Programowanie komputerów

Wykład 3: „Programowanie graficznych interfejsów użytkownika w Javie”

dr inż. Walery Susłow

walery.suslow@tu.koszalin.pl

Graficzny interfejs użytkownika

GUI (Graphical User Interface), środowisko graficzne – jest to ogólne określenie sposobu prezentacji informacji przez komputer oraz interakcji z użytkownikiem, polegającego na rysowaniu i obsługiwaniu widget'ów (kontrolek).

Interfejs graficzny został wymyślony przez pracowników laboratorium PARC firmy Xerox. W Polsce pierwsze GUI powstały w latach 70. do maszyn Odra.

Środowisko graficzne zapewnia alternatywny dla konsoli sposób pracy na komputerze.

Słownictwo związane z projektowaniem GUI

Komponenty – elementy wyświetlane (widget'y).

Kontener – obszar zawierający widget'y.

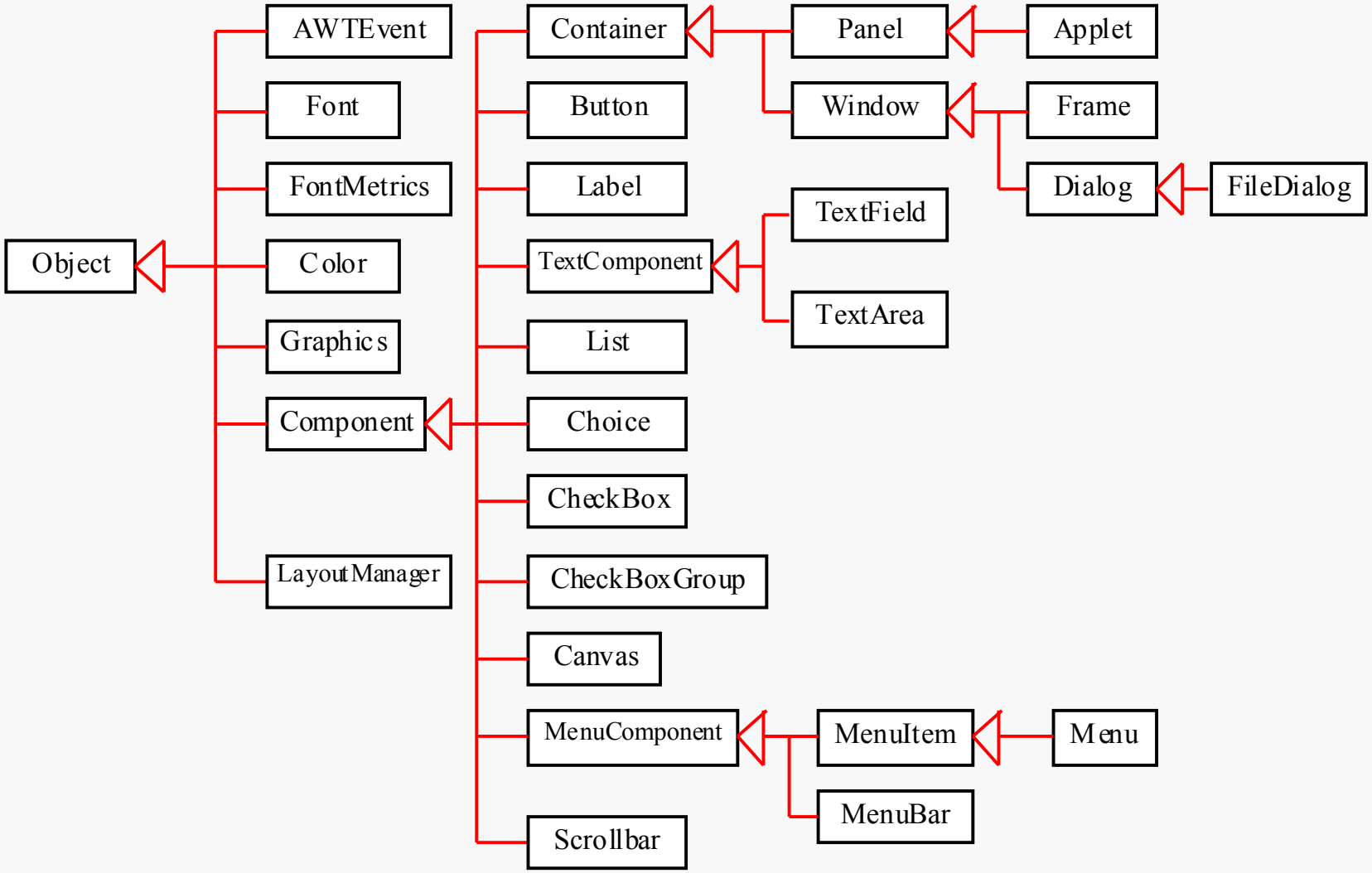
Układ (Layout) – sposób aranżacji kontenera.

Zdarzenia (Events) – interakcje użytkownika z widget'ami.

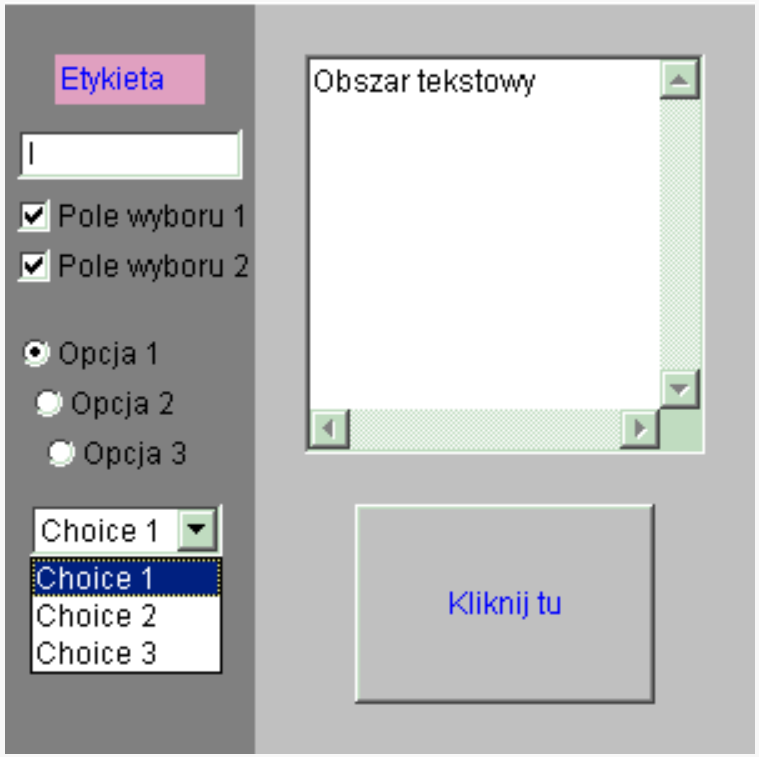
Ustawienia regionalne (Locale)

