Wykład X

Java - środowisko GUI
Biblioteka Swing

Wprowadzenie do systemu menu pakietu Swing

Menu stanowią integralną część wielu aplikacji. Ze względu na znaczenie menu pakiet Swing udostępnia rozbudowane narzędzia do ich tworzenia i obsługi.

.

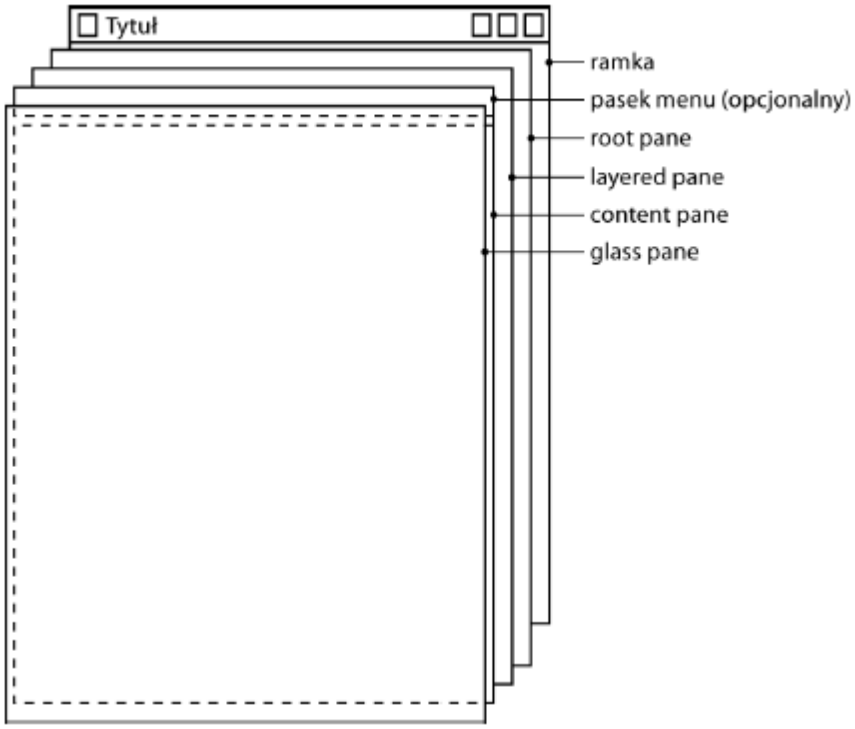
System menu pakietu Swing składa się z kilku kluczowych elementów, takich jak:

- pasek menu stanowiący główne menu aplikacji;
- standardowe menu, które może zawierać zarówno elementy do wybierania, jak i menu podrzędne;
- menu podręczne wyświetlane po kliknięciu prawym przyciskiem myszy;
- pasek narzędzi zapewniający błyskawiczny dostęp do możliwości funkcjonalnych programu, niejednokrotnie powielając przy tym opcje dostępne w menu;
- akcje pozwalające na obsługę dwóch lub większej liczby komponentów przez ten sam obiekt; akcje te są powszechnie stosowane wraz z menu oraz paskami narzędzi.

System menu pakietu Swing obsługuje także klawisze akceleratorów pozwalające na wybieranie opcji bez korzystania z menu, oraz mnemoniki, które pozwalają wybrać przy użyciu klawiatury opcję z już wyświetlonego menu.
System menu pakietu Swing jest obsługiwany przez grupę powiązanych ze sobą klas. Pakiet Swing zapewnia duże możliwości dostosowywania systemu menu, jeśli pojawi się taka potrzeba, jednak zazwyczaj tworzące go klasy są używane w swojej standardowej postaci, gdyż obsługują wszystkie najbardziej potrzebne opcje.

Klasa

- JMenuBar - Obiekt zawierający menu główne aplikacji.
- JMenu - Standardowe menu. Składa się ono z jednego lub więcej obiektów JMenuItem.
- JMenuItem - Obiekt stanowiący zawartość menu.
- JCheckBoxMenuItem - Element menu mający postać pola wyboru.
- JRadioButtonMenuItem - Element menu mający postać przycisku opcji.
- JSeparator - Wizualny separator oddzielający opcje menu.
- JPopupMenu Menu, menu wyświetlane po kliknięciu prawym przyciskiem myszy.



Klasy JMenuBar, JMenu JMenuItem

Trzy podstawowe klasy: JMenuBar, JMenu oraz JMenuItem. Tworzą one minimalny zestaw klas niezbędnych do utworzenia menu aplikacji. Klasy JMenu oraz JMenuItem są także używane do tworzenia menu podręcznych. Dlatego też te trzy klasy tworzą podstawę systemu menu pakietu Swing.

Klasa JMenuBar

JMenuBar jest kontenerem na menu. Jak wszystkie inne komponenty, także i ona dziedziczy po klasie JComponent (która z kolei dziedziczy po klasach Container oraz Component). Klasa ta definiuje tylko jeden konstruktor, który jednocześnie jest konstruktorem domyślnym. Z tego powodu nowy pasek menu zawsze będzie pusty i przed użyciem należy wypełnić go opcjami. Każda aplikacja ma jeden i tylko jeden pasek menu.

Klasa JMenuBar definiuje kilka metod, jednak zazwyczaj używana jest tylko jedna z nich — metoda add(). Metoda ta, przedstawiona poniżej, dodaje do paska menu obiekt JMenu.

JMenu add(JMenu *menu*)

Przy czym parametr *menu* jest instancją typu JMenu, dodawaną do paska menu. Metoda ta zwraca referencję do menu. Elementy dodawane do paska menu są na nim rozmieszczane od strony lewej do prawej, w kolejności dodawania. Aby dodać menu w konkretnym miejscu paska należy użyć poniższej wersji tej metody, odziedziczonej po klasie Container:

Component add(Component *menu*, int *idx*)

W tym przypadku *menu* zostanie dodane w miejscu określonym przez parametr *idx*. Indeksowanie elementów paska menu rozpoczyna się od 0, przy czym wartość ta reprezentuje skrajny, lewy element.

Usunięcia niepotrzebnego już menu.

Do tego celu służy metoda remove(), dziedziczona po klasie Component. Jest ona dostępna w dwóch wersjach:

void remove(Component *menu*)

void remove(int *idx*)

W tym przypadku *menu* jest referencją do usuwanego menu, a *idx* indeksem menu, które należy usunąć.

Kolejną metodą, która czasami może się przydać, jest przedstawiona poniżej metoda getMenuCount():
int getMenuCount()
Zwraca ona liczbę elementów dodanych do paska menu.
Metoda isSelected() sprawdza, czy dane menu zostało wybrane.
Po utworzeniu i wypełnieniu paska menu należy go dodać do obiektu JFrame. Do tego celu służy metoda setJMenuBar(). (Pasków menu *nie* dodaje się do paneli zawartości). Poniżej przedstawiona została postać metody setJMenuBar():
void setJMenuBar(JMenuBar *mb*)
Parametr *mb* reprezentuje referencję do paska menu. Pasek menu zostanie wyświetlony w miejscu zależnym od wyglądu i sposobu obsługi wybranego w aplikacji. Zazwyczaj będzie on wyświetlany poniżej górnej krawędzi okna.

Klasa JMenu

Klasa JMenu reprezentuje menu, które jest wypełniane obiektami JMenuItem.
Klasa ta definiuje kilka różnych konstruktorów. przedstawiony
Jeden z najczęściej stosowanych:
JMenu(String *name*)
Ten konstruktor tworzy menu posiadające tytuł określony parametrem *name*. Oczywiście podanie nazwy menu nie jest konieczne. Aby utworzyć menu bez nazwy, wystarczy użyć konstruktora domyślnego:
JMenu()
Klasa JMenu udostępnia także inne konstruktory. Każdy z konstruktorów tej klasy zwraca menu, które jest początkowo puste, i dopiero później należy dodać jego zawartość.
Klasa JMenu definiuje wiele metod.
Do dodawania elementów do menu służy metoda add():
JMenuItem add(JMenuItem *item*)
JMenuItem add(Component *item*, int *idx*)
W obu powyższych metodach parametr *item* określa dodawany element menu. Pierwsza wersja metody dodaje nowy element na końcu menu, natomiast druga — w miejscu określonym parametrem *idx*. Obie wersje metody zwracają referencję do dodanego elementu menu. W ramach ciekawostki warto dodać, że
elementy menu można także dodawać przy użyciu metody insert().
Do dodawania separatorów (obiektów typu JSeparator) służy przedstawiona poniżej metoda
addSeparator():
void addSeparator()
Separator dodawany jest na końcu menu. Separator można też wstawić w wybrane miejsce menu, wywołując w tym celu poniższą metodę insertSeparator():
void insertSeparator(int *idx*)
Parametr *idx* określa liczony od zera indeks miejsca, w którym zostanie wstawiony separator.
Do usuwania elementów menu służy metoda remove(). Poniżej przedstawione zostały jej dwie wersje:
void remove(JMenuItem *menu*)
void remove(int *idx*)
W tym przypadku parametr *menu* jest referencją do usuwanego elementu menu, natomiast parametr *idx* indeksem elementu, który należy usunąć.
Liczbę elementów menu można pobrać przy użyciu przedstawionej poniżej metody getMenuComponentCount():
int getMenuComponentCount()
Można także pobrać tablicę zawierającą referencje do wszystkich elementów menu; służy do tego metoda getMenuComponents():
Component[] getMenuComponents()
Metoda zwraca tablicę komponentów.

