

Stefan Sokołowski

PROGRAMOWANIE OBIEKTOWE II

wykład 11

Inst. Informatyki Stosowanej, ANS
Elbląg, 2024/2025

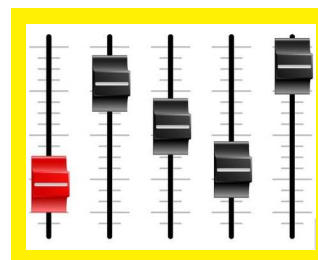
Wykład 11, str. 1

Suwak w oknie

iis.ans-elblag.pl/~stefan/Dydaktyka/ProgObiII/Prog11/P1_Suwaki.java

Suwak jest obiektem klasy `JSlider`,
nasłuchiwcem suwaków jest

```
interface ChangeListener {  
    void stateChanged(ChangeEvent ev);  
}
```



Suwak rozjaśniający/ściemniający tło:

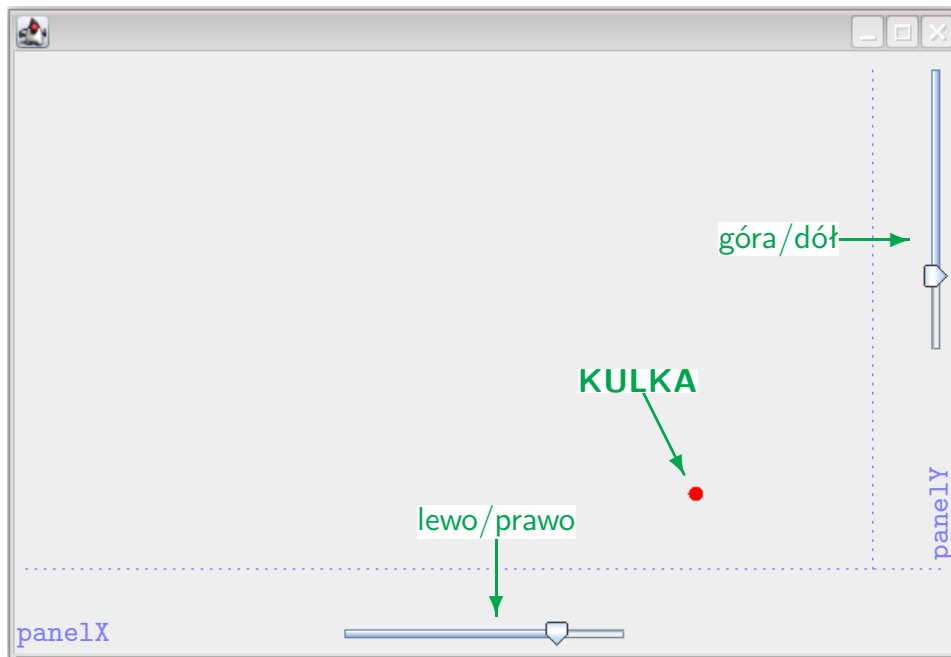
```
class MojSuwak implements ChangeListener {  
    private JSlider suwak; private JPanel panel;  
    public MojSuwak(JSlider suwak, JPanel panel) {  
        this.suwak = suwak; this.panel = panel;  
    }  
    public void stateChanged(ChangeEvent ev) {  
        float v = (float)suwak.getValue()/100;  
        panel.setBackground(new Color(v,v,v));  
        panel.revalidate();  
    }  
}
```

konstruktor

metoda
implementująca

iis.ans-elblag.pl/~stefan/Dydaktyka/ProgObiII/Prog11/P2_Suwaki.java

Przesuwanie kulki suwakami



iis.ans-elblag.pl/~stefan/Dydaktyka/ProgObiII/Prog11/P3_Wspolrzedne.java

Przesuwanie kulki suwakami

Rysowanie kulki w ustalonym miejscu:

```
class Rys extends JPanel {
    private int wspX, wspY;

    public Rys(int wspX, int wspY) {
        this.wspX = wspX; this.wspY = wspY; ← konstruktor
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g); ← czyszczenie panelu
        g.setColor(Color.RED);
        g.fillOval(wspX-5, wspY-5, 10,10);
    }
}
```

```
new Rys(..., ...)
```