Pakiety lokalizacyjne (Resource Bundle)

Hierarchia klas GUI: biblioteka AWT

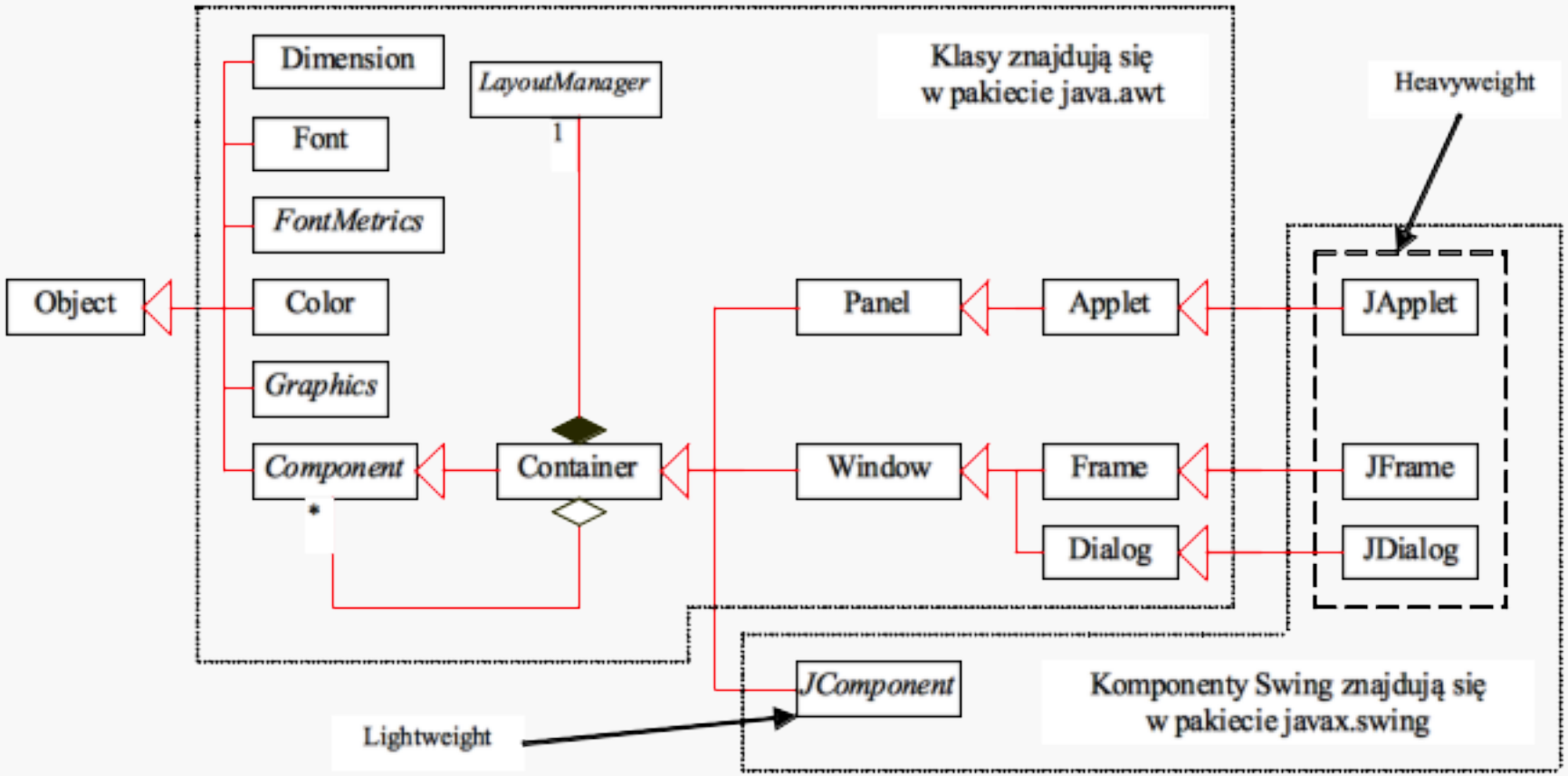


Komponenty „Abstract Windowing Toolkit”