Tworzenie menu głównego

Tradycyjnie najczęściej używanym menu w aplikacji jest **menu główne**. To właśnie to menu jest definiowane przez pasek menu i zapewnia dostęp do funkcji aplikacji.
Proces tworzenia menu głównego składa się z kilku kroków.
W pierwszej kolejności należy utworzyć obiekt JMenuBar, który będzie zawierał menu.
Kolejnym krokiem jest utworzenie każdego menu, które ma być dostępne w pasku menu.
Ogólnie rzecz biorąc, w celu przygotowania menu należy najpierw utworzyć obiekt JMenu, a następnie dodać do niego obiekty JMenuItem. Po przygotowaniu wszystkich menu należy je dodać do paska menu. W końcu sam pasek menu trzeba dodać do ramki, wywołując w tym celu metodę setJMenuBar(). Ostatnim etapem przygotowań jest określenie obiektów nasłuchujących, które będą obsługiwać zdarzenia ActionEvent generowane przez poszczególne elementy menu.
Doskonałym sposobem na zrozumienie procesu tworzenia i zarządzania menu jest przeanalizowanie przykładu.

```
// Demonstruje proste menu główne.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class J_Menu implements ActionListener {
JLabel jlab;
J_Menu() {
// Tworzy kontener JFrame.
JFrame jfrm = new JFrame("Prezentacja Menu");
// Zastosowanie menedżera układu FlowLayout.
jfrm.setLayout(new FlowLayout());
// Określenie początkowych wymiarów ramki.
jfrm.setSize(520, 300);
// Zamknięcie okna aplikacji ma kończyć działanie programu.
jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// Tworzy etykietę, w której będą wyświetlane nazwy
// wybieranych opcji menu.
jlab = new JLabel();
// Tworzy pasek menu.
JMenuBar jmb = new JMenuBar();
// Utworzenie menu Plik menu.
JMenu jmFile = new JMenu("Plik");
JMenuItem jmiOpen = new JMenuItem("Otwórz");
JMenuItem jmiClose = new JMenuItem("Zamknij");
JMenuItem jmiSave = new JMenuItem("Zapisz");
JMenuItem jmiExit = new JMenuItem("Zakończ");
jmFile.add(jmiOpen);
jmFile.add(jmiClose);
jmFile.add(jmiSave);
jmFile.addSeparator();
jmFile.add(jmiExit);
jmb.add(jmFile);
// Utworzenie menu Opcje.
JMenu jmOptions = new JMenu("Opcje");
// Utworzenie podmenu Kolory.
JMenu jmColors = new JMenu("Kolory");
JMenuItem jmiRed = new JMenuItem("Czerwony");
JMenuItem jmiGreen = new JMenuItem("Zielony");
JMenuItem jmiBlue = new JMenuItem("Niebieski");
jmColors.add(jmiRed);
jmColors.add(jmiGreen);
jmColors.add(jmiBlue);
//Dodanie podmenu do menu Opcje
jmOptions.add(jmColors);
//Utworzenie podmenu Priorytet.
JMenu jmPriority = new JMenu("Priorytet");
```

```
JMenuItem jmiHigh = new JMenuItem("Wysoki");
JMenuItem jmiLow = new JMenuItem("Niski");
jmPriority.add(jmiHigh);
jmPriority.add(jmiLow);
//Dodanie podmenu do menu Opcje
jmOptions.add(jmPriority);
//Utworzenie opcji Resetuj.
JMenuItem jmiReset = new JMenuItem("Resetuj");
jmOptions.addSeparator();
jmOptions.add(jmiReset);
//Dodaje całe menu Opcje do paska menu.
jmb.add(jmOptions);
//Tworzy menu Pomoc.
JMenu jmHelp = new JMenu("Pomoc");
JMenuItem jmiAbout = new JMenuItem("O programie");
jmHelp.add(jmiAbout);
jmb.add(jmHelp);
//Dodaje wszystkie obiekty nasłuchujące.
jmiOpen.addActionListener(this);
jmiClose.addActionListener(this);
jmiSave.addActionListener(this);
jmiExit.addActionListener(this);
jmiRed.addActionListener(this);
jmiGreen.addActionListener(this);
jmiBlue.addActionListener(this);
jmiHigh.addActionListener(this);
jmiLow.addActionListener(this);
jmiReset.addActionListener(this);
jmiAbout.addActionListener(this);
//Dodaje etykietę do panelu zawartości.
jfrm.add(jlab);
//Dodaje pasek menu do ramki.
jfrm.setJMenuBar(jmb);
//Wyświetla ramkę.
jfrm.setVisible(true);
}
//Obsługa zdarzeń generowanych przez wybierane opcje menu.
public void actionPerformed(ActionEvent ae) {
//Pobranie łańcucha polecenia wybranej opcji menu.
String comStr = ae.getActionCommand();
//W przypadku wybrania opcji Zakończ należy zakończyć
//działanie programu.
if(comStr.equals("Zakończ")) System.exit(0);
//W pozostałych przypadkach wyświetlana jest nazwa opcji.
jlab.setText("Wybrano " + comStr);
}
public static void main(String args[]) {
//Tworzy ramkę w wątku obsługi zdarzeń.
SwingUtilities.invokeLater(new Runnable() {
public void run() {
new J_Menu();
}
});
}
}
```

Dodawanie mnemonik i kombinacji klawiszy do opcji menu

Menu utworzone w przedstawionym przykładzie jest standardowym i funkcjonalnym rozwiązaniem. W rozbudowanych aplikacjach menu zazwyczaj udostępniają skróty klawiszowe (nazywane także czasami kombinacjami klawiszy), gdyż dają one bardziej zaawansowanym użytkownikom możliwość szybszego wybierania opcji menu. Dostępne są dwa rodzaje skrótów klawiszowych: mnemoniki (ang.*mnemonic*) oraz akceleratory (ang. *acceleators*). W kontekście systemu menu pakietu Swing **mnemoniki** pozwalają na definiowanie klawisza służącego do wyboru opcji z aktywnego menu. A zatem pozwalają one wybrać opcję z już rozwiniętego, widocznego menu poprzez naciśnięcie odpowiedniego klawisza raczej rzadko stosowane. **Akcelerator** to klawisz, który można nacisnąć, by wybrać opcję menu bez konieczności jego wyświetlania.
Mnemoniki można określać zarówno dla obiektów JMenuItem, jak i JMenu. Można to robić na dwa sposoby. Pierwszym z nich jest podanie mnemoniki podczas tworzenia obiektu; możliwość tę zapewnia następujący konstruktor:
JMenuItem(String *name*, int *mnem*)
W tym przypadku nazwa elementu menu zostaje przekazana przy użyciu parametru *name*, natomiast parametr *mnem* określa mnemonikę. Oprócz tego mnemoniki można także określać, wywołując metodę setMnemonic(). Jest ona dziedziczona przez obie klasy po klasie AbstractButton i ma następującą postać:
void setMnemonic(int *mnem*)
Także w tej metodzie parametr *mnem* określa mnemonikę. Jego wartością powinna być jedna ze stałych zdefiniowanych w klasie java.awt.event.KeyEvent, takich jak KeyEvent.VK_F lub KeyEvent.VK_Z.
W przypadku mnemonik wielkość litery nie ma znaczenia, zatem użycie stałej VK_A sprawi, że mnemonika zadziała zarówno po wpisaniu litery *a*, jak i *A*.
Domyślnie po określeniu mnemoniki pierwsza pasująca do niej litera nazwy widocznej w elemencie menu zostanie podkreślona. Gdy konieczne jest wyróżnienie innej litery, należy podać jej indeks w wywołaniu metody setDisplayedMnemonicIndex(), którą klasy JMenu oraz JMenuItem dziedziczą po klasie AbstractMenu. Postać tej metody została przedstawiona poniżej:
void setDisplaydMnemonicIndex(int *idx*)
Indeks litery, którą należy podkreślić, zostaje określony przez parametr *idx*.