← współrzędne kulki

Przesuwanie kulki suwakami

Obsługa suwaków:

```
class MojSuwak implements ChangeListener {
    static private int wspX=300, wspY=200;
    private int ktory;
    private JSlider suwak;
    private JFrame okno;

    public MojSuwak(int ktory, JSlider suwak, JFrame okno) {
        this.ktory = ktory; this.suwak = suwak; this.okno = okno;
    }

    public void stateChanged(ChangeEvent ev) {
        if (ktory == 0) wspX = suwak.getValue()*6;
        else wspY = suwak.getValue()*4;
        okno.add(new Rys(wspX,wspY), BorderLayout.CENTER);
        okno.revalidate();
    }
}
```

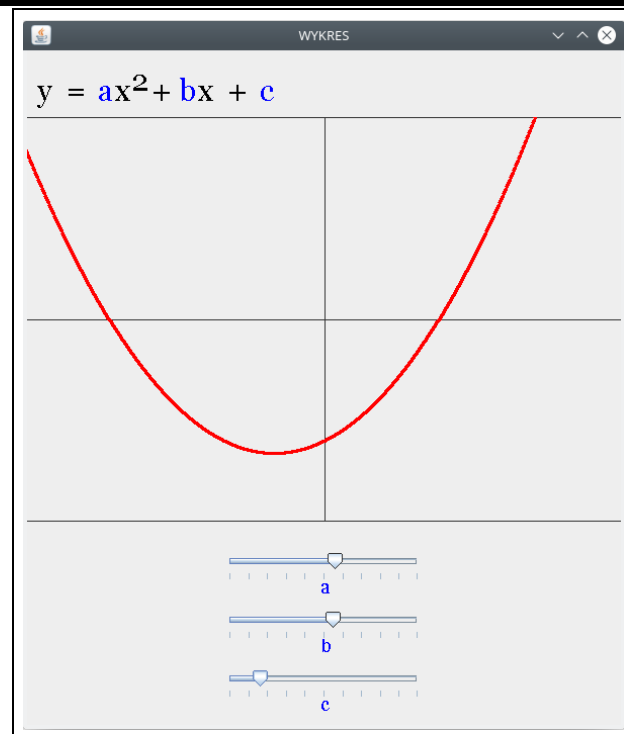
konstruktor

Rodzaje suwaków



zwykły:	<code>suwak = new JSlider();</code>
z podziałką:	<code>suwak = new JSlider();</code> <code>suwak.setPaintTicks(true);</code> <code>suwak.setMajorTickSpacing(20);</code> <code>suwak.setMinorTickSpacing(5);</code> <code>suwak.setPaintLabels(true);</code>
odwrotny kierunek:	<code>suwak = new JSlider();</code> <code>suwak.setInverted(true);</code>
poziomy (domyślny):	<code>suwak = new JSlider(SwingConstants.HORIZONTAL);</code>
pionowy:	<code>suwak = new JSlider(SwingConstants.VERTICAL);</code>
z minimum, maximum i wartością początkową:	<code>suwak =</code> <code>new JSlider(int min, int max, int initVal);</code>