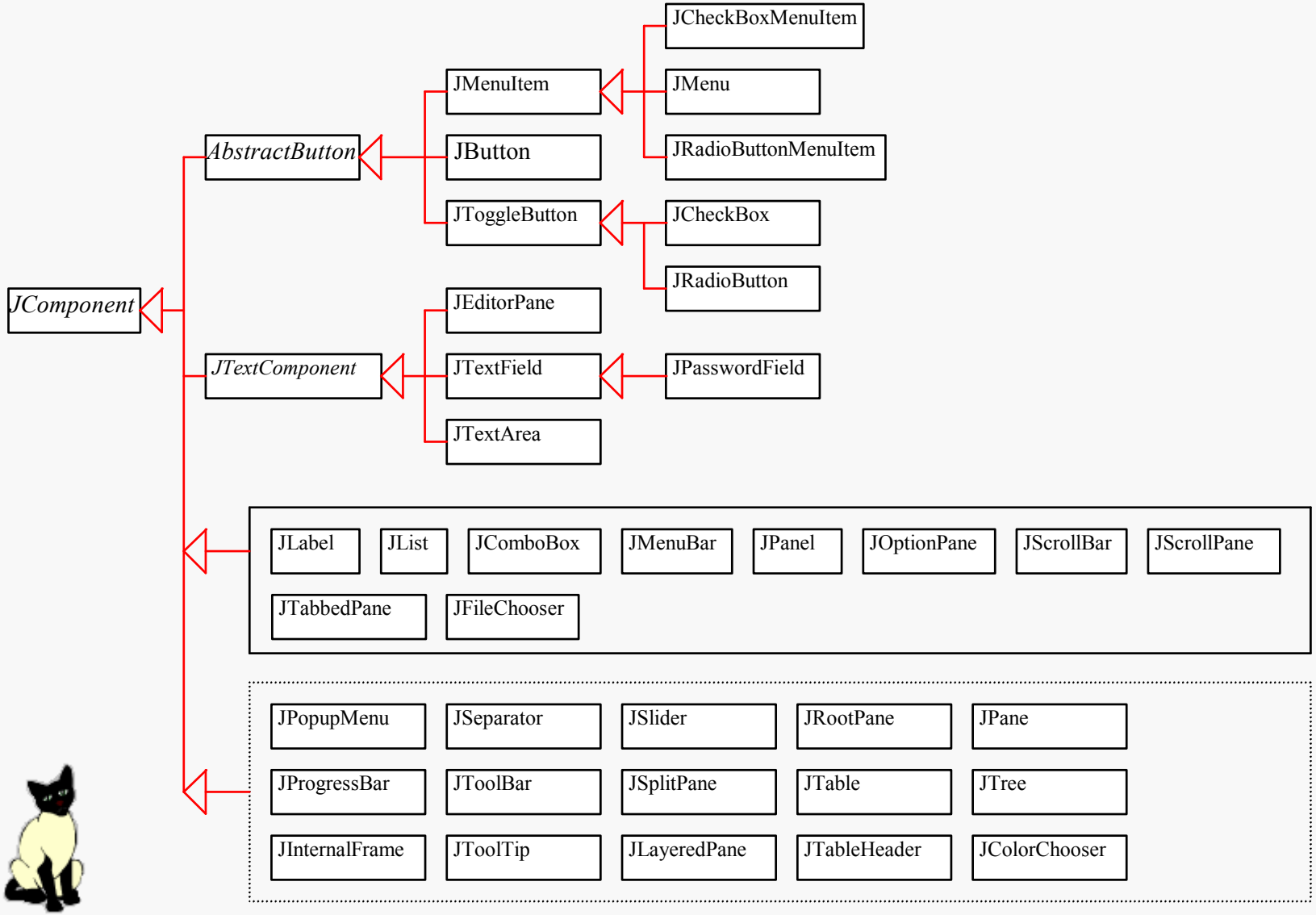


- AWT zawiera kompletny zestaw klas, służących do tworzenia GUI.
- Poszczególne elementy GUI to komponenty, takie jak przyciski, listy, etykiety czy pola wyboru. Są one reprezentacją klas wywodzących się z pakietu java.awt.Component.

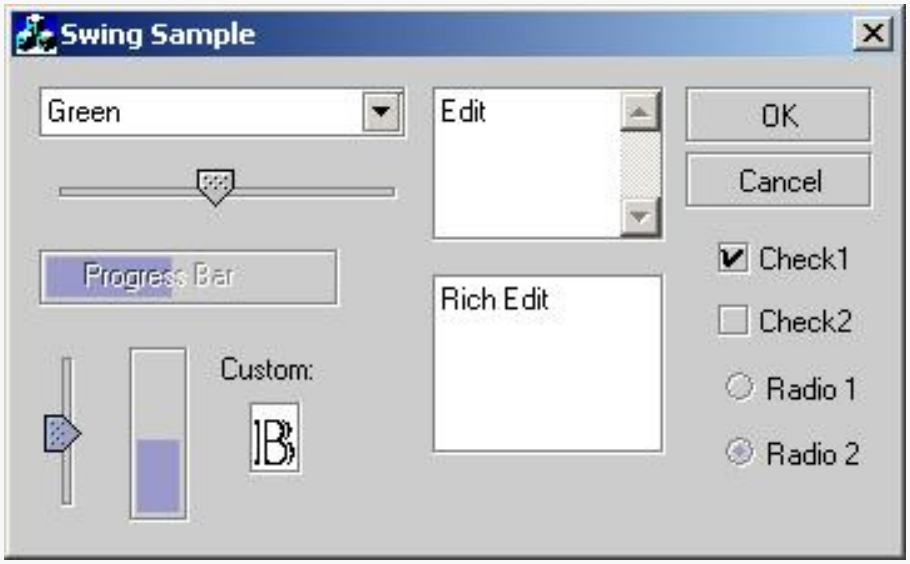
Hierarchia klas GUI: biblioteka Swing

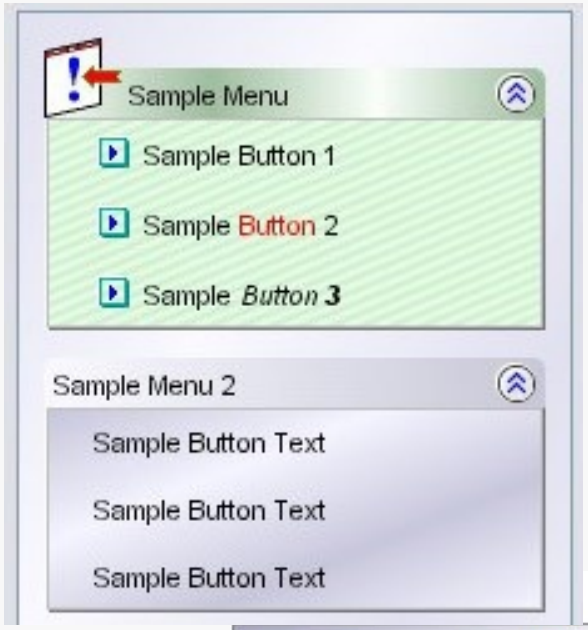


Hierarchia klas Swing (JComponent)



Komponenty Swing





- Obecnie Swing jest podstawową biblioteką służącą do tworzenie interfejsów użytkownika w Javie.
- Swing składa się z komponentów lekkich. Mają one tą zaletę że aplikacja wyglądała tak samo pod różnymi systemami operacyjnymi i na dodatek można zmieniać styl interfejsu podczas pracy z programem.

