Język JAVA
podstawy programowania

Jacek Rumiński

Wykład 5, część 1

Plan wykładu:

1. Wprowadzenie do grafiki w Javie
2. Budowa GUI: komponenty, kontenery i układanie komponentów
3. Budowa GUI: obsługa zdarzeń
4. Grafika wektorowa – porysujmy sobie
5. Reprezentacja koloru
6. Grafika rastrowa - obrazy
7. Obsługa czcionek

1. Wprowadzenie do grafiki w Javie

Grafika w Javie związana jest trzema podstawowymi elementami:

- tworzenie interfejsu graficznego użytkownika (GUI),
- rysowanie z wykorzystaniem grafiki wektorowej,
- tworzenie i przetwarzanie obrazów w ramach grafiki rastrowej.

Realizacja poszczególnych elementów związana jest z opracowaniem określonych modeli i ich implementacja w ramach bibliotek klas.

Opracowano dwie biblioteki graficzne, dystrybuowane w ramach JDK:

- Abstract Window Toolkit (AWT) – pierwsza i zasadnicza biblioteka graficzna w Javie (standardowa część Javy, pakiet java.awt),
- SWING – druga i bardziej rozbudowana (graficznie urozmaicone komponenty – standardowe rozszerzenie Javy, pakiet javax.swing).

1. Wprowadzenie do grafiki w Javie

Obie biblioteki, ***AWT*** i ***SWING***, koordynowane są przez Suna. Istnieją innego biblioteki graficzne proponowane przez innych dostawców (np. Standard Window Toolkit, ***SWT*** – IBM).

Biblioteka ***SWING*** często wprowadza nowe elementy poprzez rozbudowanie (np. dziedziczenie) komponentów graficznych, które są w ***AWT***. Przykładowo przycisk w ***AWT*** realizowany jest poprzez klasę Button. W ***SWING*** również jest przycisk o nazwie JButton (przedrostek J dodawany jest do nazw wielu klas z ***AWT*** budując rozszerzenia w ***SWING***).

Różnicę pomiędzy przyciskami w ***AWT*** i ***SWING*** można przykładowo zilustrować jedną cechą: w ***AWT*** na przycisku można wyświetlić napis; w ***SWING*** napis i/lub ikonę.

1. Wprowadzenie do grafiki w Javie

Należy pamiętać również, że oprócz bibliotek graficznych ogólnego zastosowania (dla komputerów klasy DESKTOP) istnieją różne biblioteki powiązane z innymi dystrybucjami Javy lub innymi zastosowaniami Javy.

Przykładowo dla telefonów komórkowych specjalną wersję elementów graficznych zdefiniowano w ***J2ME***.

Podobnie w przypadku platformy ***Android***.

Omawiając grafikę należy wskazać istniejącą bibliotekę do budowania i przetwarzania scen trójwymiarowych: ***Java3D***, czy bardzo interesującą technologię bazującą na Javie: ***JavaFX*** (interaktywna grafika dla stron WWW i aplikacji mobilnych).

Plan wykładu:

1. Wprowadzenie do grafiki w Javie
2. Budowa GUI: komponenty, kontenery i układanie komponentów
3. Budowa GUI: obsługa zdarzeń
4. Grafika wektorowa – porysujmy sobie
5. Reprezentacja koloru
6. Grafika rastrowa - obrazy
7. Obsługa czcionek

Budowa GUI: komponenty, kontenery i układanie komponentów

Komponenty to podstawowe elementy graficzne aktywne lub bierne służące do tworzenia interfejsu graficznego.
Komponenty są reprezentowane przez klasy, przeważnie dziedziczące po klasie Component (np. Button, Canvas, Checkbox, Choice, Container, Label, List).

Klasa Component dostarcza kilkadziesiąt ogólnie wykorzystywanych metod np.:
getFont()/setFont(Font f) - zwraca informacje o czcionce/ustawia czcionkę dla komponentu,
getGraphics() - zwraca kontekst graficzny komponentu potrzebny przy wywoływaniu metod graficznych
paint(Graphics g) - rysuje komponent,
setSize(int, int) – ustawia rozmiar komponentu, itp.

