DevCAST ZONE

Interfejs użytkownika

Wprowadzenie do platformy Java
i programowanie w języku Java

Spis treści

Biblioteka graficzna

- `AWT (Abstract Window Toolkit)`
  - AWT Java 1.1 – wzbogacona o model zdarzeń
- `JFC (Java Fundation Classes)`
  - Swing - część `JFC`, która skupia się na interfejsie graficznym

Elementy GUI

- Komponent – podstawowy element graficzny
- Kontener – komponent służący do osadzania innych komponentów lub innych kontenerów
- Zdarzenia – akcje, które mogą zachodzić dla konkretnego komponentu

Komponent

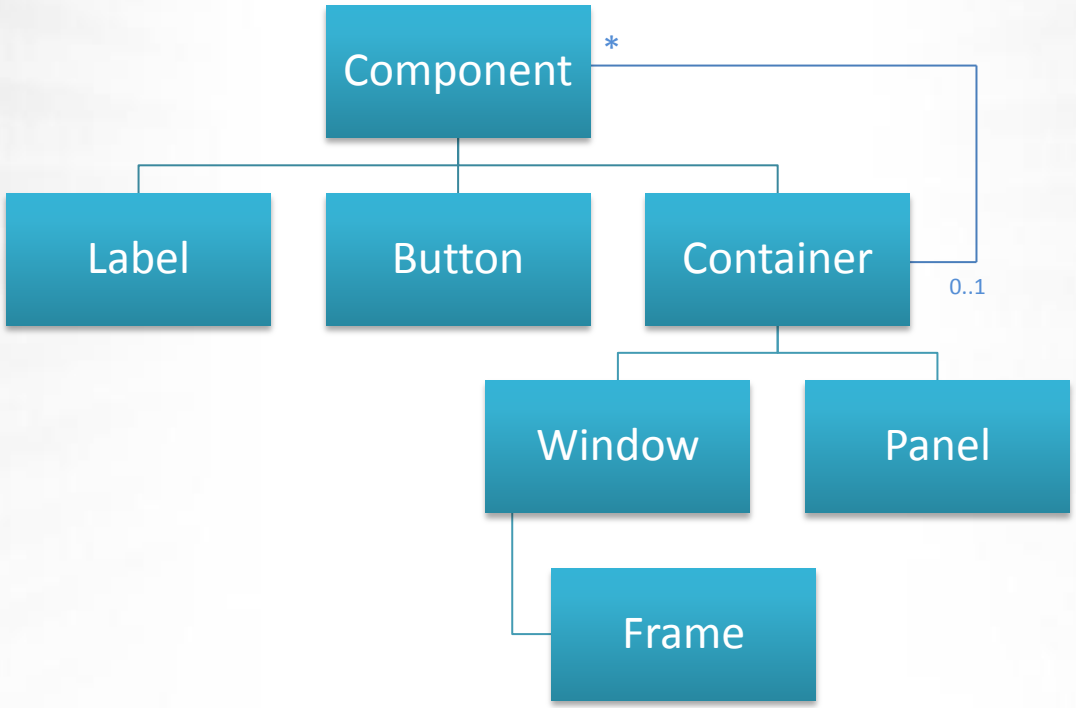
DevCAST ZONE

- Podstawowa klasa, z której tworzone są konkretne komponenty
- Do tworzenia interfejsu wykorzystuje się pochodne klasy Component np. `JButton`, `JCheckBox`, `JLabel`, `JList`, `JTextField`, `JTextArea`, `JMenu`
- Komponent może zarządzać wyglądem i zdarzeniami tworzonych elementów

Kontener

- Umieszczanie i wyświetlane komponenty (kontenery)
- Zaimplementowane metody służą głównie dodawaniu komponentów oraz zarządzaniu ich układem
- Do podstawowych typów kontenerów zaliczamy:
  - `JPanel` (będący również komponentem)
  - `JFrame`
  - `JWindow`
  - `JDialog`
  - `JScrollPane`

Relacje pomiędzy komponentami



Ramka (JFrame)

- `JFrame` pełni funkcję podstawowego elementu dla budowanych interfejsów – większość aplikacji bazuje właśnie na tej klasie
- Ramka generuje okno aplikacji typowe dla danej platformy systemowej wraz z paskiem tytułowym, menu systemowym oraz mechanizmem zarządzania rozmiarem okna

```
public class MyJFrame{
   public static void main(String a[]) {
            JFrame f = new JFrame("Tytuł okna");
            f.setSize(150, 150);
            f.setVisible(true);
   }
}
public class MyJFrame extends JFrame {
   public MyJFrame() {
            super("Tytuł okna");
            setSize(150, 150);
            setVisible(true);
   }
}
```

Menadżer ułożenia (zarządca rozkładu)

- Każdy kontener domyślnie posiada określony układ rozłożenia komponentów
- Układ definiowany jest poprzez obiekt – menadżera ułożenia (inaczej zarządcę rozkładu)
- W ramach danego kontenera wykorzystywany jest domyślny zarządca rozkładu – może być wyłączony i/lub zastąpiony innym