Akcelerator można skojarzyć z obiektem JMenuItem. W tym celu należy wywołać przedstawioną poniżej metodę setAccelerator():
void setAccelerator(KeyStroke *ks*)
W tej metodzie parametr *ks* określa kombinację klawiszy, którą należy nacisnąć w celu wybrania elementu menu. Klasa KeyStroke posiada kilka metod fabrycznych służących do tworzenia różnych rodzajów akceleratorów. Poniżej przedstawione zostały trzy spośród tych metod:
static KeyStroke getKeyStroke(char *ch*)
static KeyStroke getKeyStroke(Character *ch*, int *modifier*)
static KeyStroke getKeyStroke(int *ch*, int *modifier*)
static KeyStroke getKeyStrokeForEvent(KeyEvent *anEvent*)
W każdej z tych metod parametr *ch* określa znak akceleratora. W pierwszej z nich znak ten jest określany przy użyciu wartości typu char. Druga metoda pozwala na określenie znaku akceleratora przy użyciu obiektu klasy Character. Trzecia określa znak akceleratora przy użyciu wartości typu int. W końcu ostatnia z przedstawionych metod używa wartości opisanego wcześniej typu KeyEvent.

Wartością parametru *modifier* musi być jedna ze stałych zdefiniowanych w klasie java.awt.event. - InputEvent bądź ich alternatywa bitowa:

- InputEvent.ALT_DOWN_MASK
- InputEvent.ALT_GRAPH_DOWN_MASK
- InputEvent.CTRL_DOWN_MASK
- InputEvent.META_DOWN_MASK
- InputEvent.SHIFT_DOWN_MASK

Jeśli więc znak akceleratora zostanie określony przy użyciu stałej VK_A, a dodatkowo zostanie zastosowany modyfikator InputEvent.CTRL_DOWN_MASK, to akceleratorem będzie kombinacja klawiszy *Ctrl+A*.

Przykład programu z zastosowaniem akceleratorów i mnemonik.

```
// Demonstruje proste menu główne.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class J_Menu implements ActionListener {
JLabel jlab;
J_Menu() {
// Tworzy kontener JFrame.
JFrame jfrm = new JFrame("Prezentacja Menu");
// Zastosowanie menedżera układu FlowLayout.
jfrm.setLayout(new FlowLayout());
// Określenie początkowych wymiarów ramki.
jfrm.setSize(520, 300);
// Zamknięcie okna aplikacji ma kończyć działanie programu.
jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// Tworzy etykietę, w której będą wyświetlane nazwy
// wybieranych opcji menu.
jlab = new JLabel();
// Tworzy pasek menu.
JMenuBar jmb = new JMenuBar();
//Utworzenie menu Plik menu wraz z mnemonikami i akceleratorami.
JMenu jmFile = new JMenu("Plik");
jmFile.setMnemonic(KeyEvent.VK_P);
JMenuItem jmiOpen = new JMenuItem("Otwórz", KeyEvent.VK_O);
jmiOpen.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_O, InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiClose = new JMenuItem("Zamknij", KeyEvent.VK_M);
jmiClose.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_M, InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiSave = new JMenuItem("Zapisz", KeyEvent.VK_Z);
jmiSave.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_Z, InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiExit = new JMenuItem("Zakończ", KeyEvent.VK_K);
jmiExit.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_K, InputEvent.CTRL_DOWN_MASK));

jmFile.add(jmiOpen);
jmFile.add(jmiClose);
jmFile.add(jmiSave);
jmFile.addSeparator();
jmFile.add(jmiExit);
jmb.add(jmFile);
// Utworzenie menu Opcje.
JMenu jmOptions = new JMenu("Opcje");
// Utworzenie podmenu Kolory.
JMenu jmColors = new JMenu("Kolory");
JMenuItem jmiRed = new JMenuItem("Czerwony");
JMenuItem jmiGreen = new JMenuItem("Zielony");
JMenuItem jmiBlue = new JMenuItem("Niebieski");
jmColors.add(jmiRed);
```

```
jmColors.add(jmiGreen);
jmColors.add(jmiBlue);
//Dodanie podmenu do menu Opcje
jmOptions.add(jmColors);
//Utworzenie podmenu Priorytet.
JMenu jmPriority = new JMenu("Priorytet");
JMenuItem jmiHigh = new JMenuItem("Wysoki");
JMenuItem jmiLow = new JMenuItem("Niski");
jmPriority.add(jmiHigh);
jmPriority.add(jmiLow);
//Dodanie podmenu do menu Opcje
jmOptions.add(jmPriority);
//Utworzenie opcji Resetuj.
JMenuItem jmiReset = new JMenuItem("Resetuj");
jmOptions.addSeparator();
jmOptions.add(jmiReset);
//Dodaje całe menu Opcje do paska menu.
jmb.add(jmOptions);
//Tworzy menu Pomoc.
JMenu jmHelp = new JMenu("Pomoc");
JMenuItem jmiAbout = new JMenuItem("O programie");
jmHelp.add(jmiAbout);
jmb.add(jmHelp);
//Dodaje wszystkie obiekty nasłuchujące.
jmiOpen.addActionListener(this);
jmiClose.addActionListener(this);
jmiSave.addActionListener(this);
jmiExit.addActionListener(this);
jmiRed.addActionListener(this);
jmiGreen.addActionListener(this);
jmiBlue.addActionListener(this);
jmiHigh.addActionListener(this);
jmiLow.addActionListener(this);
jmiReset.addActionListener(this);
jmiAbout.addActionListener(this);
//Dodaje etykietę do panelu zawartości.
jfrm.add(jlab);
//Dodaje pasek menu do ramki.
jfrm.setJMenuBar(jmb);
//Wyświetla ramkę.
jfrm.setVisible(true);
}
//Obsługa zdarzeń generowanych przez wybierane opcje menu.
public void actionPerformed(ActionEvent ae) {
//Pobranie łańcucha polecenia wybranej opcji menu.
String comStr = ae.getActionCommand();
//W przypadku wybrania opcji Zakończ należy zakończyć
//działanie programu.
if(comStr.equals("Zakończ")) System.exit(0);
//W pozostałych przypadkach wyświetlana jest nazwa opcji.
jlab.setText("Wybrano " + comStr);
}
public static void main(String args[]) {
//Tworzy ramkę w wątku obsługi zdarzeń.
SwingUtilities.invokeLater(new Runnable() {
public void run() {
new J_Menu();
}
});
}
}
```

Klasy JRadioButtonMenuItem oraz JCheckBoxMenuItem

Pakiet Swing udostępnia także dwa inne pola menu; są nimi pola wyboru oraz przyciski opcji. Te typy elementów menu mogą ułatwić korzystanie z programów, gdyż zapewniają możliwości funkcjonalne, które w innych przypadkach byłyby dostępne wyłącznie przy użyciu niezależnych komponentów.
W celu dodania do menu pola wyboru należy utworzyć obiekt klasy JChecBoxMenuItem. Klasa ta definiuje kilka różnych konstruktorów:
JCheckBoxMenuItem(String *name*)
Parametr *name* określa nazwę elementu. Początkowo tak utworzone pole wyboru nie będzie zaznaczone.
Jeśli konieczne jest określenie innego początkowego stanu pola, to można to zrobić, używając następującego konstruktora:
JCheckBoxMenuItem(String *name*, boolean *state*)
W tym przypadku, jeśli parametr *state* przyjmie wartość true, to utworzone pole wyboru będzie zaznaczone. W przeciwnym razie pole będzie puste. Klasa JCheckBoxMenuItem udostępnia także konstruktor pozwalający na określenie ikony:
JCheckBoxMenuItem(String *name*, Icon *icon*)
W tym przypadku parametr *name* określa nazwę elementów, natomiast obrazek, który będzie wyświetlony obok nazwy, jest przekazywany jako parametr *icon*. Także w tym przypadku utworzone pole wyboru początkowo nie będzie zaznaczone. Klasa JCheckBoxMenuItem udostępnia także kilka innych konstruktorów.
Pola wyboru umieszczane w opcjach menu działają tak samo jak niezależne pola wyboru. Na przykład w razie zmiany stanu generują one zdarzenia ActionEvent i ItemEvent. Stosowanie opcji tego rodzaju w menu jest szczególnie przydatne w sytuacjach, gdy aplikacja udostępnia opcje, które można wybierać, i chcemy pokazać ich aktualny status.

