 3
0 C	1 +	2 0

Reakcja na zdarzenia

Przykład: kalkulator

Po każdym naciśnięciu klawisza uaktualnia dwie liczby:

- liczbę nowo pisaną,
- sumę dotychczas zsumowanych liczb.

Np. tak:

klawisz	wyświetla	
	suma	nowa
5	0	5

9 7	10 8	11 9
6 4	7 5	8 6
3 1	4 2	5 3
0 C	1 +	2 0

Reakcja na zdarzenia

Przykład: kalkulator

Po każdym naciśnięciu klawisza uaktualnia dwie liczby:

- liczbę nowo pisaną,
- sumę dotychczas zsumowanych liczb.

Np. tak:

klawisz	wyświetla	
	suma	nowa
5	0	5
7	0	57

9 7	10 8	11 9
6 4	7 5	8 6
3 1	4 2	5 3
0 C	1 +	2 0

Reakcja na zdarzenia

Przykład: kalkulator

Po każdym naciśnięciu klawisza uaktualnia dwie liczby:

- liczbę nowo pisaną,
- sumę dotychczas zsumowanych liczb.

Np. tak:

klawisz	wyświetla	
	suma	nowa
5	0	5
7	0	57
+	57	0

9 7	10 8	11 9
6 4	7 5	8 6
3 1	4 2	5 3
0 C	1 +	2 0

Reakcja na zdarzenia

Przykład: kalkulator

Po każdym naciśnięciu klawisza uaktualnia dwie liczby:

- liczbę nowo pisaną,
- sumę dotychczas zsumowanych liczb.

Np. tak:

klawisz	wyświetla	
	suma	nowa
5	0	5
7	0	57
+	57	0
4	57	4

9 7	10 8	11 9
6 4	7 5	8 6
3 1	4 2	5 3
0 C	1 +	2 0

Reakcja na zdarzenia

Przykład: kalkulator

Po każdym naciśnięciu klawisza uaktualnia dwie liczby:

- liczbę nowo pisaną,
- sumę dotychczas zsumowanych liczb.

Np. tak:

klawisz	wyświetla	
	suma	nowa
5	0	5
7	0	57
+	57	0
4	57	4
4	57	44

9 7	10 8	11 9
6 4	7 5	8 6
3 1	4 2	5 3
0 C	1 +	2 0

Reakcja na zdarzenia

Przykład: kalkulator

Po każdym naciśnięciu klawisza uaktualnia dwie liczby:

- liczbę nowo pisaną,
- sumę dotychczas zsumowanych liczb.

Np. tak:

klawisz	wyświetla	
	suma	nowa
5	0	5
7	0	57
+	57	0
4	57	4
4	57	44
+	101	0

9 7	10 8	11 9
6 4	7 5	8 6
3 1	4 2	5 3
0 C	1 +	2 0

Reakcja na zdarzenia

Przykład: kalkulator

Po każdym naciśnięciu klawisza uaktualnia dwie liczby:

- liczbę nowo pisaną,
- sumę dotychczas zsumowanych liczb.

Np. tak:

klawisz	wyświetla	
	suma	nowa
5	0	5
7	0	57
+	57	0
4	57	4
4	57	44
+	101	0
C	0	0

9 7	10 8	11 9
6 4	7 5	8 6
3 1	4 2	5 3
0 C	1 +	2 0

Reakcja na zdarzenia

Zdarzenia: naciskanie klawiszy. Każdy klawisz musi mieć swoją akcję (implementującą `ActionListener`)

Reakcja na zdarzenia

Zdarzenia: naciskanie klawiszy. Każdy klawisz musi mieć swoją akcję (implementującą `ActionListener`); te akcje mogą być obiektami tej samej klasy:

```
class Akcja implements ActionListener {  
    private int nr;  
    public Akcja(int n) { nr = n; }  
    public void actionPerformed(ActionEvent e) { ??? }  
}
```

Reakcja na zdarzenia

Zdarzenia: naciskanie klawiszy. Każdy klawisz musi mieć swoją akcję (implementującą `ActionListener`); te akcje mogą być obiektami tej samej klasy:

```
class Akcja implements ActionListener {  
    private int nr;      ← numer klawisza  
    public Akcja(int n) { nr = n; }  
    public void actionPerformed(ActionEvent e) { ??? }  
}
```