Główne pakiety Java GUI

Główne komponenty Java GUI:

- java.awt.Component
- java.awt.Container
- javax.swing.Jcomponent

Manager układu komponentów GUI:

- java.awt.LayoutManager
- java.awt.LayoutManager2

Obsługa zdarzeń:

- java.awt.event
- javax.swing.event

Pakiety lokalizacyjne dla różnych ustawień regionalnych:

- java.util.ResourceBundle

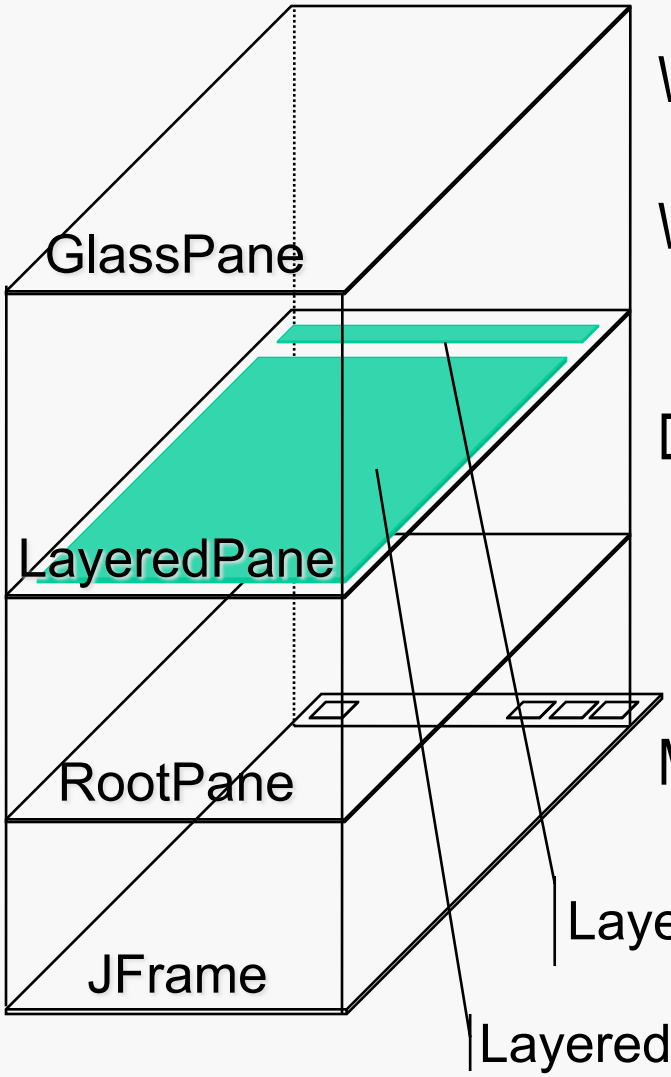
Ramka (Frame) podstawowym komponentem GUI

- Frame to okno, które nie zawiera wewnątrz innych okien.
- Frame jest to fundament na którym umieszczane są inne komponenty graficznego interfejsu użytkownika danej aplikacji.
- Klasa Frame może być wykorzystana do tworzenia okna.
- W aplikacjach Java, wykorzystujących bibliotekę Swing, klasa JFrame służy do budowania okien.

Kod ramki testowej

```
import javax.swing.*;
public class MyFrame {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Test Frame");
    frame.setSize(400, 300);
    frame.setVisible(true);
    frame.setDefaultCloseOperation(
      JFrame.EXIT_ON_CLOSE);
  }
 }
```

Struktura warstw ramki JFrame



Większość elementów GUI jest umieszczana w obszarze ContentPane.

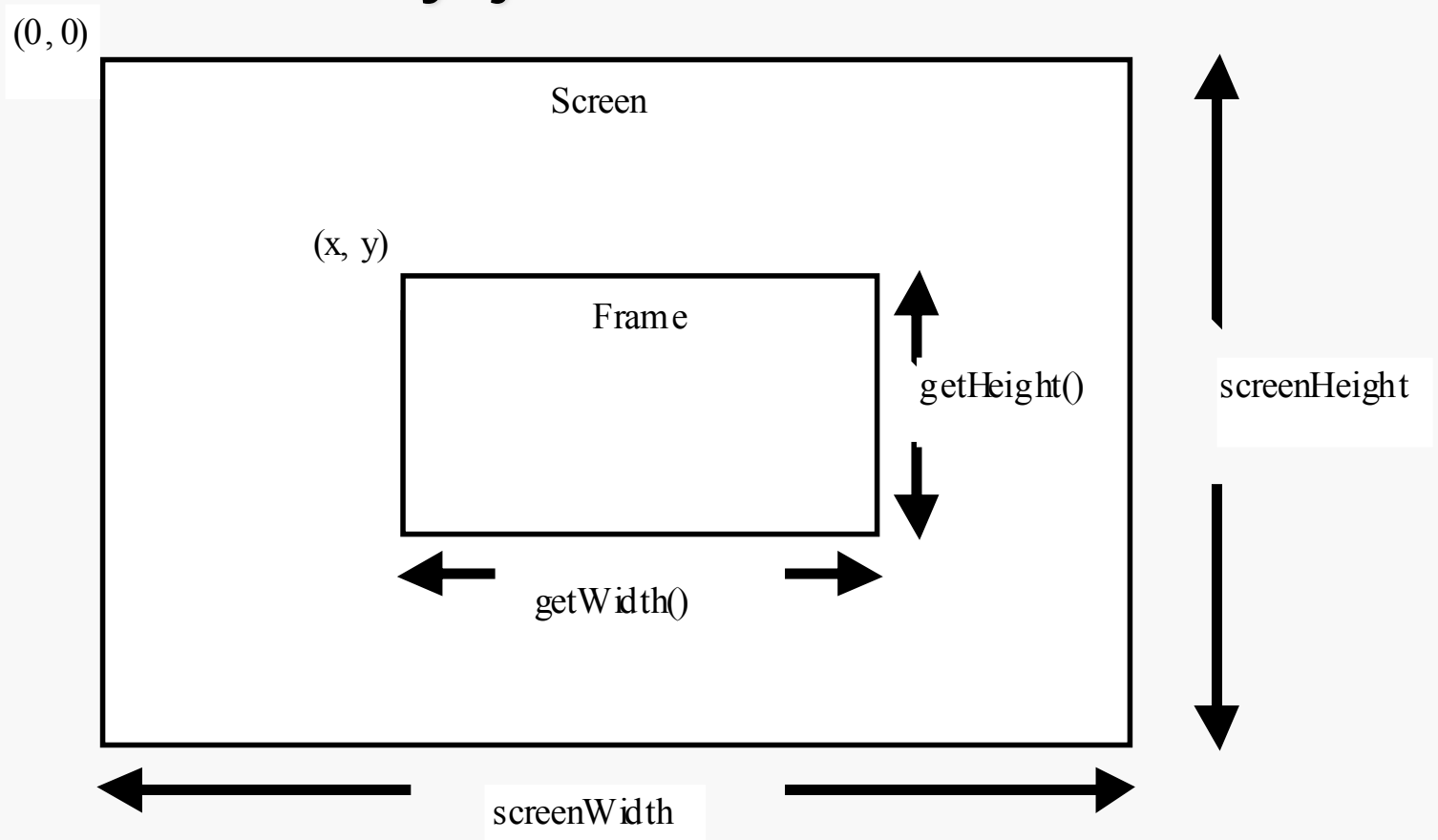
Warstwa GlassPane jest wykorzystywana do „pop up” menu, oraz do niektórych animacji.

