

15. Událostmi řízené programování v Javě

- Komponenty GUI v Javě
- Řízení událostmi a asynchronní programování
- Typy událostí
- Posluchači událostí, anonymní vnitřní třídy

Komponenty GUI v Javě

V Javě lze psát přenositelné aplikace s "okenním" rozhraním - s GUI

- při jejich vývoji se s výhodou uplatní prostředí IDE - „builder“, např. JBuilder, Sun Studio ONE, NetBeans...

V tomto kurzu budeme pracovat s moderním Swing GUI, což je součást JFC (Java Foundation Classes). Starší variantou - dosud živou v Javě verzí 1.1.x, je grafické rozhraní nad komponentami AWT (Abstract Windowing Toolkit).

Událostmi řízené programování

Základním principem tvorby aplikací s GUI je **řízení programu událostmi**.

Netýká se však pouze GUI, je to obecnější pojem označující typ **asynchronního programování**, kdy je:

- tok programu řízen událostmi;
- události nastávají obvykle určitou uživatelskou akcí - klik či pohyb myši, stisk tlačítka...;
- událostmi řízené aplikace musí být většinou programovány jako vícevláknové (i když spouštění vláken obvykle explicitně programovat nemusíme).

Řízení událostmi

Postup (životní cyklus události):

1. Událost vznikne (typicky uživatelskou akcí nad komponentou GUI).
2. Na komponentu musí být "zavěšen" **posluchač** dané **události** (event listener).
3. Systém vyvolá příslušnou metodu posluchače - my tu metodu obvykle smysluplně implementujeme tak, aby realizovala potřebnou akci.

Viz též příklad - [První GUI aplikace](#).

Kde to všechno je?

Technicky jsou dotyčné třídy a rozhraní komponent definovány obvykle v balících:

- `java.awt` - základní komponenty GUI AWT
- `java.awt.event` - události GUI AWT
- a v ostatních balících `java.awt.*`

- `javax.swing` - základní komponenty GUI Swing - rozšíření AWT
- `javax.swing.event` - události GUI Swing
- a v mnoha ostatních balících `javax.swing.*`

Typy událostí vznikajících v GUI

Události mohou souviset s uživatelskou akcí nad/s:

- oknem - `WindowEvent`
- klávesnicí - `KeyEvent`
- myší (klikání a pohyb) - `MouseEvent`
- získáním nebo ztrátou fokusu - `FocusEvent`
- (obecnou) akcí nad GUI (stisk tlačítka v GUI) - `ActionEvent`

Přidání posluchače události

Aby událost mohla být ošetřena, tj. mohlo se na ni někde **reagovat**, je třeba k dané komponentě přidat objekt **posluchače událostí**.

Velmi často - a skoro nikde jinde - se jako objekt posluchače používá objektu **anonymní vnitřní třídy**:

- Třída je (jakoby „on-the-fly“) definována a ihned - jen jedenkrát! - použita.
- Ve skutečnosti samozřejmě se daná třída (její bajtkód) vytvoří a přeloží s ostatními hned ve fázi překladačské mateřské třídy.

V případě posluchačů události obvykle vnitřní třída má jen jednu metodu.

Příklad - událost zavření okna

Proč vůbec pomocí **vnitřní třídy**?

výhody

vnitřní třída má přístup k (i chráněným) prvkům mateřské třídy!

nevýhody

poněkud nepřehledné, třída je skryta v ostatním kódu

navíc: pokud si speciálně nepamätujeme odkaz na jednou vytvořený a zapojený posluchač, pak jej nelze z paměti odstranit - nemáme na něj odkaz

V následujícím úryvku kódu se:

```
okno.addWindowListener(  
    new WindowListener() {  
        public void windowClosing(WindowEvent e) {  
            System.exit(0);  
        }  
    }  
);
```

...vytvoří jedna instance anonymní vnitřní třídy a ta se předá/použije jako posluchač událostí.

Totéž bez anonymity...

```
class MyWindowListener implements WindowListener {
```

```
public void windowClosing(WindowEvent e) {
    System.exit(0);
}
...
okno.addWindowListener(new MyWindowListener());
```

Příklad

- [Mírně rozsáhlejší demo na GUI](#)

Další odkazy

Tvorba Swing-GUI aplikací - [Trail: Creating a GUI with JFC/Swing](#)

Tvorba appletů - [Trail: Writing Applets](#)

Vytváření aplikací přístupných i uživatelům s omezeními:

- potíže se zrakem
- problémy s ovládáním myši...[viz JFC-Accessibility](#)

Vynikající článek o GUI v Javě: [Ray Toal:Developing Complex User Interface Applications in Java](#)