Metody programowania

Wprowadzenie do platformy Java

Dr inż. Andrzej Grosser

Częstochowa, 2013

Spis treści

1. Tworzenie graficznego interfejsu użytkownika cz. 2

Rozdział kontynuuje opis zagadnienia tworzenia graficznego interfejsu użytkownika z wykorzystaniem Swing. W niniejszym rozdziale opisano obsługę zdarzeń, menadżerów układu graficznego (layouty), menu i podstawowe kontrolki interfejsu użytkownika. Do napisania rozdziału wykorzystano książki "Core Java"[2], "Thinking in Java"[1] i dokumentację biblioteki Javy.

1.1. Obsługa zdarzeń

Środowisko operacyjne raportuje zdarzenia do uruchomionego programu. Każdy program decyduje, czy i w jaki sposób odpowiedzieć na te zdarzenia. Sposobów obsługi komunikatów jest wiele, w Swing używany jest model delegacji zdarzeń - każdy obiekt może być słuchaczem zdarzeń (o ile implementuje odpowiedni interfejs). Można w ten sposób dokładnie kontrolować, jak zdarzenia są przesyłane od źródeł zdarzeń do słuchaczy zdarzeń (ang. *event listeners*).

Informacje o zdarzeniu są w Javie zbierane w obiekcie zdarzenia. Wszystkie takie obiekty zdarzenia są wyprowadzane od klasy `java.util.EvenetObject` (lub jednej z jej podklas np. `WindowEvent`, `MenuEvent` i `ActionEvent`).

Obsługa zdarzeń w Javie prezentuje się w sposób następujący:

- Obiekt słuchacza jest instancją klasy, która musi implementować interfejs słuchacza (np. `ActionListener`).
- Źródło zdarzenia jest obiektem, który potrafi rejestrować obiekty słuchaczy i wysyłać im obiekty zdarzeń, gdy wystąpi zdarzenie.
- Obiekty słuchaczy, mogą wykorzystywać informacje zawarte w obiekcie zdarzenia do reakcji na to zdarzenie.

Obiekt słuchacza jest przyporządkowany obiektowi źródła za pomocą wywołania metody add*NazwaZdarzenia*Listener(), gdzie człon NazwaZdarzenia zostaje zastąpiony stosowną nazwą zdarzenia (np. dla Action - addActionListener()). Od momentu wywołania

wymaganej metody słuchacz będzie otrzymywał informacje o zdarzeniu. Dla przykładu można podać przypisanie słuchacza do zdarzenia kliknięcia przycisku (klasa MyListener implementuje interfejs ActionListener):

```
1 ActionListener listener = new MyListener();
2 JButton button = new JButton("Przycisk");
3 button.addActionListener(listener);
```

Implementacja wymaganego interfejsu może być realizowane z wykorzystaniem nazwanej klasy np. dla wymienionej na wcześniejszym listingu klasy MyListener kod może wyglądać następująco:

```
1 class MyListener implements ActionListener {
2    public void actionPerformed(ActionEvent evt) {
3    //...
4    //Kod obsługi zdarzenia
5    //...
6    }
7 }
```

Można również skorzystać z obsługi anonimowych klas wewnętrznych, na przykład:

```
1 JButton button = new JButton("Przycisk");
2  button.addActionListener(new ActionListener() {
3     public void actionPerformed(ActionEvent evt) {
4     //...
5     //Kod obsługi zdarzenia
6     //...
7     }
8  }
9 );
```

1.2. Menadżer układu graficznego

Java obsługuje dynamiczny układ kontrolek na formie za pomocą menadżera układu (ang. *layout manager*). Menadżery pozwalają na dopasowanie rozmiaru kontrolek do zawartości i rozmiarów okna rodzicielskiego (na przykład w reakcji na maksymalizację głównej ramki). Ułożenie kontrolek jest dopasowane za pomocą relacji wyznaczonych przez menadżerów. Automatyczne dopasowanie kontrolek jest szczególnie ważne, gdy aplikacja umożliwia zmianę wyglądu i wrażenia (ang. *look and feel*).

1.2.1. BorderLayout

BorderLayout jest domyślnym menadżerem. Rozmieszcza i zmienia rozmiar swych komponentów tak aby dopasowały się do pięciu regionów: północnego, południowego, wschodniego, zachodniego i centralnego. Każdy region nie może zawierać więcej niż jednego komponentu. Regiony są identyfikowane przez następujące stałe:

- północny - NORTH,
- południowy - SOUTH,
- wschodni - EAST,
- zachodni - WEST,
- centralny - CENTER.

Do dodawania komponentów należy użyć jednej z tych stałych:

```
1 Panel panel = new Panel();
2 panel.setLayout(new BorderLayout());
3 panel.add(new Button("Test"), BorderLayout.NORTH);
```

1.2.2. FlowLayout

Przy tego rodzaju menadżerze kontrolki są rozmieszczane od lewej do prawej, aż do momentu, gdy zostanie wypełniona cała wolna przestrzeń w górnej części formy. Następnie wypełnianie formy kontrolkami jest kontynuowane od nowego wiersza.