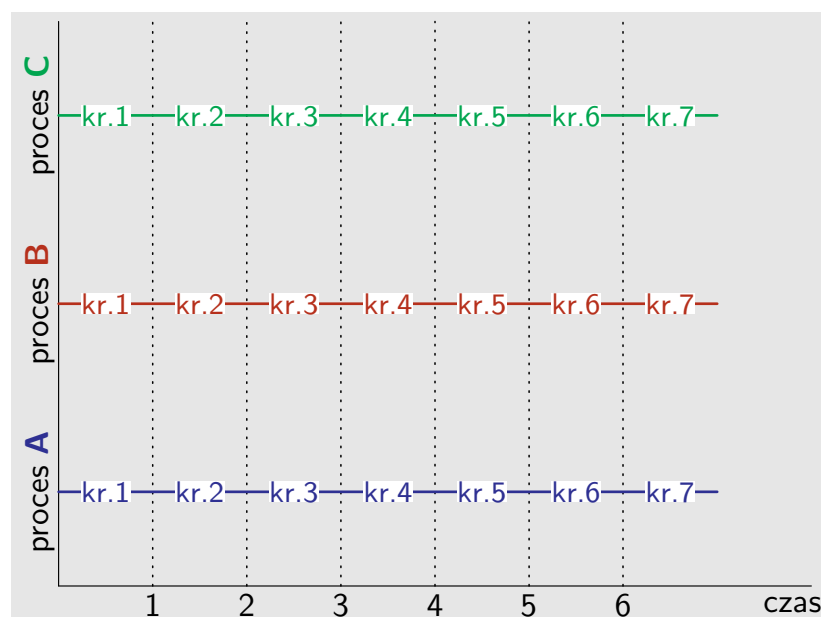
Parametry funkcji wyznaczone suwakami



iis.ans-elblag.pl/~stefan/Dydaktyka/ProgObiII/Prog11/P4-Wykres.java

Wielowątkowość — (udawana) współbieżność

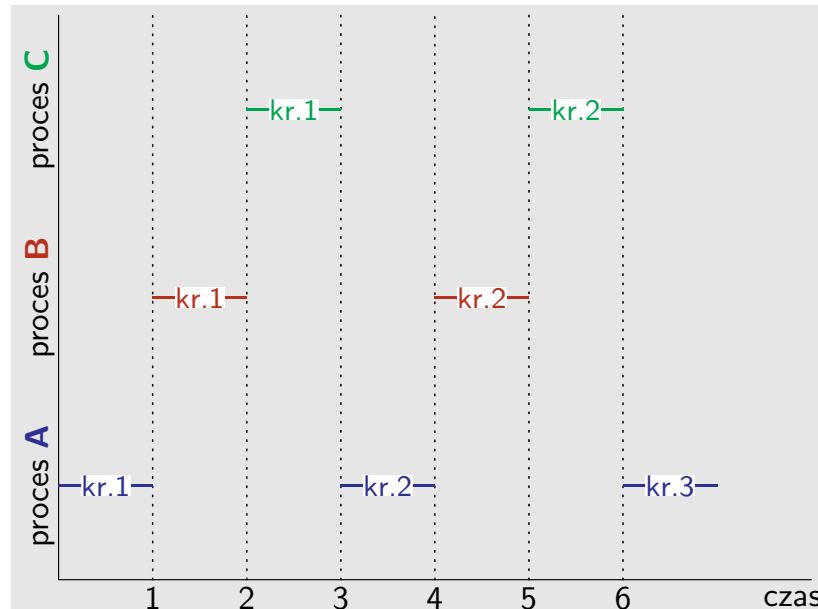
Prawdziwa współbieżność:



Kroki kilku procesów wykonują się **jednocześnie**; obsługiwane przez różne procesory.

Wielowątkowość — (udawana) współbieżność

Udawana współbieżność (przeplot, podział czasu):



Kroki kilku procesów wykonują się **na przemian**; obsługiwane przez ten sam procesor.

Wielowątkowość — (udawana) współbieżność

Jeśli procesor jest wystarczająco szybki a procesy nie wpływają na siebie wzajemnie, to udawana współbieżność może niewiele różnić się od prawdziwej.

W Javie można zaprogramować współbieżne wykonanie wielu procesów. Jeśli dostępnych procesorów jest zbyt mało na prawdziwą współbieżność, to zostaje zastosowany podział czasu.

Terminologia:

Anglosasom wszystko kojarzy się z krawiectwem:

boleć	—	pin	(= szpilka)
napis	—	String	(= sznurek, tasiemka)
wątek	—	thread	(= nitka)

wątek — program dla jednego „wykonawcy” (procesora)

proces — wątek samodzielny, nie interagujący z innymi (proces może mieć kilka wątków)

Wielowątkowość — (udawana) współbieżność

Wątek trzeba najpierw zaprogramować:

```
interface Runnable {
    void run();
}
```

Implementujemy interface `Runnable` z metodą `run` — jej ciałem ma być program pojedynczego wątku.

```
class Watek implements Runnable {
    public void run() {
        . . . kolejne kroki . . .
    }
}
```