Reakcja na zdarzenia

Zdarzenia: naciskanie klawiszy. Każdy klawisz musi mieć swoją akcję (implementującą `ActionListener`); te akcje mogą być obiektami tej samej klasy:

```
class Akcja implements ActionListener {  
    private int nr;      ← numer klawisza  
    public Akcja(int n) { nr = n; } ← konstruktor  
    public void actionPerformed(ActionEvent e) { ??? }  
}
```

Reakcja na zdarzenia

Zdarzenia: naciskanie klawiszy. Każdy klawisz musi mieć swoją akcję (implementującą `ActionListener`); te akcje mogą być obiektami tej samej klasy:

```
class Akcja implements ActionListener {  
    private int nr;      ← numer klawisza  
    public Akcja(int n) { nr = n; } ← konstruktor  
    public void actionPerformed(ActionEvent e) { ??? }  
}
```

↑
obsługa naciśnięcia
klawisza

Reakcja na zdarzenia

Zdarzenia: naciskanie klawiszy. Każdy klawisz musi mieć swoją akcję (implementującą `ActionListener`); te akcje mogą być obiektami tej samej klasy:

```
class Akcja implements ActionListener {  
    private int nr;      ← numer klawisza  
    public Akcja(int n) { nr = n; } ← konstruktor  
    public void actionPerformed(ActionEvent e) { ??? }  
}
```

↑
obsługa naciśnięcia
klawisza

Potrzebne dwie zmienne **statyczne** `suma` i `nowa`.

Reakcja na zdarzenia

Zdarzenia: naciskanie klawiszy. Każdy klawisz musi mieć swoją akcję (implementującą `ActionListener`); te akcje mogą być obiektami tej samej klasy:

```
class Akcja implements ActionListener {  
    private int nr;      ← numer klawisza  
    public Akcja(int n) { nr = n; } ← konstruktor  
    public void actionPerformed(ActionEvent e) { ??? }  
}
```

↑
obsługa naciśnięcia
klawisza

Potrzebne dwie zmienne **statyczne** `suma` i `nowa`. Naciśnięcie klawisza ma je zmieniać:

- **klawisz C** — obie zerować

Reakcja na zdarzenia

Zdarzenia: naciskanie klawiszy. Każdy klawisz musi mieć swoją akcję (implementującą `ActionListener`); te akcje mogą być obiektami tej samej klasy:

```
class Akcja implements ActionListener {  
    private int nr;      ← numer klawisza  
    public Akcja(int n) { nr = n; } ← konstruktor  
    public void actionPerformed(ActionEvent e) { ??? }  
}
```

↑
obsługa naciśnięcia
klawisza

Potrzebne dwie zmienne **statyczne** `suma` i `nowa`. Naciśnięcie klawisza ma je zmieniać:

- klawisz **C** — obie zerować
- klawisz **+** — `nowa` dodawać do `suma`, `nowa` zerować

Reakcja na zdarzenia

Zdarzenia: naciskanie klawiszy. Każdy klawisz musi mieć swoją akcję (implementującą `ActionListener`); te akcje mogą być obiektami tej samej klasy:

```
class Akcja implements ActionListener {
    private int nr;
    public Akcja(int n) { nr = n; }
    public void actionPerformed(ActionEvent e) { ??? }
}
```

numer klawisza
konstruktor
obsługa naciśnięcia klawisza

Potrzebne dwie zmienne **statyczne** `suma` i `nowa`. Naciśnięcie klawisza ma je zmieniać:

- klawisz **C** — obie zerować
- klawisz **+** — `nowa` dodawać do `suma`, `nowa` zerować
- klawisz **cyfra** — `nowa` mnożyć przez 10 i dodawać cyfrę

Reakcja na zdarzenia

Zdarzenia: naciskanie klawiszy. Każdy klawisz musi mieć swoją akcję (implementującą `ActionListener`); te akcje mogą być obiektami tej samej klasy:

```
class Akcja implements ActionListener {
    private int nr;
    public Akcja(int n) { nr = n; }
    public void actionPerformed(ActionEvent e) { ??? }
}
```

numer klawisza
konstruktor
obsługa naciśnięcia klawisza

Potrzebne dwie zmienne **statyczne** `suma` i `nowa`. Naciśnięcie klawisza ma je zmieniać:

- **klawisz C** — obie zerować
- **klawisz +** — `nowa` dodawać do `suma`, `nowa` zerować
- **klawisz cyfra** — `nowa` mnożyć przez 10 i dodawać cyfrę

Następnie należy wyświetlać obie zmienne.