Przyciski opcji można dodawać do menu przy użyciu obiektów klasy JRadioButtonMenuItem. Klasa ta dziedziczy po klasie JMenuItem. Udostępnia ona kilka konstruktorów:
JRadioButtonMenuItem(String *name*)
JRadioButtonMenuItem(Stirng *name*, boolean *state*)
Pierwszy z tych konstruktorów tworzy przycisk opcji skojarzony z nazwą przekazaną przy użyciu parametru *name*; początkowo przycisk ten nie będzie zaznaczony. Drugi z konstruktorów umożliwia określenie początkowego stanu przycisku. Jeśli parametr *state* przyjmie wartość true, to przycisk zostanie zaznaczony. Przekazanie wartości false oznacza, że początkowo przycisk nie będzie zaznaczony. Dostępne są także konstruktory pozwalające na określenie obrazka; poniżej przedstawiony został jeden z nich:
JRadioButtonMenuItem(String *name*, Icon *icon*, boolean *state*)
Powyższy konstruktor tworzy element menu zawierający przycisk opcji skojarzony z nazwą przekazaną przy użyciu parametru *name* oraz obrazkiem określonym przy użyciu parametru *icon*.
Jeśli parametr *state* przyjmie wartość true, to przycisk opcji początkowo będzie zaznaczony; jeśli parametr ten przyjmie wartość false, to przycisk będzie pusty. Dostępnych jest także kilka innych konstruktorów.
Opcje menu tworzone przy użyciu klasy JRadioButtonMenuItem działają jak niezależne przyciski opcji — generują zdarzenia ActionEvent oraz ItemEvent. Podobnie jak niezależne przyciski opcji, także i elementy menu tego typu muszą być umieszczane w grupach przycisków, by działały w oczekiwany sposób — tak by w danej chwili mógł być zaznaczony tylko jeden przycisk z grupy.
Ponieważ klasy JCheckBoxMenuItem oraz JRadioButtonMenuItem dziedziczą po JMenuItem, zatem każda z nich dysponuje udostępnianymi przez nią możliwościami funkcjonalnymi. Innymi słowy, nie tylko posiadają cechy charakterystyczne dla pól wyboru oraz przycisków opcji, ale też działają jak inne elementy menu i w taki sposób

można ich używać.

```
// Demonstruje proste menu główne.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class J_Menu implements ActionListener {
JLabel jlab;
J_Menu() {
// Tworzy kontener JFrame.
JFrame jfrm = new JFrame("Prezentacja Menu");
// Zastosowanie menedżera układu FlowLayout.
jfrm.setLayout(new FlowLayout());
// Określenie początkowych wymiarów ramki.
jfrm.setSize(520, 300);
// Zamknięcie okna aplikacji ma kończyć działanie programu.
jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// Tworzy etykietę, w której będą wyświetlane nazwy
// wybieranych opcji menu.
jlab = new JLabel();
// Tworzy pasek menu.
JMenuBar jmb = new JMenuBar();
//Utworzenie menu Plik menu wraz z mnemonikami i akceleratorami.
JMenu jmFile = new JMenu("Plik");
jmFile.setMnemonic(KeyEvent.VK_P);
JMenuItem jmiOpen = new JMenuItem("Otwórz", KeyEvent.VK_O);
jmiOpen.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_O, InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiClose = new JMenuItem("Zamknij", KeyEvent.VK_M);
jmiClose.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_M, InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiSave = new JMenuItem("Zapisz", KeyEvent.VK_Z);
jmiSave.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_Z, InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiExit = new JMenuItem("Zakończ", KeyEvent.VK_K);
jmiExit.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_K, InputEvent.CTRL_DOWN_MASK));

jmFile.add(jmiOpen);
jmFile.add(jmiClose);
jmFile.add(jmiSave);
jmFile.addSeparator();
jmFile.add(jmiExit);
jmb.add(jmFile);
JMenu jmOptions = new JMenu("Opcje");
//Utworzenie podmenu Kolory.
JMenu jmColors = new JMenu("Kolory");
//Tworzy elementy menu z polami wyboru i dodaje je do podmenu Kolory. Dzięki
//temu użytkownik będzie mógł wybrać więcej niż jeden kolor.
JCheckBoxMenuItem jmiRed = new JCheckBoxMenuItem("Czerwony");
JCheckBoxMenuItem jmiGreen = new JCheckBoxMenuItem("Zielony");
JCheckBoxMenuItem jmiBlue = new JCheckBoxMenuItem("Niebieski");
jmColors.add(jmiRed);
jmColors.add(jmiGreen);
jmColors.add(jmiBlue);
jmOptions.add(jmColors);
//Utworzenie podmenu Priorytet.
JMenu jmPriority = new JMenu("Priorytet");
//Tworzy elementy menu z przyciskami opcji i dodaje je do menu Priorytet.
//W ten sposób możemy pokazać aktualnie wybrany priorytet,
//a jednocześnie mamy pewność, że w dowolnej chwili będzie wybrany
//tylko jeden priorytet. Warto zwrócić uwagę, że początkowo zostanie
//zaznaczona opcja „Wysoki”.
JRadioButtonMenuItem jmiHigh = new JRadioButtonMenuItem("Wysoki", true);
```

```
JRadioButtonMenuItem jmiLow = new JRadioButtonMenuItem("Niski");
jmPriority.add(jmiHigh);
jmPriority.add(jmiLow);
jmOptions.add(jmPriority);
//Tworzy grupę przycisków dla elementów menu zawierających
//przyciski opcji.
ButtonGroup bg = new ButtonGroup();
bg.add(jmiHigh);
bg.add(jmiLow);
//Utworzenie opcji Resetuj.
JMenuItem jmiReset = new JMenuItem("Resetuj");
jmOptions.addSeparator();
jmOptions.add(jmiReset);
//Dodaje całe menu Opcje do paska menu.
jmb.add(jmOptions);

//Tworzy menu Pomoc.
JMenu jmHelp = new JMenu("Pomoc");
JMenuItem jmiAbout = new JMenuItem("O programie");
jmHelp.add(jmiAbout);
jmb.add(jmHelp);
//Dodaje wszystkie obiekty nasłuchujące.
jmiOpen.addActionListener(this);
jmiClose.addActionListener(this);
jmiSave.addActionListener(this);
jmiExit.addActionListener(this);
jmiRed.addActionListener(this);
jmiGreen.addActionListener(this);
jmiBlue.addActionListener(this);
jmiHigh.addActionListener(this);
jmiLow.addActionListener(this);
jmiReset.addActionListener(this);
jmiAbout.addActionListener(this);
//Dodaje etykietę do panelu zawartości.
jfrm.add(jlab);
//Dodaje pasek menu do ramki.
jfrm.setJMenuBar(jmb);
//Wyświetla ramkę.
jfrm.setVisible(true);
}
//Obsługa zdarzeń generowanych przez wybierane opcje menu.
public void actionPerformed(ActionEvent ae) {
//Pobranie łańcucha polecenia wybranej opcji menu.
String comStr = ae.getActionCommand();
//W przypadku wybrania opcji Zakończ należy zakończyć
//działanie programu.
if(comStr.equals("Zakończ")) System.exit(0);
//W pozostałych przypadkach wyświetlana jest nazwa opcji.
jlab.setText("Wybrano " + comStr);
}
public static void main(String args[]) {
//Tworzy ramkę w wątku obsługi zdarzeń.
SwingUtilities.invokeLater(new Runnable() {
public void run() {
new J_Menu();
}
});
}
}
```

Tworzenie menu podręcznych

Popularną alternatywą bądź też uzupełnieniem dla paska menu są menu podręczne. Zazwyczaj menu tego typu aktywuje się poprzez kliknięcie prawym przyciskiem myszy na komponencie. Pakiet Swing obsługuje menu podręczne, udostępniając klasę JPopupMenu. Klasa ta udostępnia dwa konstruktory, przy czym w tym rozdziale będzie używany wyłącznie przedstawiony poniżej konstruktor domyślny:
JPopupMenu()
Konstruktor ten tworzy domyślne menu podręczne. Druga wersja konstruktora pozwala określić tytuł menu. To, czy tytuł ten zostanie wyświetlony, zależy od wybranego wyglądu i sposobu obsługi.
W pierwszej kolejności należy utworzyć obiekt JPopupMenu, a następnie dodać do niego elementy menu.
Obsługa opcji wybieranych z menu podręcznych realizowana jest w taki sam sposób jak obsługa menu głównego: poprzez odbieranie zdarzeń. Podstawowa różnica pomiędzy menu podręcznymi a menu głównym polega na sposobie ich wyświetlania.
Aby wyświetlić menu podręczne, należy wykonać następujące czynności:

1. Zarejestrować obiekt nasłuchujący, obsługujący zdarzenia myszy.
2. W kodzie obsługującym zdarzenia myszy należy sprawdzić, czy nie wystąpił warunek, który ma powodować wyświetlenie menu podręcznego.
3. Jeśli taki warunek wystąpił, to należy wyświetlić menu, wywołując w tym celu metodę show().