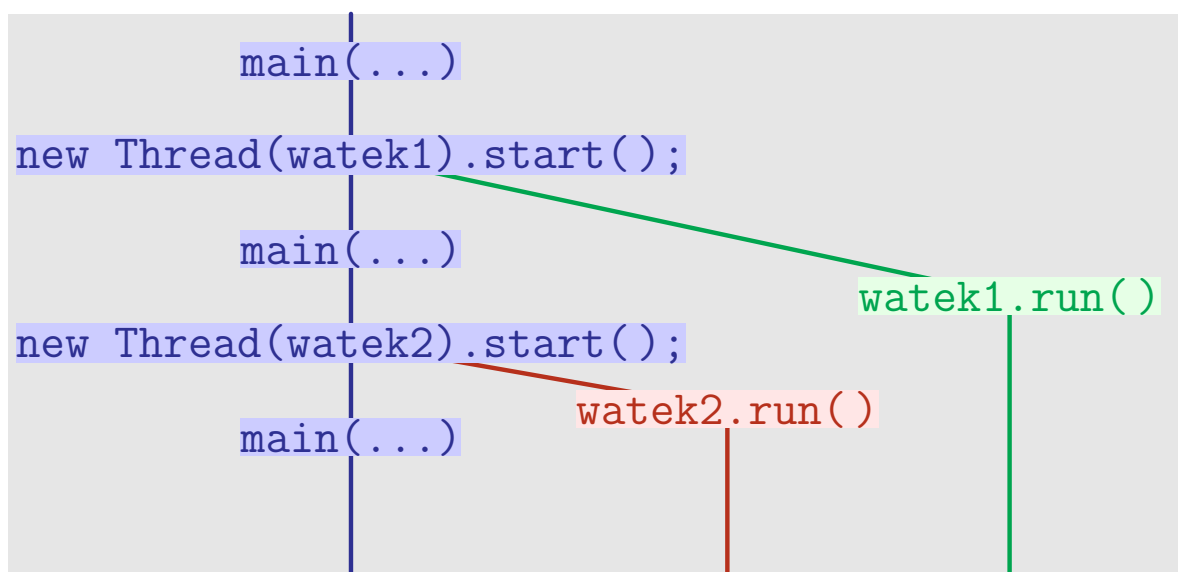
Następnie utworzyć wykonanie tego wątku, i wystartować je:

```
Watek watek = new Watek();
Thread thr = new Thread(watek);
thr.start();
```

iis.ans-elblag.pl/~stefan/Dydaktyka/ProgObiII/Prog11/W0_Watki.java

Wielowątkowość — (udawana) współbieżność

Metoda `start()` obiektu klasy `Thread` powoduje „odgałęzienie” nowego wątku:



Odtąd nowy wątek wykonuje się współbieżnie (lub z podziałem czasu) z wątkiem głównym.

Wielowątkowość: odbijające się kulki

iis.ans-elblag.pl/~stefan/Dydaktyka/ProgObiII/Prog11/W1_Kulki.java

Program tworzy okno z dwoma przyciskami, wyrzucającymi kulki na pole: czerwone i zielone. Kulki poruszają się po polu i odbijają się od jego brzegów.

Każda kulka prowadzona jest przez osobno liczący wątek, tworzony przez przycisk.

Program wykorzystuje m.in.

- klasę abstrakcyjną `Graphics` (rysowanie/ścieranie w panelu),
- interface `ActionListener` (obsługa przycisków),
- interface `Runnable` (tworzenie wątków sterujących kulkami).



Wielowątkowość — do czego potrzebna?

Na przykład:

- obsługa zewnętrznych żądań — każde powinno mieć swój wątek;
- zadania niezależne o trudnym do przewidzenia czasie działania; np. ściąganie z internetu wielu stron;
- obliczenia, które można podzielić na mniejsze fragmenty — dla wykorzystania kilku procesorów (lub rdzeni);
- animacje i gry;
- przyspieszanie działania;
- itd.