Dostępne są metody:

```
getRootPane();
getLayeredPane();
getContentPane();
getGlassPane();
```

Można zarządzać widocznością warstw.

Pozycja ramki na ekranie



Domyślnie ramka (Frame) jest pokazywana w lewym górnym rogu ekranu. Żeby pokazać ramkę w innym miejscu ekranu można użyć metody setLocation(x, y) z klasy JFrame.

Dodawanie komponentów do ramki

```
// Dodajemy przycisk do ramki
frame.getContentPane().add(new JButton("OK"));
```

- Content Pane jest podklasą klasy Container. Dlatego, poprzedni kod można zamienić na:

```
Container container = frame.getContentPane();
container.add(new JButton("OK"));
```

- Obiekt klasy Container jest budowany gdy buduje się obiekt JFrame.
- JFrame wykorzystuje ContentPane żeby utrzymać komponenty wewnątrz ramki.

Menedżer układu

- Klasy Javy umożliwiające automatyczne rozmieszczanie komponentów nazywamy menedżerami układu (Layout Managers).
- Menedżer układu zapewnia właściwy poziom abstrakcji, umożliwiając automatyczne mapowanie okna interfejsu użytkownika na wszystkich systemach.
- Elementy GUI są umieszczone w kontenerach. Każdy kontener posiada własny menedżer układu do zarządzenia elementami GUI wewnątrz siebie.

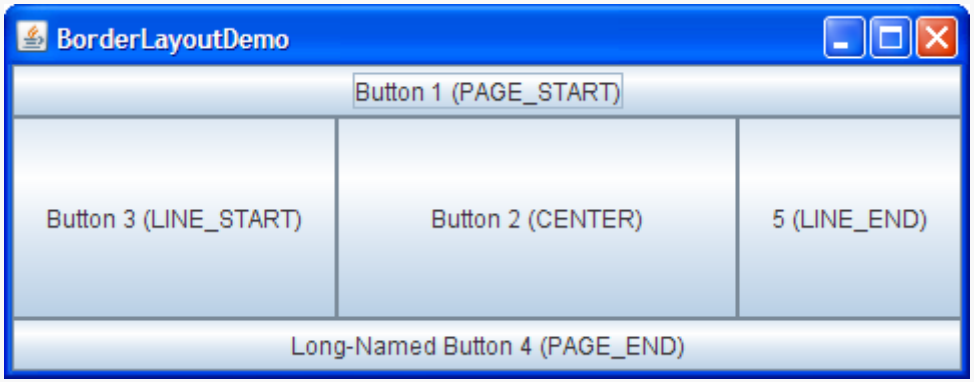
Menedżer układu, cd.

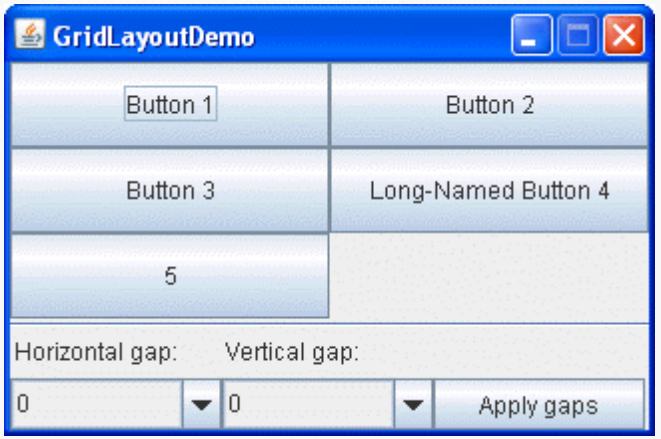
Sposób ułożenia komponentów w oknie aplikacji zależy od użytej klasy menedżera. Pakiet AWT udostępnia pięć takich klas:

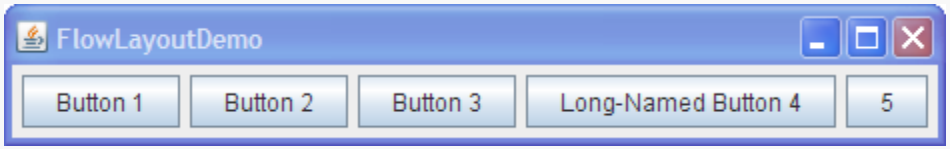
- FlowLayout - układ ciągły;
- GridLayout - układ siatkowy;
- BorderLayout - układ brzegowy;
- CardLayout - układ kartkowy;
- GridBagLayout - układ torebkowy.

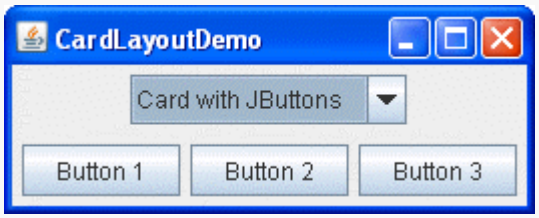
Menedżera układu można pominąć, wywołując metodę setLayout(null). Następnie należy własnoręcznie określić współrzędne i rozmiary poszczególnych komponentów i włożonych kontenerów.

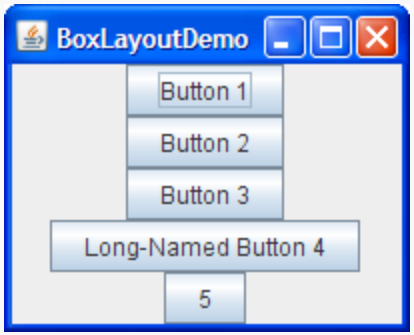
http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