Reakcja na zdarzenia

iis.ans-elblag.pl/~stefan/Dydaktyka/Prog0biII/Prog09/P8_Kalkulator.java

Reakcja na zdarzenia

iis.ans-elblag.pl/~stefan/Dydaktyka/Prog0biII/Prog09/P8_Kalkulator.java

W jakiejś innej klasie:

- utworzyć napisy (`JLabel`) do umieszczenia w oknie

Reakcja na zdarzenia

iis.ans-elblag.pl/~stefan/Dydaktyka/Prog0biII/Prog09/P8_Kalkulator.java

W jakiejś innej klasie:

- utworzyć napisy (`JLabel`) do umieszczenia w oknie
 - `SUMA: ...`
 - `nowa: ...`

Reakcja na zdarzenia

iis.ans-elblag.pl/~stefan/Dydaktyka/Prog0biII/Prog09/P8_Kalkulator.java

W jakiejś innej klasie:

- utworzyć napisy (`JLabel`) do umieszczenia w oknie
 - `SUMA: ...`
 - `nowa: ...`
- w pętli po `nr` od 0 do 11:
 - utworzyć klawisz `nr`

Reakcja na zdarzenia

iis.ans-elblag.pl/~stefan/Dydaktyka/Prog0biII/Prog08/P8_Kalkulator.java

W jakiejś innej klasie:

- utworzyć napisy (`JLabel`) do umieszczenia w oknie
 - `SUMA: ...`
 - `nowa: ...`
- w pętli po `nr` od 0 do 11:
 - utworzyć klawisz `nr`
 - utworzyć nasłuchiwanca tego klawisza:
`klawisz[nr].addActionListener(new Akcja(nr));`

Reakcja na zdarzenia

iis.ans-elblag.pl/~stefan/Dydaktyka/Prog0biII/Prog08/P8_Kalkulator.java

W jakiejś innej klasie:

- utworzyć napisy (`JLabel`) do umieszczenia w oknie
 - `SUMA: ...`
 - `nowa: ...`
- w pętli po `nr` od 0 do 11:
 - utworzyć klawisz `nr`
 - utworzyć nasłuchiwanca tego klawisza:
`klawisz[nr].addActionListener(new Akcja(nr));`
- utworzyć panel (`JPanel`)

Reakcja na zdarzenia

iis.ans-elblag.pl/~stefan/Dydaktyka/Prog0biII/Prog08/P8_Kalkulator.java

W jakiejś innej klasie:

- utworzyć napisy (`JLabel`) do umieszczenia w oknie
 - `SUMA: ...`
 - `nowa: ...`
- w pętli po `nr` od 0 do 11:
 - utworzyć klawisz `nr`
 - utworzyć nasłuchiacza tego klawisza:
`klawisz[nr].addActionListener(new Akcja(nr));`
- utworzyć panel (`JPanel`)
 - dodać do niego klawisze

Reakcja na zdarzenia

iis.ans-elblag.pl/~stefan/Dydaktyka/Prog0biII/Prog08/P8_Kalkulator.java

W jakiejś innej klasie:

- utworzyć napisy (`JLabel`) do umieszczenia w oknie
 - `SUMA: ...`
 - `nowa: ...`
- w pętli po `nr` od 0 do 11:
 - utworzyć klawisz `nr`
 - utworzyć nasłuchiwanca tego klawisza:
`klawisz[nr].addActionListener(new Akcja(nr));`
- utworzyć panel (`JPanel`)
 - dodać do niego klawisze
 - dodać do niego napisy