Wielowątkowość — przyspieszanie

Sortujemy dwie tablice

- albo zwyczajnie, najpierw jedną a potem drugą,
- albo jednocześnie, dwoma różnymi wątkami.

iis.ans-elblag.pl/~stefan/Dydaktyka/ProgObiII/Prog11/W2.Pomiar/

ZWYCZAJNIE

```
public void sort() {
    for (int i=ileLiczby; i>1; i--)
        for (int j=1; j<i; j++)
            if (tab[j-1]>tab[j]) {
                double pom = tab[j-1];
                tab[j-1] = tab[j]; tab[j] = pom;
            }
}

...

new Babelki(1, liczby1)
    .sort();
new Babelki(2, liczby2)
    .sort();
```

WSPÓŁBIEŻNIE

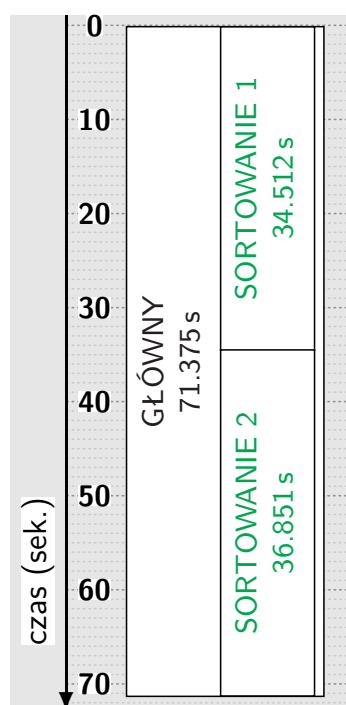
```
public void run() {
    for (int i=ileLiczby; i>1; i--)
        for (int j=1; j<i; j++)
            if (tab[j-1]>tab[j]) {
                double pom = tab[j-1];
                tab[j-1] = tab[j]; tab[j] = pom;
            }
}

...

new Thread(new Babelki(1, liczby1))
    .start();
new Thread(new Babelki(2, liczby2))
    .start();
```

Wielowątkowość — przyspieszanie

sortowanie 2 tablic po 100 000 liczb rzeczywistych

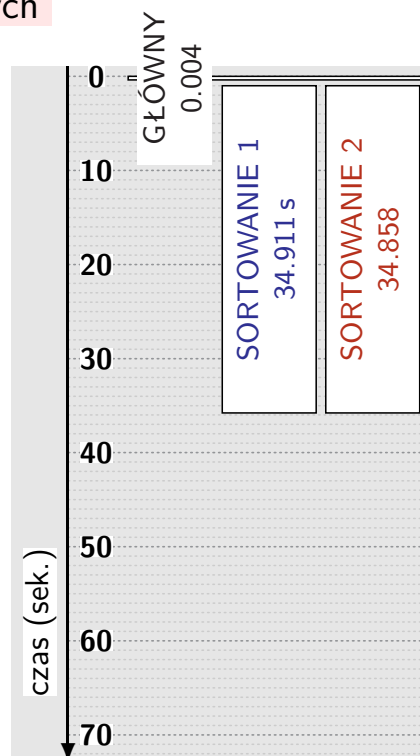


Dwa sortowania, jedno po drugim:

sortowanie 1	— 34.512 sek.
sortowanie 2	— 36.851 sek.
główny	— 71.375 sek.
ŁĄCZNIE	— 71.375 sek.

Dwa sortowania, współbieżnie:

sortowanie 1	— 34.911 sek.
sortowanie 2	— 34.858 sek.
główny	— 0.004 sek.
ŁĄCZNIE	— 35.897 sek.



Wnioski z eksperymentu

- Dwa wątki działały współbieżnie. Skutek: **skrócenie czasu** wykonania zadania.
- Konkretnie czasy zależą od okoliczności **poza kontrolą** programu, takich jak:
 - liczba rdzeni w procesorze,
 - inne programy działające jednocześnie w komputerze (np. system operacyjny, obsługa przerwań przychodzących z zewnątrz, ...),
 - sprawność rozdzielania zadań przez system operacyjny,
 - niedeterminizm (przypadek).
- Są **ograniczenia** na skracanie czasu tą metodą:
 - liczba rdzeni,
 - zależności między częściami zadania.