Menadżer ułożenia (zarządca rozkładu)

- Odpowiada za dokonanie rozmieszczenia komponentów wewnątrz kontenera z określonymi zasadami

- Podczas zmiany rozmiaru okna automatycznie dokonuje przebudowy okna

BorderLayout

- Domyślny menadżer rozkładu dla `JFrame` oraz `JDialog`
- Cztery rejony skrajne oraz jeden środkowy
- `add()` - dodanie elementu:
  - `BorderLayout.NORTH` - góra
  - `BorderLayout.SOUTH` - dół
  - `BorderLayout.WEST` - lewo
  - `BorderLayout.EAST` - prawo
  - `BorderLayout.CENTER` - środek obszaru
- Jeden komponent na obszar
- Umieszczane elementy dopasowują swój rozmiar do rozmiaru komórek

BorderLayout

DevCAST ZONE

```
public BorderLayoutFrame() {
        add(new JButton(„Środek"), BorderLayout.NORTH);
        add(new JButton(„Góra"), BorderLayout.SOUTH);
        add(new JButton(„Dół"), BorderLayout.EAST);
        add(new JButton(„Lewo"), BorderLayout.WEST);
        add(new JButton(„Prawo"), BorderLayout.CENTER);
        pack();
        setVisible(true);
}
```

GridLayout

- Podział elementów na kolumny i wiersze
- Dodanie elementu to utworzenie nowej komórki
- Dodawane elementy wpływają na rozmiar i zapełniają przestrzeń komórki

GridLayout


```
public GridLayoutFrame() {
        setLayout(new GridLayout(3,2));
        add(new JButton(„Komórka 1x1"));
        add(new JButton(„Komórka 1x2"));
        add(new JButton(„Komórka 2x1"));
        add(new JButton(„Komórka 2x2"));
        pack();
        setVisible(true);
}
```

FlowLayout

- Domyślny menadżer rozkładu dla `JPanel`
- Umieszcza komponenty po kolei (od lewej do prawej)
- Zapełnienie przestrzeni formatki powoduje przejście do kolejnego wiersza
- Komponenty zostają zeskalowane do ich najmniejszego rozmiaru

FlowLayout


```
public FlowLayoutFrame() {
        setLayout(new FlowLayout());
        add(new JButton(„Start"));
        add(new JButton(„Opcje"));
        add(new JButton(„Zakończ"));
        pack();
        setVisible(true);
}
```

Przykłady komponentów

- JButton
- JCheckBox
- JChoice
- JColorChooser
- JComboBox
- JDialog
- JFileChooser
- JLabel
- JList
- JMenu
- JMenuItem
- JPopupMenu
- JProgressBar
- JSlider
- JTextArea
- JTextField
- JToolBar

Menu

DevCAST ZONE

- `JMenuBar` – podstawa (uchwyt) budowanego menu
- `JMenuItem` – poszczególne elementy menu (np. w formie odbiorcy zdarzeń typu `Action`)
- Hierarchiczne menu poprzez dodawanie kolejnych elementów typu `Menu`

Zdarzenia

- Interakcja użytkownika z aplikacją np.
  - przesunięcie kursora myszki/okna,
  - naciśnięcie klawisza myszki/klawiatury.
  - wybór elementu z listy elementów,
  - upływ interwału czasowego itp.
- Komponenty dysponują swoimi zestawami zdarzeń
- Przetwarzanie zdarzenia przez obiekt wymaga rejestracji w źródle na wybrany typ zdarzenia (w celu powiadomienia o zdarzeniu) oraz implementacji odpowiedniego interfejsu:

```
ActionHandler ac = new ActionHandler();
Button button = new Button(„Naciśnij");
button.addActionListener(ac);

public class ActionHandler implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        System.out.println(„Naciśnięto!");
    }
}
```

Przykładowe źródła zdarzeń

- `ActionEvent` – akcje klawiatury i myszki, wyboru pozycji menu
- `AdjustmentEvent` – akcje paska przewijania
- `ContainerEvent` – zdarzenie generowane przy dodaniu lub usunięciu komponentu w kontenerze
- `FocusEvent` – generowanie zdarzenia przy uzyskaniu lub utracie sterowania komponentu
- `ItemEvent` – akcje zaznaczenia lub wyboru pozycji z menu oraz kliknięcia pola wyboru
- `TextEvent` – akcje pól obraz obszarów tekstowych (wprowadzanie tekstu)
- `WindowEvent` – zdarzenie generowanie przez zmiany stanów okna np. minimalizacja, maksymalizacja, aktywacja.

Zdarzenia złożone

- Obsługa zdarzenia przez więcej niż jedną metodę
- Konieczność implementacji wszystkich niezbędnych metod związanych z obsługą
- Interfejsy zdarzeń mogą zostać dowolnie zaimplementowane w klasie do obsługi zdarzeń