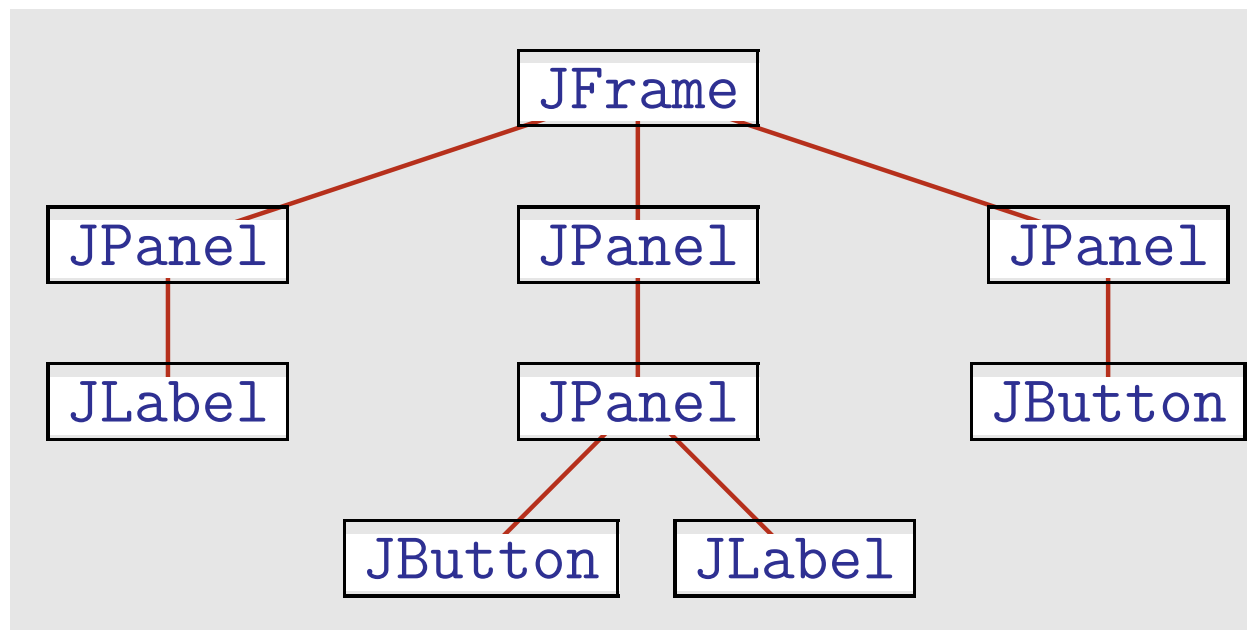
Reakcja na zdarzenia

iis.ans-elblag.pl/~stefan/Dydaktyka/Prog0biII/Prog08/P8_Kalkulator.java

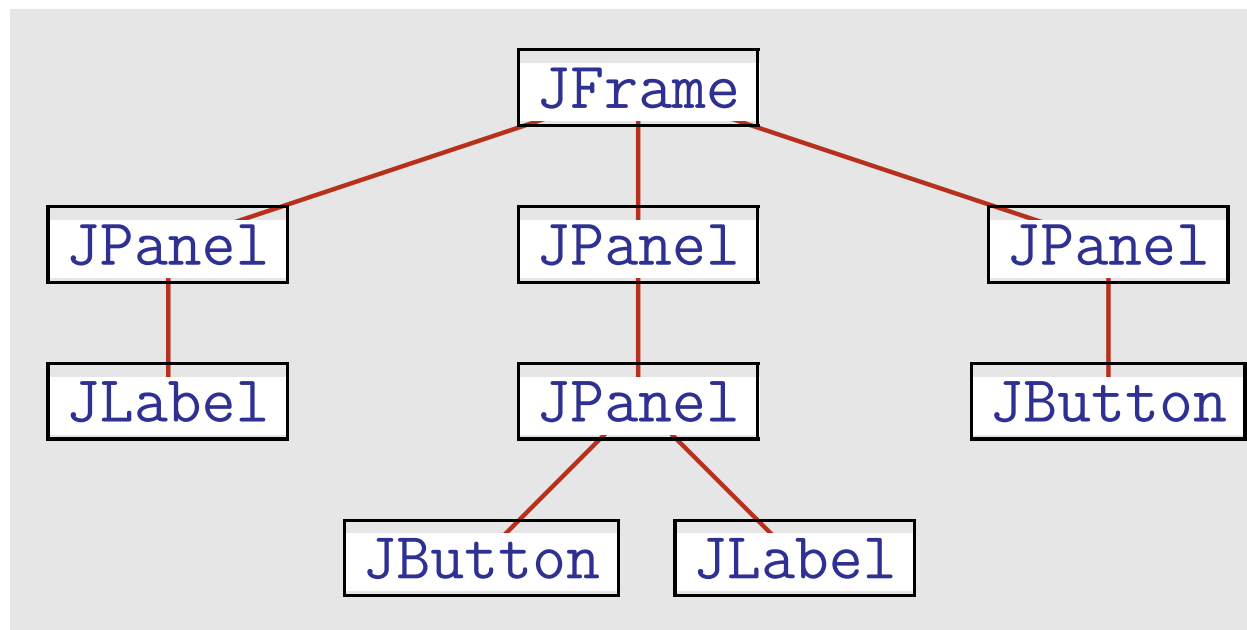
W jakiejś innej klasie:

- utworzyć napisy (`JLabel`) do umieszczenia w oknie
 - `SUMA: ...`
 - `nowa: ...`
- w pętli po `nr` od 0 do 11:
 - utworzyć klawisz `nr`
 - utworzyć nasłuchiwanca tego klawisza:
`klawisz[nr].addActionListener(new Akcja(nr));`
- utworzyć panel (`JPanel`)
 - dodać do niego klawisze
 - dodać do niego napisy
- utworzyć okno (`JFrame`) i włożyć do niego panel

Przykładowa struktura okna

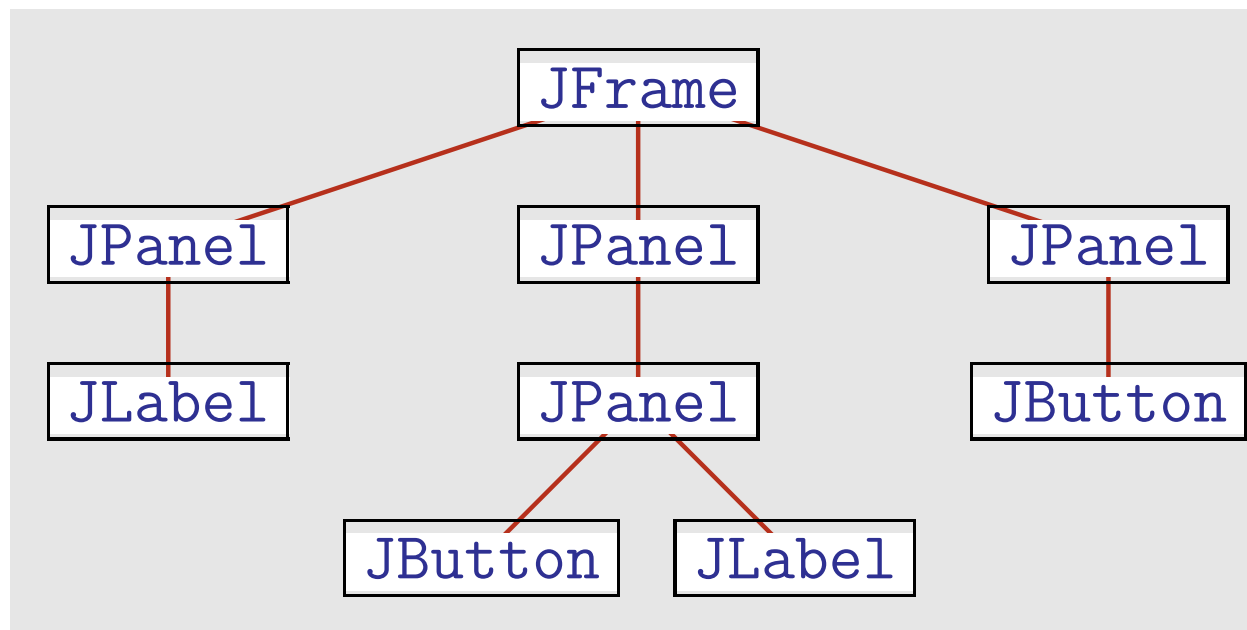


Przykładowa struktura okna



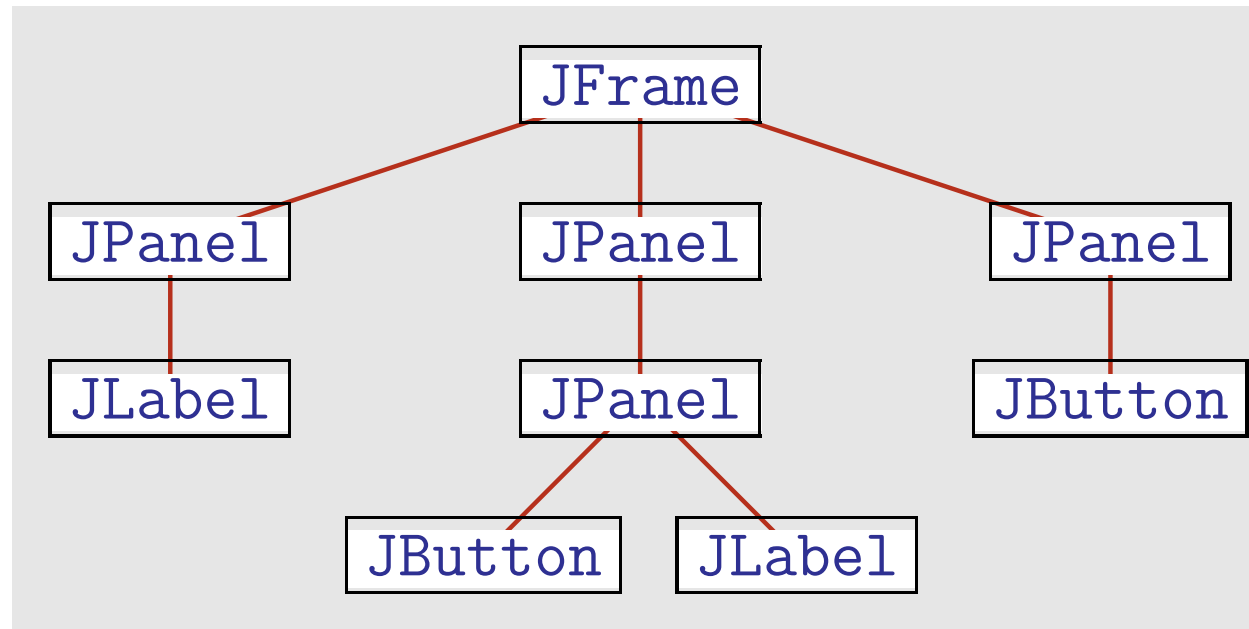
- `JFrame` i `JPanel` to t.zw. *kontenery* — można do nich wkładać różne rzeczy

Przykładowa struktura okna



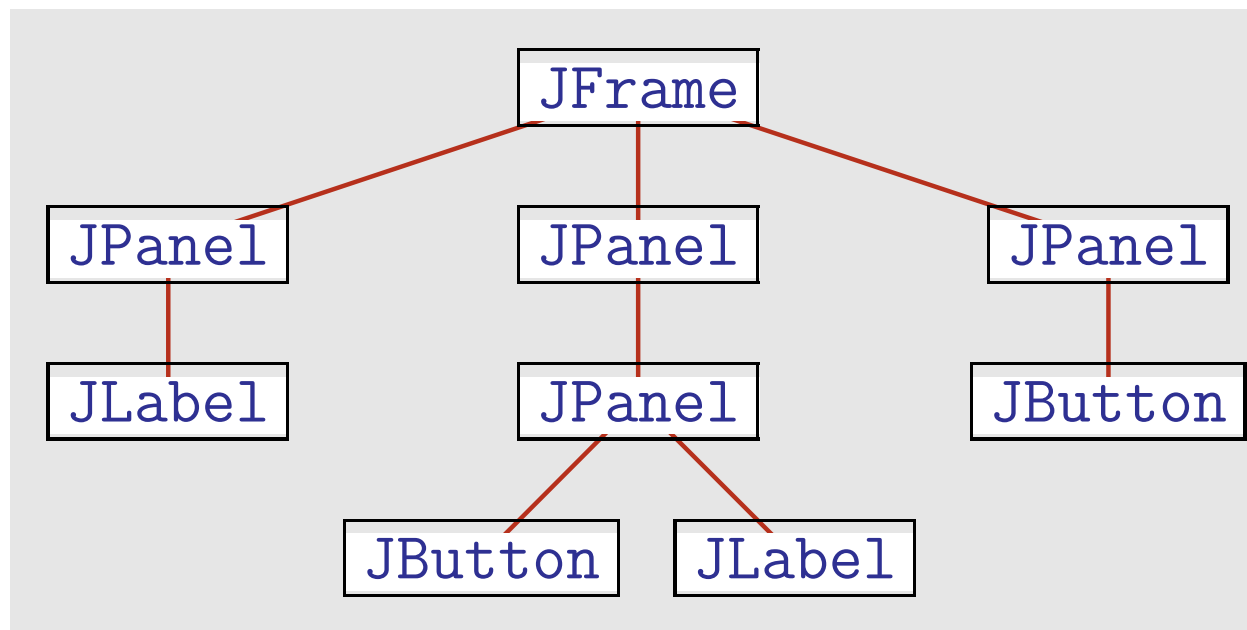
- `JFrame` i `JPanel` to t.zw. *kontenery* — można do nich wkładać różne rzeczy
- do `JPanel` można wkładać również mniejsze `JPanele`