Menu podręczne są zazwyczaj wyświetlane w odpowiedzi na kliknięcie prawym przyciskiem myszy na komponencie, dla którego konkretne menu zostało zdefiniowane. A zatem **warunkiem wyświetlenia** jest przeważnie kliknięcie prawym przyciskiem myszy na odpowiednim elemencie.
Aby móc odbierać te zdarzenia, należy zaimplementować interfejs MouseListener, a następnie zarejestrować obiekt nasłuchujący, wywołując metodę addMouseListener().
Interfejs MouseListener definiuje następujące metody:
void mouseClicked(MouseEvent *me*)
void mouseEntered(MouseEvent *me*)
void mouseExited(MouseEvent *me*)
void mousePressed(MouseEvent *me*)
void mouseReleased(MouseEvent *me*)
Spośród tych wszystkich metod dwie mają szczególne znaczenie w kontekście obsługi menu podręcznego. Chodzi o metody mousePressed() oraz mouseReleased(). W zależności od wybranego wyglądu i sposobu obsługi menu podręczne można wyświetlać w odpowiedzi na jedno z tych dwóch zdarzeń. Z tego względu zazwyczaj najłatwiej jest zaimplementować interfejs MouseListener, rozszerzając klasę MouseAdapter i przesłaniając metody mousePressed() i mouseReleased().
Klasa MouseEvent definiuje kilka metod, jednak zazwyczaj podczas wyświetlania menu podręcznego używane są jedynie cztery z nich:
int getX()
int getY()
boolean isPopupTrigger()
Component getComponent()
Bieżące współrzędne X i Y, określające położenie wskaźnika myszy w stosunku do źródła zdarzenia, można pobrać przy użyciu metod getX() oraz get(). Współrzędne te służą do określenia położenia lewego górnego wierzchołka wyświetlanego menu. Metoda isPopupTrigger() zwraca wartość true, jeśli zdarzenie odpowiada warunkowi wyświetlenia menu; w przeciwnym razie zwraca ona wartość false. Metoda ta jest używana do określania, kiedy należy wyświetlić menu. Z kolei metoda getComponent() pozwala pobrać referencję do komponentu, który wygenerował zdarzenie myszy.
Aby wyświetlić utworzone wcześniej menu, należy wywołać przedstawioną poniżej

metodę show() klasy JPopupMenu:
void show(Component *invoker*, int *upperX*, int *upperY*)
W tym przypadku parametr *invoker* określa komponent, względem którego będzie określane położenie menu. Z kolei wartości parametrów *upperX* oraz *upperY* określają odpowiednio współrzędną poziomą i pionową lewego górnego wierzchołka menu, wyrażone względem komponentu *invoker*. Popularnym sposobem określania tego komponentu jest wywołanie metody getComponent() na rzecz obiektu zdarzenia.

Tworzenie paska narzędzi

Pasek narzędzi jest komponentem, który może stanowić zarówno alternatywę, jak i dodatek do menu. Paski narzędzi zawierają listę przycisków (lub innych komponentów) zapewniających natychmiastowy dostęp do różnych opcji programu. Na przykład pasek narzędzi może zawierać przyciski pozwalające na wybieranie różnych opcji określających postać czcionki, takich jak pogrubienie, kursywa, wysokość wiersza bądź podkreślenie. Przyciski umieszczane na paskach narzędzi zazwyczaj prezentują ikony, a nie tekst, choć mogą także prezentować tekst bądź zarówno tekst, jak i ikony.
Co więcej, z przyciskami na paskach narzędzi są zazwyczaj skojarzone etykiety ekranowe. Paski narzędzi można umieszczać przy dowolnej krawędzi okna aplikacji, wystarczy je przeciągnąć; można je także umieszczać poza oknem, a wtedy stają się „paskami pływającymi".

W pakiecie Swing paski narzędzi tworzone są przy użyciu klasy JToolBar. Jej konstruktory pozwalają na tworzenie pasków narzędzi, które mają tytuł lub nie. Można także określić układ paska narzędzi może być pionowy bądź poziomy. Poniżej przedstawione zostały dostępne konstruktory klasy JToolBar:
JToolBar()
JToolBar(String *title*)
JToolBar(int *how*)
JToolBar(String *title*, int *how*)
Pierwszy z przedstawionych konstruktorów tworzy poziomy pasek narzędzi, który nie ma tytułu. Drugi konstruktor tworzy poziomy pasek z tytułem określonym parametrem *title*. Tytuł paska będzie widoczny wyłącznie w przypadku, gdy pasek zostanie przeciągnięty poza obszar okna aplikacji. Trzeci z przedstawionych konstruktorów tworzy pasek narzędzi o układzie określonym przez parametr *how*. Parametr ten może przyjmować dwie wartości: JToolBar.VERTICAL lub JToolBar.HORIZONTAL.
Z kolei ostatni konstruktor tworzy pasek narzędzi o określonej nazwie i układzie.
Paski narzędzi są zazwyczaj używane w oknach korzystających z układu BorderLayout. Wynika to z dwóch przyczyn. Przede wszystkim dzięki temu pasek narzędzi można początkowo wyświetlić wzdłuż jednej z czterech krawędzi okna. Zazwyczaj jest on umieszczany poniżej górnej krawędzi. Po drugie, zapewnia to możliwość przeciągania paska narzędzi i wyświetlania go wzdłuż dowolnej krawędzi okna. Oprócz przeciągania paska w różne miejsca wewnątrz okna aplikacji można go także przeciągnąć zupełnie poza to okno. Taki pasek umieszczony poza oknem aplikacji nazywany jest paskiem *niezadokowanym*.
Jeśli podczas tworzenia paska został określony jego tytuł, to na niezadokowanym pasku będzie on widoczny. Przyciski (oraz inne komponenty) można dodawać do pasków narzędzi dokładnie tak samo jak do paska menu. Wystarczy skorzystać z metody add(). Komponenty są wyświetlane na pasku narzędzi w kolejności, w jakiej zostały do niego dodane.
Po utworzeniu paska narzędzi *nie należy* dodawać go do paska menu (jeśli taki istnieje). Zamiast tego jest on dodawany do pojemnika okna. Jak już wspomniano, paski narzędzi zazwyczaj są wyświetlane poniżej górnej krawędzi okna (w położeniu określanym jako BorderLayout.NORTH) i mają układ poziomy. Pozostałe komponenty są dodawane do środkowego obszaru menedżera układu.

Takie rozwiązanie sprawia, że po uruchomieniu programu pasek narzędzi będzie widoczny w oczekiwanym miejscu. Niemniej jednak można go przeciągnąć i umieścić w dowolnym innym miejscu. Oczywiście można także przeciągnąć pasek narzędzi zupełnie poza okno aplikacji.