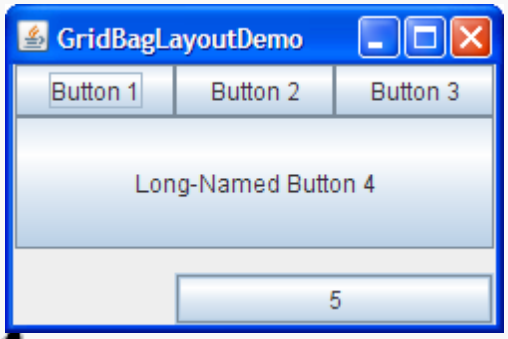




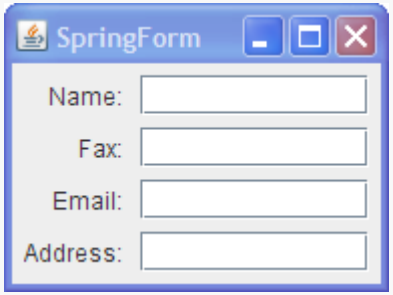




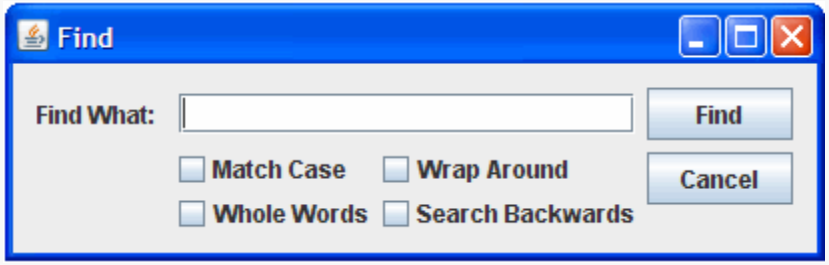




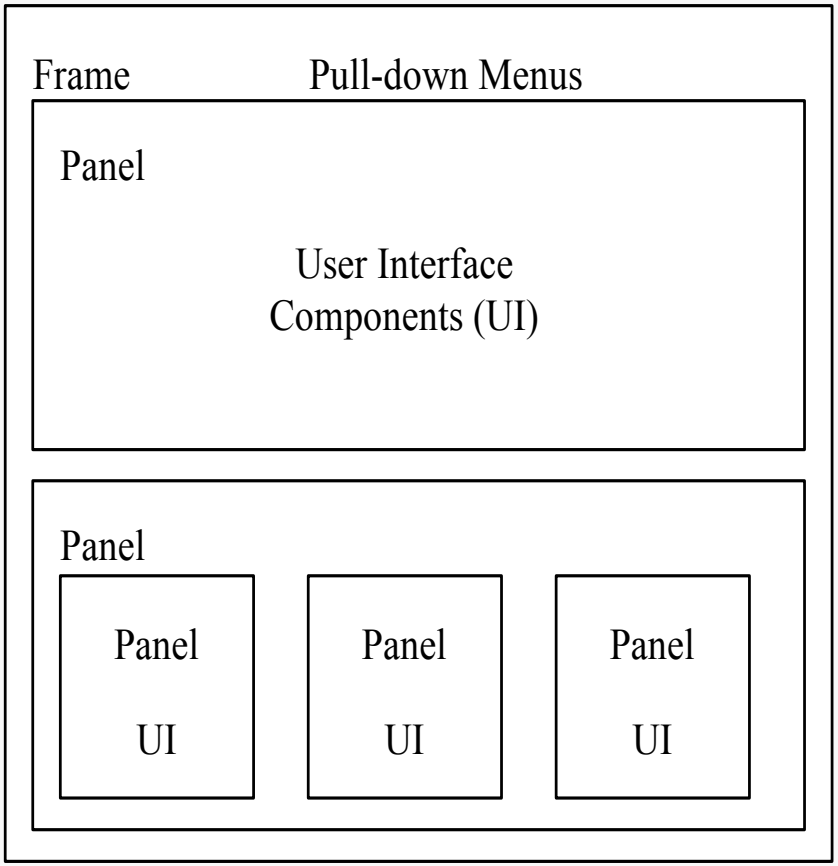
SpringLayout

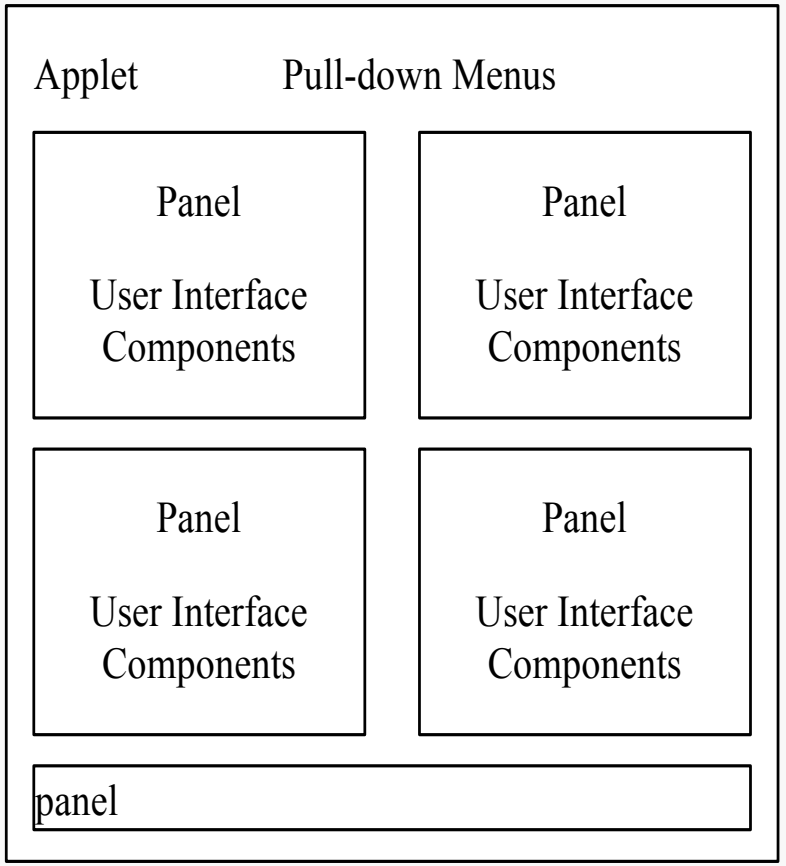


GroupLayout



Budowanie GUI aplikacji i apletu Java





Komponenty i kontenery

przycisk (Button)
płótno (Canvas)
pole wyboru (Checkbox)
etykieta (Label)
pasek przewijania (ScrollBar)
lista (List)
lista rozwijana (Choice)
polecenia menu (MenuItem)
pole tekstowe (TextField)
obszar tekstowy (TextArea)

kontener (Container)
okno (Window)
ramka (Frame)
okno dialogowe (Dialog)
dialog plikowy (FileDialog)
panel (Panel)
okna przewijane (ScrollPane)

Klasa Component

W klasie Component zdefiniowane są metody wspólne dla wszystkich komponentów. Metod tych używa się do pobierania (get), ustawiania (set) i sprawdzania (is) właściwości komponentów AWT.