1.2.3. GridLayout

Ten rodzaj menadżera umożliwia utworzenie tabeli kontrolek. Kontrolki są umieszczane od lewej do prawej i od góry do dołu. Kontrolki są dopasowane do tego samego rozmiaru. Na przykład:

```
1 JPanel panel = new JPanel();
2 panel.setLayout(new GridLayout(2,3));
3
4 panel.add(new JButton("1"));
5 panel.add(new JButton("2"));
6 panel.add(new JButton("3"));
7 panel.add(new JButton("4"));
8 panel.add(new JButton("5"));
```

1.2.4. GridBagLayout

GridBagLayout jest najbardziej skomplikowanym menadżerem. Nie jest najczęściej używany ręcznie, lecz wykorzystują go graficzne narzędzia do budowy interfejsu użytkownika.

1.2.5. BoxLayout

BoxLayout umożliwia kontolę rozmieszczenia kontrolek w pionie i w poziomie, a także odstępów pomiędzy kontrolkami. W przeciwieństwie do menadżera GridLayout próbuje dopasować wygląd kontrolek do ich preferowanej wielkości, nie zawsze jest to jednak możliwe.

1.2.6. Usuwanie menadżera układu graficznego

W niektórych sytuacjach wymagana jest absolutne ułożenie kontrolek (w ustalonych miejscach i rozmiarach). W takiej sytuacji należy:

- Ustawić menadżer układu na null.
- Dodać wybrany komponent do pojemnika, na jakim ma się pojawić.
- Ustawić wymaganą pozycję i rozmiar komponentu.

Na przykład:

```
1 frame.setLayout(null);
2 JButton btn = new JButton("Test");
3
4 frame.add(btn);
5 btn.setBounds(20, 20, 30, 40);
```

1.3. Menu

Menu może zostać utworzone w komponentach JApplet, JFram, JDialog (i pochodzących od wymienionych). Każdy z tych komponentów posiada metodę `setJMenuBar()` przyjmuje ona obiekt JMenuBar i tworzy menu pionowe, przy czym można utworzyć tylko jedno menu główne dla tych komponentów. Na kontrolce JMenuBar można umieszczać kolejne menu dodając obiekty klasy JMenu. W nich można już umieszczać kolejne pozycje menu reprezentowane przez obiekty klasy `JMenuItem`.

Można podać następujący przykład tworzenia menu (w menu głównym tworzone są dwa menu Plik i Edycja, menu Plik zawiera pozycje Nowy i Zamknij):

```
JMenu[] menu = {new JMenu("Plik"), new JMenu("Edycja"),
        new JMenu("Pomoc")};
JMenuItem[] fileMenu = {new JMenuItem("Nowy"),
        new JMenuItem("Zamknij")};
for(JMenuItem fm : fileMenu)
    menu[0].add(fm);
JMenuBar menuBar = new JMenuBar();
for(JMenu m : menu)
    menubar.add(m);
setJMenuBar(menuBar);
```

1.4. Kontrolki interfejsu użytkownika

Wszystkie kontrolki Swing dziedziczą bezpośrednio lub pośrednio po klasie `JComponent`. Dziedziczą w ten sposób możliwość ustawiania podstawowych właściwości takich jak na przykład kolor tła, podpowiedzi czy obsługi zdarzeń z klawiatury.

Tabela 1.1 przedstawia najważniejsze kontrolki Swing. Skróty w kolumnie kategorii oznaczają odpowiednio:

- KP - kontrolki podstawowe, często występujące w programach takie jak przyciski, suwaki, listy listy rozwijane itp.,
- IW - kontrolki przeznaczone do wyświetlania informacji, której składniki mogą być modyfikowane przez użytkownika - takie jak kontrolki pozwalające wybrać plik, kolor czy na edycję dłuższych tekstów,
- WI - kontrolki wyświetlające informację, bez możliwości edycji, takie jak etykiety, wskaźniki postępu itp.,
- PP - kontenery przeznaczone do przechowywania innych kontrolek - panele, zakładki, paski narzędziowe itp.

Tabela 1.1. Najważniejsze kontrolki Swing

Komponent	Znaczenie	Kategoria
JButton	Przycisk	KP
JCheckBox	Pola wyboru	KP
JComboBox	Lista rozwijana	KP
JList	Lista	KP
JMenu	Menu pionowe	KP
JRadioButton	Przyciski radiowe	KP
JSlider	Suwak	KP
JSpinner	Spinner	KP
JTextField	Pole tekstowe	KP
JPasswordField	Pole hasła	KP
JColorChooser	Wybór koloru	IW
JEditorPane	Edytor tekstu z formatowaniem	IW
JTextPane	Edytor tekstu z formatowaniem	IW
JFileChooser	Wybór pliku	IW
JTable	Tabela	IW
JTextArea	Edytor tekstu bez formatowania	IW
JTree	Drzewo	IW
JLabel	Etykieta	WI
JProgressBar	Pasek postępu	WI
JSeparator	Pionowa lub pozioma linia podziału	WI
JToolTip	Podpowiedź	WI
JPanel	Panel	PP
JScrollPane	Panel z możliwością przewijania	PP
JSplitPane	Panel rozszerzalny	PP
JTabbedPane	Zakładki	PP
JToolBar	Pasek narzędziowy	PP

Literatura

[1] B. Eckel. *Thinking in Java*. Helion, 2006.

[2] C. S. Horstmann and G. Cornell. *Core Java Volume I–Fundamentals*. Prentice Hall, 2011.