Przykład programu:

```
// Demonstruje proste menu główne.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class J_Menu implements ActionListener {
JLabel jlab;
JPopupMenu jpu;
J_Menu() {
// Tworzy kontener JFrame.
JFrame jfrm = new JFrame("Prezentacja Menu");
// Zastosowanie menedżera układu FlowLayout.
//jfrm.setLayout(new FlowLayout());
// Określenie początkowych wymiarów ramki.
jfrm.setSize(520, 300);
// Zamknięcie okna aplikacji ma kończyć działanie programu.
jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// Tworzy etykietę, w której będą wyświetlane nazwy
// wybieranych opcji menu.
jlab = new JLabel();
// Tworzy pasek menu.
JMenuBar jmb = new JMenuBar();
//Utworzenie menu Plik menu wraz z mnemonikami i akceleratorami.
JMenu jmFile = new JMenu("Plik");
jmFile.setMnemonic(KeyEvent.VK_P);
JMenuItem jmiOpen = new JMenuItem("Otwórz", KeyEvent.VK_O);
jmiOpen.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_O, InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiClose = new JMenuItem("Zamknij", KeyEvent.VK_M);
jmiClose.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_M, InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiSave = new JMenuItem("Zapisz", KeyEvent.VK_Z);
jmiSave.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_Z, InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiExit = new JMenuItem("Zakończ", KeyEvent.VK_K);
jmiExit.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_K, InputEvent.CTRL_DOWN_MASK));

jmFile.add(jmiOpen);
jmFile.add(jmiClose);
jmFile.add(jmiSave);
jmFile.addSeparator();
jmFile.add(jmiExit);
jmb.add(jmFile);
JMenu jmOptions = new JMenu("Opcje");
//Utworzenie podmenu Kolory.
JMenu jmColors = new JMenu("Kolory");
//Tworzy elementy menu z polami wyboru i dodaje je do podmenu Kolory. Dzięki
//temu użytkownik będzie mógł wybrać więcej niż jeden kolor.
JCheckBoxMenuItem jmiRed = new JCheckBoxMenuItem("Czerwony");
JCheckBoxMenuItem jmiGreen = new JCheckBoxMenuItem("Zielony");
JCheckBoxMenuItem jmiBlue = new JCheckBoxMenuItem("Niebieski");
jmColors.add(jmiRed);
jmColors.add(jmiGreen);
jmColors.add(jmiBlue);
jmOptions.add(jmColors);
//Utworzenie podmenu Priorytet.
JMenu jmPriority = new JMenu("Priorytet");
//Tworzy elementy menu z przyciskami opcji i dodaje je do menu Priorytet.
```

```
//W ten sposób możemy pokazać aktualnie wybrany priorytet,
//a jednocześnie mamy pewność, że w dowolnej chwili będzie wybrany
//tylko jeden priorytet. Warto zwrócić uwagę, że początkowo zostanie
//zaznaczona opcja „Wysoki”.
JRadioButtonMenuItem jmiHigh =
new JRadioButtonMenuItem("Wysoki", true);
JRadioButtonMenuItem jmiLow =
new JRadioButtonMenuItem("Niski");
jmPriority.add(jmiHigh);
jmPriority.add(jmiLow);
jmOptions.add(jmPriority);
//Tworzy grupę przycisków dla elementów menu zawierających
//przyciski opcji.
ButtonGroup bg = new ButtonGroup();
bg.add(jmiHigh);
bg.add(jmiLow);
//Utworzenie opcji Resetuj.
JMenuItem jmiReset = new JMenuItem("Resetuj");
jmOptions.addSeparator();
jmOptions.add(jmiReset);
//Dodaje całe menu Opcje do paska menu.
jmb.add(jmOptions);

//Tworzy menu Pomoc.
JMenu jmHelp = new JMenu("Pomoc");
JMenuItem jmiAbout = new JMenuItem("O programie");
jmHelp.add(jmiAbout);
jmb.add(jmHelp);
//Dodaje wszystkie obiekty nasłuchujące.
jmiOpen.addActionListener(this);
jmiClose.addActionListener(this);
jmiSave.addActionListener(this);
jmiExit.addActionListener(this);
jmiRed.addActionListener(this);
jmiGreen.addActionListener(this);
jmiBlue.addActionListener(this);
jmiHigh.addActionListener(this);
jmiLow.addActionListener(this);
jmiReset.addActionListener(this);
jmiAbout.addActionListener(this);
//Tworzenie menu podręcznego Edycja.
jpu = new JPopupMenu();
//Tworzy elementy menu.
JMenuItem jmiCut = new JMenuItem("Wytnij");
JMenuItem jmiCopy = new JMenuItem("Kopiuj");
JMenuItem jmiPaste = new JMenuItem("Wklej");
//Dodaje elementy do menu.
jpu.add(jmiCut);
jpu.add(jmiCopy);
jpu.add(jmiPaste);
//Dodaje obiekt nasłuchujący sprawdzający wystąpienie warunku wyświetlenia
menu.
jfrm.addMouseListener(new MouseAdapter() {
public void mousePressed(MouseEvent me) {
if(me.isPopupTrigger())
jpu.show(me.getComponent(), me.getX(), me.getY());
}
public void mouseReleased(MouseEvent me) {
if(me.isPopupTrigger())
jpu.show(me.getComponent(), me.getX(), me.getY());
}
});
jmiCut.addActionListener(this);
jmiCopy.addActionListener(this);
```

```
jmiPaste.addActionListener(this);

JToolBar jtb = new JToolBar("Uruchamianie");
//Wczytuje obrazki.
ImageIcon set = new ImageIcon("setBP.gif");
ImageIcon clear = new ImageIcon("clearBP.gif");
ImageIcon resume = new ImageIcon("resume.gif");
//Tworzy przyciski paska narzędzi.
JButton jbtnSet = new JButton(set);
jbtnSet.setActionCommand("Ustaw punkt wstrzymania");
jbtnSet.setToolTipText("Ustaw punkt wstrzymania");
JButton jbtnClear = new JButton(clear);
jbtnClear.setActionCommand("Usuń punkt wstrzymania");
jbtnClear.setToolTipText("Usuń punkt wstrzymania");
JButton jbtnResume = new JButton(resume);
jbtnResume.setActionCommand("Wznów");
jbtnResume.setToolTipText("Wznów");
//Dodaje przyciski do paska narzędzi.
jtb.add(jbtnSet);
jtb.add(jbtnClear);
jtb.add(jbtnResume);
//Dodaje pasek narzędzi u góry obszaru zawartości
//(w pozycji określonej jako BorderLayout.NORTH).
jfrm.add(jtb, BorderLayout.NORTH);
//Dodanie obiektów nasłuchujących do przycisków paska narzędzi.
jbtnSet.addActionListener(this);
jbtnClear.addActionListener(this);
jbtnResume.addActionListener(this);
//Dodaje etykietę do panelu zawartości.
//jfrm.add(jlab);
jfrm.add(jlab, BorderLayout.CENTER);
//Dodaje pasek menu do ramki.
jfrm.setJMenuBar(jmb);
//Wyświetla ramkę.
jfrm.setVisible(true);
}
//Obsługa zdarzeń generowanych przez wybierane opcje menu.
public void actionPerformed(ActionEvent ae) {
//Pobranie łańcucha polecenia wybranej opcji menu.
String comStr = ae.getActionCommand();
//W przypadku wybrania opcji Zakończ należy zakończyć
//działanie programu.
if(comStr.equals("Zakończ")) System.exit(0);
//W pozostałych przypadkach wyświetlana jest nazwa opcji.
jlab.setText("Wybrano " + comStr);
}
public static void main(String args[]) {
//Tworzy ramkę w wątku obsługi zdarzeń.
SwingUtilities.invokeLater(new Runnable() {
public void run() {
new J_Menu();
}
});
}
}
```

Stosowanie akcji

Bardzo często zdarza się, że paski narzędzi oraz menu zawierają te same elementy. Na przykład możliwości funkcjonalne oferowane przez utworzony wcześniej pasek narzędzi *Uruchamianie* mogłyby także być dostępne za pośrednictwem opcji menu. W takim przypadku wybranie jednej z opcji (takiej jak ustawienie punktu wstrzymania) spowoduje wykonanie takiej samej akcji, niezależnie od tego, czy kliknięty został przycisk na pasku narzędzi, czy wybrany element menu. Oprócz tego w takiej sytuacji zarówno na przycisku, jak i w elemencie menu będzie (najprawdopodobniej) wyświetlona ta sama ikona. Co więcej, w przypadku dezaktywacji przycisku menu powinien zostać wyłączony także odpowiadający mu element menu. Takie sytuacje zazwyczaj powodowałyby powstawanie stosunkowo rozbudowanego, uzależnionego od siebie kodu, który raczej nie jest pożądany. W pakiecie Swing udostępnia rozwiązanie tego problemu; są nim **akcje.**

Akcja jest instancją interfejsu Action. Rozszerza interfejs ActionListener i umożliwia połączenie informacji o stanie z obsługą zdarzeń przy wykorzystaniu metody actionPerformed(). Takie połączenie pozwala jednej akcji na obsługę dwóch lub większej liczby komponentów. Akcje pozwalają na przykład scentralizować obsługę przycisków paska narzędzi oraz elementów menu i umożliwiają zarządzanie nimi. Zamiast powielać kod, program może jedynie utworzyć akcję, która będzie automatycznie obsługiwać dwa komponenty. Ponieważ interfejs Action rozszerza interfejs ActionListener, zatem akcja musi zawierać implementację metody actionPerformed(). Metoda ta będzie obsługiwać zdarzenia ActionEvent generowane przez wszystkie komponenty powiązane z daną akcją. Oprócz odziedziczonej metody actionPerformed() interfejs Action definiuje także kilka własnych metod. Szczególnie interesująca jest metoda putValue(). Służy ona do ustawiania wartości różnych właściwości skojarzonych z akcją. Poniżej przedstawiona została postać tej metody:

void putValue(String *key*, Object *val*)

Metoda ta zapisuje wartość *val* we właściwości, której nazwę określa parametr *key*. Warto wspomnieć, że interfejs Action definiuje także metodę getValue(), pozwalającą na pobranie wartości podanej właściwości.