Przykładowa struktura okna



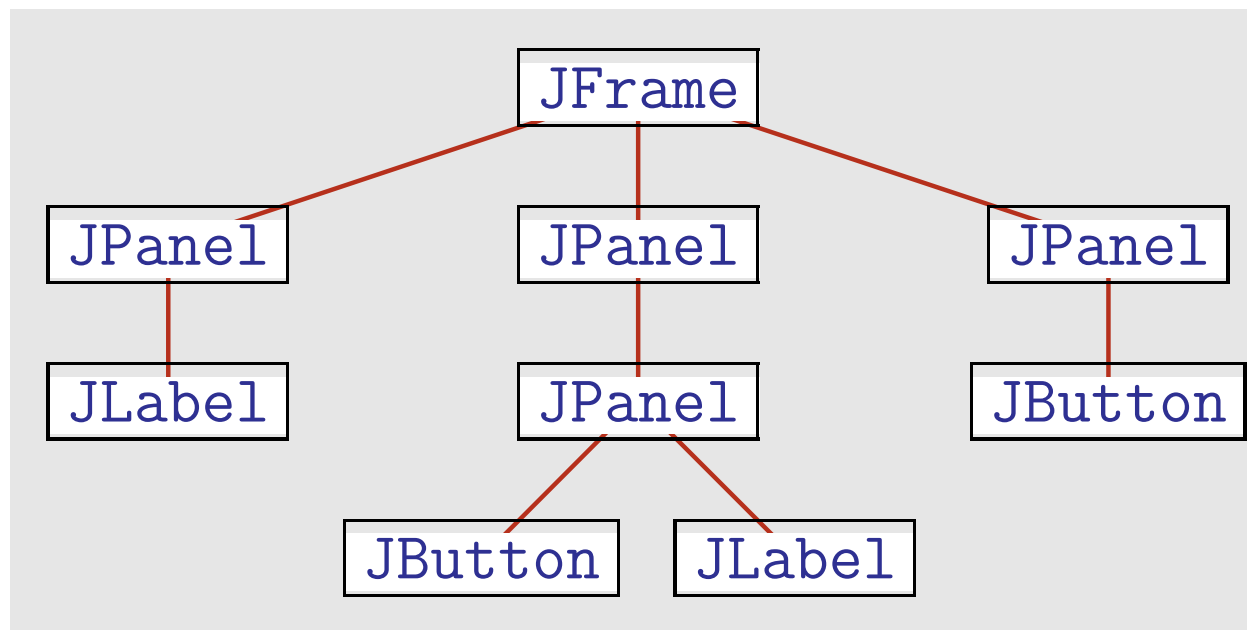
- `JFrame` i `JPanel` to t.zw. *kontenery* — można do nich wkładać różne rzeczy
- do `JPanel` można wkładać również mniejsze `JPanele`
- do kontenerów można wkładać na przykład `JButton` (przycisk)

Przykładowa struktura okna



- `JFrame` i `JPanel` to t.zw. *kontenery* — można do nich wkładać różne rzeczy
- do `JPanel` można wkładać również mniejsze `JPanele`
- do kontenerów można wkładać na przykład `JButton` (przycisk) i `JLabel` (napis)

Przykładowa struktura okna



- `JFrame` i `JPanel` to t.zw. *kontenery* — można do nich wkładać różne rzeczy
- do `JPanel` można wkładać również mniejsze `JPanele`
- do kontenerów można wkładać na przykład `JButton` (przycisk) i `JLabel` (napis) oraz jeszcze inne