Size rozmiar komponentu	*getSize()* *getSize(Dimension rozmiar)* *setSize(int width, int height)*
Location położenie	*getLocation()* *getLocation(Point p)* *setLocation(Point p)*
Bounds rozmiar i położenie	*getBounds() lub getBounds(Rectangle r)* *setBounds(x, y, width, height)* *setBounds(Rectangle r)*

Metody klasy Component

Font pismo	*getFont()* *setFont(Font f)*
Background kolor tła	*getBackground()* *setBackground(Color c)*
Foreground kolor pierwszego planu	*getForeground()* *setForeground(Color c)*
Visible widzialność	*isVisible()* *setVisible(boolean b)*
Enabled dostępność	*isEnabled()* *setEnabled(boolean b)*

Wyrównywanie komponentów

Do wyrównywania komponentów używane są następujące stałe, zdefiniowane w klasie Component:

- BOTTOM_ALIGNMENT
- CENTER_ALIGNMENT
- LEFT_ALIGNMENT
- RIGHT_ALIGNMENT
- TOP_ALIGNMENT

Zdarzenia GUI

Graficzny interfejs użytkownika powinien reagować na zdarzenia, pochodzące od manipulatora czy klawiatury użytkownika. Źródłami i słuchaczami zdarzeń są obiekty:

- Zdarzenie (Event) – obiekt przechowujący informację o stanie źródła;
- Źródło (Source) – obiekt, który generuje zdarzenie;
- Słuchacz (Listener) – obiekt powiadamiany o wystąpieniu zdarzenia.

Zdarzenia GUI, cd.

Każdy obiekt-słuchacz, który ma reagować na zdarzenia, musi spełniać dwa wymogi:

- być zarejestrowany na liście słuchaczy zdarzeń.

Rejestracji dokonuje się za pomocą metody addxxxListener(), wywoływanej na rzecz komponentu inicjującego zdarzenie, gdzie "xxx" reprezentuje typ nasłuchiwanego zdarzenia;

- mieć zaimplementowany odpowiedni interfejs nasłuchu:
  - ActionListener (dla przycisku i pola tekstowego),
  - Adjustable (dla paska przewijania) oraz,
  - ItemListener (dla pól wyboru i list).

Klasa Button

Do tworzenia przycisków wykorzystuje się konstruktor klasy Button:

- Button(); //Tworzy pusty przycisk (bez napisu).
- Button(String); //Tworzy przycisk z napisem.

Użyteczne metody klasy Button:

- setLabel(String); //Ustawia tekst na przycisku.
- getLabel(); //Pobiera tekst wyświetlany na przycisku.

Zarządzanie nasłuchem przycisku:

- addActionListener(); //Rejestruje słuchacza zdarzeń.
- removeActionListener(); //Wyrejestrowuje słuchacza zdarzeń.

Przycisk i zdarzenia

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Ok extends Applet implements ActionListener {
  Button button; String s;
  public void init() {
     s="";
     button = new Button("OK");
     button.addActionListener(this);
     add(button);
  }
  public void paint(Graphics g)        {
     g.drawString("" + s, 80, 70);
  }
  public void actionPerformed(ActionEvent e) {
     s = "" + Math.random();
     repaint();
  }  //Możemy sprawdzić, który z przycisków wygenerował
}    //zdarzenie: if (e.getSource() == button1) { ...}
```

Klasa Label

Klasa Label służy do tworzenia etykiet umożliwiających wyświetlanie tekstu jednowierszowego. Etykiety są często wykorzystywane do opisywania komponentów GUI, więc tekst wyświetlany na nich nie może być modyfikowany przez użytkownika.

Instrukcja tworząca etykietę ma postać:

- Label label = new Label("Etykieta", Label.CENTER);
- add(label);

Wyrównanie tekstu etykiety poprzez wykorzystanie stałych Java:

- Label.LEFT;
- Label.RIGHT;
- Label.CENTER.

Własności etykiet

Aby utworzyć etykietę o zasięgu globalnym należy najpierw zadeklarować na poziomie klasy referencję do obiektu klasy Label, a następnie wewnątrz metody inicjującej utworzyć obiekt klasy Label i ustawić jego początkowe własności.

```
public class Elementy extends Applet
{
   Label label;
   public void init()
   {  setLayout(null);
      label = new Label("Etykieta", Label.CENTER);
      label.setBackground(new Color(96,148,148));
      label.setForeground(Color.orange);
      label.setFont(new Font("Arial", Font.BOLD, 16));
      label.setLocation(20,20);
      label.setSize(80,20);
      add(label); // dodanie etykiety do apletu
```

Komponenty tekstowe

Pola tekstowe i obszary tekstowe to podklasy klasy TextComponent.

Pola tekstowe – obiekty klasy TextField – umożliwiają wprowadzanie i modyfikowanie tekstu, ale ich rozmiar jest ograniczony do jednego wiersza.

Do wprowadzania większych, wielowierszowych tekstów stosuje się obszary tekstowe - obiekty klasy TextArea.

Za pomocą odpowiednich stałych można wyposażyć obszar tekstowy w żądane paski przewijania:

- *TextArea.SCROLLBARS_BOTH;*
- *TextArea.SCROLLBARS_HORIZONTAL_ONLY;*
- *TextArea.SCROLLBARS_NONE ;*
- *TextArea.SCROLLBARS_VERTICAL_ONLY* ;

Konstruktory klasy TextField

Konstruktor	Opis
`TextField()`	Tworzy nowe puste pole tekstowe
`TextField(int)`	Tworzy nowe puste pole tekstowe o podanej liczbie kolumn
`TextField(String)`	Tworzy nowe pole tekstowe zawierające podany tekst
`TextField(String, int)`	Tworzy nowe pole tekstowe zawierające podany tekst o podanej liczbie kolumn

Metody klasy TextField

Metoda	Opis
`echoCharIsSet()`	Sprawdza, czy pole tekstowe wyświetla zaszyfrowane informacje.
`getColumns()`	Zwraca liczbę widocznych kolumn
`getEchoChar()`	Zwraca znak zastępujący wprowadzane litery (szyfrowanie tekstu)
`setColumns(int)`	Ustawia liczbę widocznych kolumn
`setEchoChar(char)`	Ustawia znak do szyfrowania
`setText(String)`	Ustawia domyślny tekst w polu tekstowym

Konstruktory klasy TextArea

Konstruktor	Opis
`TextArea()`	Tworzy nowy pusty obszar tekstowy.
`TextArea(int w, int k)`	Tworzy nowy pusty obszar tekstowy o określonej liczbie wierszy i kolumn.
`TextArea(String tekst)`	Tworzy nowy obszar tekstowy zawierający podany tekst.
`TextArea(String tekst, int w, int k)`	Tworzy nowy obszar tekstowy zawierający podany tekst, o określonej liczbie wierszy i kolumn.
`TextArea(String tekst, int w, int k, int paski)`	Tworzy nowy obszar tekstowy zawierający podany tekst, o określonej liczbie wierszy i kolumn, z podanymi paskami przewijania.

Metody klasy TextArea

Metoda	Opis
`append(String s)`	Dołącza podany tekst na koniec obszaru tekstowego
`getColumns()`	Zwraca liczbę kolumn obszaru tekstowego
`getRows()`	Zwraca liczbę wierszy obszaru tekstowego
`getScrollbarVisibility()`	Zwraca informację o paskach przewijania
`insert(String s, int n)`	Wstawia podany tekst na podanej pozycji
`replaceRange(String s, int start, int end)`	Zastępuje podanym tekstem informację w określonym zakresie
`setColumns(int c)`	Ustawia liczbę wyświetlanych kolumn
`setRows(int r)`	Ustawia liczbę wyświetlanych wierszy

Klasa ScrollBar

Klasa Scrollbar służy do tworzenia pasków przewijania. Komponenty te pozwalają na ustawianie wartości liczbowych za pomocą myszki.

Wyróżniamy dwa rodzaje pasków przewijania:

- Scrollbar.HORIZONTAL
- Scrollbar.VERTICAL

Każda zmiana ustawienia paska przewijania generuje zdarzenie typu Adjustment. Jeśli klasa ma przechwytywać zdarzenia tego typu, powinna implementować interfejs AdjustmentListener. Włącza on tylko jedną metodę:

```
public void adjustmentValueChanged(AdjustmentEvent  e)
    {
    }
```

Konstruktory klasy Scrollbar

Konstruktor	Opis
Scrollbar( )	Tworzy nowy pionowy pasek przewijania
Scrollbar(int)	Tworzy nowy pionowy pasek przewijania o określonej orientacji
Scrollbar (int orientacja, int wartość, int rozmiar, int min, int max)	Tworzy nowy pionowy pasek przewijania o podanej orientacji, wartości początkowej, rozmiarze ruchomego elementu, wartości minimalnej i wartości maksymalnej

Metody klasy Scrollbar

Metoda	Opis
getMaximum( )	Zwraca maksymalną możliwą wartość
getMinimum( )	Zwraca minimalną możliwą wartość
getOrientation( )	Zwraca orientację paska
getValue( )	Zwraca wartość bieżącą paska
getBlockIncrement( )	Zwraca wartość blokowego skoku
getUnitIncrement( )	Zwraca wartość jednostkowego skoku
setOrientation(int orientacja)	Ustawia orientację paska
setValue(int wartość)	Ustawia wartość bieżącą na pasku

Klasa Checkbox

- Abstract Windowing Toolkit zawiera klasę Checkbox, która umożliwia tworzenie pól wyboru.
- Komponenty te mogą znajdować się w jednym z dwóch stanów - "wybrany" (true) albo "niewybrany" (false). Klikając na polu wyboru użytkownik może łatwo zmienić jego stan na przeciwny.
- Pola te są niezależne od siebie i użytkownik może dokonać wyboru wielokrotnego, tzn. wiele z tych pól może być ustawionych w pozycji "wybrany".
- Możliwe też jest tworzenie grup opcji za pomocą klasy CheckboxGroup, która wymusza pojedynczy wybór.
- Kiedy pola są zebrane w grupę, traktowane są jako przyciski radiowe (wzajemnie się wykluczające) i tylko jeden z nich może być wybrany.

Konstruktory klasy Checkbox

Konstruktor	Opis
Checkbox()	Tworzy puste pole wyboru
Checkbox(String opis)	Tworzy pole wyboru z określoną etykietą opisu
Checkbox(String opis, boolean stan)	Tworzy pole wyboru z etykietą opisu oraz ustala jego stan
Checkbox(String opis, boolean stan, CheckboxGroup grupa)	Tworzy pole wyboru z etykietą opisu, ustala jego stan oraz przypisuje je do danej grupy
Checkbox(String opis, CheckboxGroup grupa, boolean stan)	Tworzy pole wyboru z etykietą opisu, przydziela je do danej grupy oraz ustala jego stan.

Metody klasy Checkbox

Metoda	Opis
getCheckboxGroup()	Pobiera informację, do jakiej grupy należy dane pole wyboru
getLabel()	Pobiera etykietę pola wyboru
getSelectedObjects()	Zwraca jednoelementową tablicę zawierającą etykietę pola lub null jeśli pole nie jest zaznaczone
getState()	Sprawdza, czy pole jest zaznaczone
setCheckboxGroup(CheckboxGroup g)	Dodaje pole wyboru do podanej grupy
setLabel(String etykieta)	Ustawia etykietę pola wyboru
setState(boolean stan)	Ustawia stan pola wyboru

Listy i listy rozwijane

- Listy są wykorzystywane w GUI, w którym użytkownik może dokonywać wyboru spośród większej liczby możliwych opcji.
- Jeśli mamy niewiele miejsca na GUI, możemy utworzyć listę rozwijaną — komponent klasy Choice.

```
List l =new List();
l.add("zielony");
l.add("czerwony");
l.add("niebieski");
add(l);
```

```
Choice c =new Choice();
c.addItem("zielony");
c.addItem("czerwony");
c.addItem("niebieski");
c.addItem("żółty");
c.addItem("bronzowy");
add(c);
```

Metody klasy Choice

Metoda	Opis
add(String element)	Dodaje składnik do listy
addItem(String element)	Dodaje składnik do listy
getItem(int numer)	Zwraca napis odpowiadający elementowi o podanym numerze
getItemCount()	Zwraca liczbę elementów listy
getSelectedIndex()	Zwraca numer zaznaczonego elementu
getSelectedItem()	Zwraca napis odpowiadający zaznaczonemu elementowi
getSelectedObjects()	Zwraca jednoelementową tablicę zawierającą zaznaczony element

Metody klasy Choice

Metoda	Opis
insert(String element, int numer)	Wstawia nowy element na podanej pozycji
remove(int pozycja)	Usuwa element o podanej pozycji
remove(String elem)	Usuwa pierwszy element wyznaczony przez podany napis
removeAll()	Usuwa wszystkie elementy
select(int nr)	Wybiera element o podanym numerze
select(String s)	Wybiera element którego opis jest identyczny jak podany