Poniżej przedstawiona została postać metody getValue():

Object getValue(String *key*)

Metoda ta zwraca referencję do właściwości określonej przez parametr *key*.

Na przykład aby określić, że mnemoniką ma być litera *X*, należy użyć poniższego wywołania metody putValue():

actionOb.putValue(MNEMONIC_KEY, new Integer(KeyEvent.VK_X));

Jedyną właściwością akcji, która nie jest dostępna przy użyciu metod putValue() oraz getValue(), jest status aktywności. Aby go zmieniać, należy skorzystać z przedstawionych poniżej metod setEnabled() oraz isEnabled():

void setEnabled(boolean *enabled*) boolean isEnabled()

Jeśli chodzi o metodę setEnabled(), to przekazanie w jej wywołaniu wartości true sprawi, że akcja będzie aktywna. W przeciwnym razie zostanie ona wyłączona. Jeśli akcja jest aktywna, to wywołanie metody isEnabled() zwraca wartość true.

Choć cały interfejs Action można zaimplementować samodzielnie, to jednak zazwyczaj nie będzie to konieczne. Można bowiem skorzystać z klasy AbstractAction — rozszerzając ją, wystarczy zaimplementować tylko jedną metodę: actionPerformed(). Klasa AbstractAction udostępnia implementacje wszystkich pozostałych metod interfejsu Action. Klasa ta definiuje trzy konstruktory:

AbstractAction(String *name*, Icon *image*)

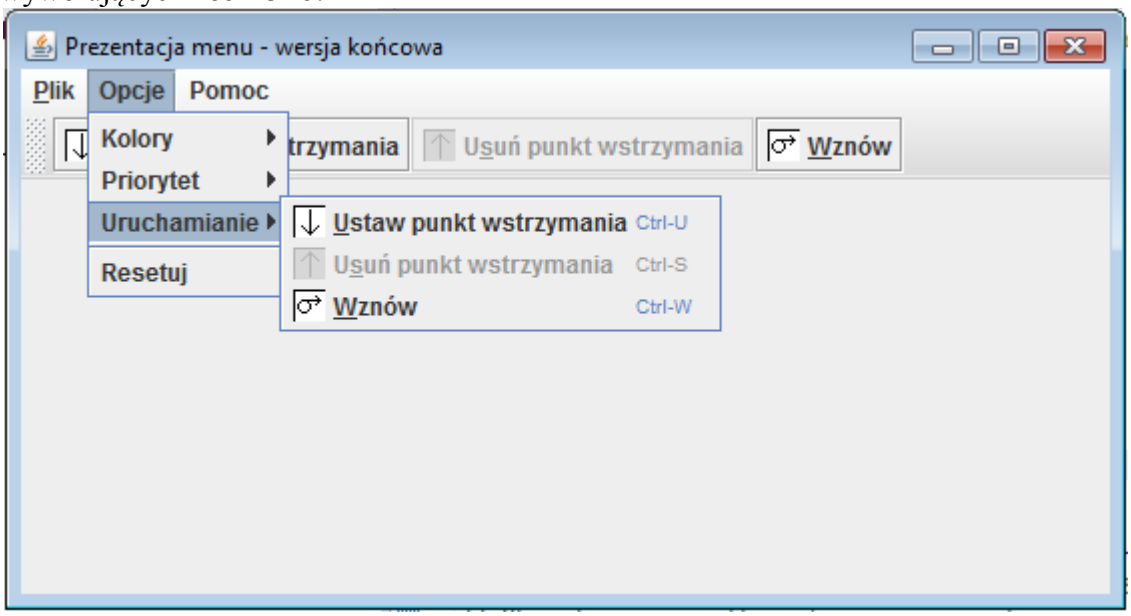
Konstruktor ten tworzy obiekt akcji, której nazwa jest określona przez parametr *name*, a ikona przez parametr *icon*. Po utworzeniu akcji można ją dodać do obiektu JToolBar oraz użyć do utworzenia obiektu JMenuItem. Aby dodać akcję do obiektu JToolBar, należy skorzystać z metody add():

void add(Action *actObj*)

Parametr *actObj* jest obiektem akcji dodawanej do paska narzędzi. Właściwości akcji

zostaną użyte do utworzenia przycisku paska narzędzi. Aby utworzyć element menu na podstawie akcji, należy użyć poniższego konstruktora klasy JMenuItem:
JMenuItem(Action *actObj*)
Także w tym przypadku parametr *actObj* jest obiektem akcji, której właściwości posłużą do utworzenia elementu menu.

Przykładowy program z zastosowanymi metodami grupowania akcji dla różnych obiektów wywołujących zdarzenia.




```
// Końcowa wersja programu MenuDemo.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class J_Menu implements ActionListener {
JLabel jlab;
JMenuBar jmb;
JToolBar jtb;
JPopupMenu jpu;

DebugAction setAct;
DebugAction clearAct;
DebugAction resumeAct;

J_Menu() {
// Tworzy kontener JFrame.
JFrame jfrm = new JFrame("Prezentacja menu - wersja końcowa");
// Program używa domyślnego menedżera układu - BorderLayout.
// Określenie początkowych wymiarów ramki.
jfrm.setSize(560, 300);
// Zamknięcie okna aplikacji ma kończyć działanie programu.
jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// Tworzy etykietę, w której będą wyświetlane nazwy
// wybieranych opcji menu.
jlab = new JLabel();
// Tworzy pasek menu.
jmb = new JMenuBar();
// Tworzy menu Plik.
makeFileMenu();
// Tworzy akcje paska narzędzi i menu Uruchamianie.
makeActions();
// Tworzy pasek narzędzi.
makeToolBar();
// Tworzy menu Opcje.
```

```
makeOptionsMenu();
// Tworzy menu Pomoc.
makeHelpMenu();
// Tworzy menu podręczne Edycja.
makeEditPUMenu();
// Dodaje obiekt nasłuchujący sprawdzający wystąpienie warunku wyświetlenia menu.
jfrm.addMouseListener(new MouseAdapter() {
public void mousePressed(MouseEvent me) {
if(me.isPopupTrigger())
jpu.show(me.getComponent(), me.getX(), me.getY());
}
public void mouseReleased(MouseEvent me) {
if(me.isPopupTrigger())
jpu.show(me.getComponent(), me.getX(), me.getY());
}
});
// Dodaje etykietę do centralnego obszaru panelu zawartości.
jfrm.add(jlab, SwingConstants.CENTER);
// Dodaje pasek narzędzi u góry obszaru zawartości
// (w pozycji określonej jako BorderLayout.NORTH).
jfrm.add(jtb, BorderLayout.NORTH);
// Dodaje pasek menu do ramki.
jfrm.setJMenuBar(jmb);
// Wyświetla ramkę.
jfrm.setVisible(true);
}

// Obsługa zdarzeń generowanych przez wybierane opcje menu.
// Ta metoda NIE obsługuje zdarzeń generowanych przez opcje
// paska narzędzi i menu Uruchamianie.
public void actionPerformed(ActionEvent ae) {
// Pobranie łańcucha polecenia wybranej opcji menu.
String comStr = ae.getActionCommand();
// W przypadku wybrania opcji Zakończ należy zakończyć
// działanie programu.
if(comStr.equals("Zakończ")) System.exit(0);
// W pozostałych przypadkach wyświetlana jest nazwa opcji.
jlab.setText("Wybrano " + comStr);
}
// Klasa akcji używana przez przyciski paska narzędzi
// i elementy podmenu Uruchamianie.
class DebugAction extends AbstractAction {
public DebugAction(String name, Icon image, int mnem,
int accel, String tTip) {
super(name, image);
putValue(ACCELERATOR_KEY,
KeyStroke.getKeyStroke(accel,
InputEvent.CTRL_DOWN_MASK));
putValue(MNEMONIC_KEY, new Integer(mnem));
putValue(SHORT_DESCRIPTION, tTip);
}
// Obsługuje zdarzenia generowane przez przyciski paska
// narzędzi i elementy menu Uruchamianie.
public void actionPerformed(ActionEvent ae) {
String comStr = ae.getActionCommand();
jlab.setText("Wybrano " + comStr);
// Przełącza status opcji Ustaw punkt wstrzymania
// oraz Usuń punkt wstrzymania.
if(comStr.equals("Ustaw punkt wstrzymania")) {
clearAct.setEnabled(true);
setAct.setEnabled(false);
} else if(comStr.equals("Usuń punkt wstrzymania")) {
clearAct.setEnabled(false);
setAct.setEnabled(true);
```

```
}
}
}
// Tworzy menu Plik wraz z mnemonikami i akceleratorami.
void makeFileMenu() {
JMenu jmFile = new JMenu("Plik");
jmFile.setMnemonic(KeyEvent.VK_P);
JMenuItem jmiOpen = new JMenuItem("Otwórz",
KeyEvent.VK_O);
jmiOpen.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_O,
InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiClose = new JMenuItem("Zamknij",
KeyEvent.VK_M);
jmiClose.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_M,
InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiSave = new JMenuItem("Zapisz",
KeyEvent.VK_Z);

jmiSave.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_Z,
InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiExit = new JMenuItem("Zakończ",
KeyEvent.VK_K);
jmiExit.setAccelerator(
KeyStroke.getKeyStroke(KeyEvent.VK_K,
InputEvent.CTRL_DOWN_MASK));
jmFile.add(jmiOpen);
jmFile.add(jmiClose);
jmFile.add(jmiSave);
jmFile.addSeparator();
jmFile.add(jmiExit);
jmb.add(jmFile);
// Dodaje obiekty nasłuchujące do elementów menu Plik.
jmiOpen.addActionListener(this);
jmiClose.addActionListener(this);
jmiSave.addActionListener(this);
jmiExit.addActionListener(this);
}
// Utworzenie menu Opcje.
void makeOptionsMenu() {
JMenu jmOptions = new JMenu("Opcje");
// Utworzenie podmenu Kolory.
JMenu jmColors = new JMenu("Kolory ");
// Tworzy elementy menu z polami wyboru i dodaje je do podmenu Kolory. Dzięki
// temu użytkownik będzie mógł wybrać więcej niż jeden kolor.
JCheckBoxMenuItem jmiRed = new JCheckBoxMenuItem("Czerwony");
JCheckBoxMenuItem jmiGreen = new JCheckBoxMenuItem("Zielony");
JCheckBoxMenuItem jmiBlue = new JCheckBoxMenuItem("Niebieski");
// Dodaje elementy do podmenu Kolory.
jmColors.add(jmiRed);
jmColors.add(jmiGreen);
jmColors.add(jmiBlue);
jmOptions.add(jmColors);
// Utworzenie podmenu Priorytet.
JMenu jmPriority = new JMenu("Priorytet");
// Tworzy elementy menu z przyciskami opcji i dodaje je do menu Priorytet.
// W ten sposób możemy pokazać aktualnie wybrany priorytet,
// a jednocześnie mamy pewność, że w dowolnej chwili będzie wybrany
// tylko jeden priorytet. Warto zwrócić uwagę, że początkowo zostanie
// zaznaczona opcja „Wysoki”.
JRadioButtonMenuItem jmiHigh =
new JRadioButtonMenuItem("Wysoki", true);
```

```
JRadioButtonMenuItem jmiLow =
new JRadioButtonMenuItem("Niski");
// Dodaje elementy do podmenu Priorytet.
jmPriority.add(jmiHigh);
jmPriority.add(jmiLow);
jmOptions.add(jmPriority);
// Tworzy grupę przycisków dla elementów menu zawierających
// przyciski opcji.
ButtonGroup bg = new ButtonGroup();
bg.add(jmiHigh);
bg.add(jmiLow);

// Tworzy podmenu Uruchamianie, umieszczone w menu
// Opcje. Elementy tego menu są tworzone na podstawie
// akcji.
JMenu jmDebug = new JMenu("Uruchamianie");
JMenuItem jmiSetBP = new JMenuItem(setAct);
JMenuItem jmiClearBP = new JMenuItem(clearAct);
JMenuItem jmiResume = new JMenuItem(resumeAct);
// Dodaje elementy do podmenu Uruchamianie.
jmDebug.add(jmiSetBP);
jmDebug.add(jmiClearBP);
jmDebug.add(jmiResume);
jmOptions.add(jmDebug);
// Tworzy opcję Resetuj.
JMenuItem jmiReset = new JMenuItem("Resetuj");
jmOptions.addSeparator();
jmOptions.add(jmiReset);
// W końcu całe menu Opcje jest dodawane do paska menu.
jmb.add(jmOptions);
// Określa obiekty nasłuchujące dla wszystkich elementów
// menu Opcje, z wyjątkiem elementów w podmenu
// Uruchamianie.
jmiRed.addActionListener(this);
jmiGreen.addActionListener(this);
jmiBlue.addActionListener(this);
jmiHigh.addActionListener(this);
jmiLow.addActionListener(this);
jmiReset.addActionListener(this);
}
// Tworzy menu Pomoc.
void makeHelpMenu() {
JMenu jmHelp = new JMenu("Pomoc");
// Dodaje ikonę do elementu menu.
ImageIcon icon = new ImageIcon("AboutIcon.gif");
JMenuItem jmiAbout = new JMenuItem("O programie", icon);
jmiAbout.setToolTipText("Informacje o programie MenuDemo.");
jmHelp.add(jmiAbout);
jmb.add(jmHelp);
// Określa obiekt nasłuchujący.
jmiAbout.addActionListener(this);
}
// Tworzy akcje używane do utworzenia i obsługi
// elementów paska narzędzi i podmenu Uruchamianie.
void makeActions() {
// Wczytuje obrazki używane przez akcje.
ImageIcon setIcon = new ImageIcon("setBP.gif");
ImageIcon clearIcon = new ImageIcon("clearBP.gif");
ImageIcon resumeIcon = new ImageIcon("resume.gif");
// Tworzy akcje.
setAct =
new DebugAction("Ustaw punkt wstrzymania",
setIcon,
KeyEvent.VK_U,
```

```
KeyEvent.VK_U,
"Ustawia punkt wstrzymania.");

clearAct =
new DebugAction("Usuń punkt wstrzymania",
clearIcon,
KeyEvent.VK_S,
KeyEvent.VK_S,
"Usuwa punkt wstrzymania.");
resumeAct =
new DebugAction("Wznów",
resumeIcon,
KeyEvent.VK_W,
KeyEvent.VK_W,
"Wznawia realizację programu.");
clearAct.setEnabled(false);
}
// Tworzy pasek narzędzi Uruchamianie.
void makeToolBar() {
// Tworzy przyciski paska narzędzi, używając przy tym akcji.
JButton jbtnSet = new JButton(setAct);
JButton jbtnClear = new JButton(clearAct);
JButton jbtnResume = new JButton(resumeAct);
// Tworzy pasek narzędzi Uruchamianie.
jtb = new JToolBar("Uruchamianie");
// Dodaje przyciski do paska narzędzi.
jtb.add(jbtnSet);
jtb.add(jbtnClear);
jtb.add(jbtnResume);
}
// Tworzenie menu podręcznego Edycja.
void makeEditPUMenu() {
jpu = new JPopupMenu();
// Tworzy elementy menu.
JMenuItem jmiCut = new JMenuItem("Wytnij");
JMenuItem jmiCopy = new JMenuItem("Kopiuj");
JMenuItem jmiPaste = new JMenuItem("Wklej");
// Dodaje elementy do menu.
jpu.add(jmiCut);
jpu.add(jmiCopy);
jpu.add(jmiPaste);
// Dodaje obiekty nasłuchujące do elementów menu podręcznego.
jmiCut.addActionListener(this);
jmiCopy.addActionListener(this);
jmiPaste.addActionListener(this);
}
public static void main(String args[]) {
// Tworzy ramkę w wątku obsługi zdarzeń.
SwingUtilities.invokeLater(new Runnable() {
public void run() {
new J_Menu();
}
});
}
}
```

Opracowano na podstawie:
HERBERT SCHILDT - „Java. Kompendium programisty. Wydanie IX” Wydawnictwo Helion.