Budowa GUI: komponenty, kontenery i układanie komponentów

Jednym z komponentów jest kontener reprezentowany przez klasę Container.

Kontener jest komponentem, w którym można umieszczać inne komponenty. Przykładowym kontenerem wykorzystywanym dotąd w tym materiale w przykładach jest Applet (dziedziczy po klasie Panel, a ta po klasie Container). Podstawowe klasy (kontenery) dziedziczące z klasy Container to:

JComponent, Panel, ScrollPane, Window

Najczęściej wykorzystywane kontenery w ***AWT*** to Panel oraz Window. Ten ostatni nie jest wprost wykorzystywany lecz poprzez swoje podklasy:

Dialog *(okno dialogowe),* Frame *(okno z ramką),* JWindow *(okno bez ramki).*

Budowa GUI: komponenty, kontenery i układanie komponentów

Klasa Frame jest kontenerem bezpośrednio wykorzystywanym do tworzenia okien graficznych popularnych w GUI.
Dialog jest kontenerem dedykowanym komunikacji z użytkownikiem i stanowi okno o zmienionej funkcjonalności względem Frame (brak możliwości dodania paska Menu).

Podstawowa praca w konstrukcji interfejsu graficznego w Javie polega na:
1. zadeklarowaniu i zdefiniowaniu kontenera,
2. zadeklarowaniu komponentu,
3. zdefiniowaniu (zainicjowanie) komponentu
4. dodanie komponentu do kontenera.

```
import java.awt.*;
public class Witajcie{
        public static void main(String args[]){
                // 1. zadeklarowanie i zdefiniowanie kontenera,
                //Utwórz obiekt istniejącej klasy Frame (okno z ramką)
                Frame f = new Frame("Witajcie");

                //2 i 3 zadeklarowanie i zdefiniowanie komponentu
                //Utwórz obiekt istniejącej klasy Button (przycisk)
                Button b = new Button("Przycisnij mnie ...");

                //4. Dodanie komponentu do kontenera
                //Dodaj przycisk do okna
                f.add(b);
                //Ustaw rozmiar okna
                f.setSize(500,500);
                //Wyświetl okno
                f.setVisible(true);
        }//main
}//class
```

Budowa GUI: komponenty

a)dziedziczące pośrednio i bezpośrednio po klasie Component:

Label – pole etykiety

Button – przycisk

Canvas – pole graficzne

Checkbox – element wyboru (logiczny)

Choice -element wyboru (z listy)

List – lista elementów

Scrollbar – suwak

TextComponent: TextField: pole tekstowe TextArea: obszar tekstowy

b) nie dziedziczące po klasie Component:

MenuBar – pasek menu,

MenuItem - element menu:

Menu – menu (zbiór elementów)

PopupMenu – menu podręczne.

Budowa GUI: komponenty

Komponent, którego klasa nie dziedziczy po klasie Component to MenuComponent.
Klasa ta jest nadklasą komponentów: MenuBar, MenuItem. Ta ostatnia dostarcza również poprzez klasy dziedziczące z niej następujące komponenty: Menu oraz PopupMenu. Łącznie cztery klasy tj. MenuBar, MenuItem, Menu oraz PopupMenu są wykorzystywane do tworzenia menu programu graficznego.

Wszystkie komponenty biblioteki AWT są dobrze opisane w dokumentacji (Java API) Javy, i dlatego nie będą szczegółowo omawiane w tym materiale.

```
import java.awt.*;
public class PulpitJedi extends Frame{
  public void init() {
            Button przyciskTest = new Button("ognia..."); add(przyciskTest);
            Label opis = new Label("Strzelac"); add(opis);
            Checkbox rak = new Checkbox("Rakieta");
            Checkbox bomb = new Checkbox("Bomba"); add(rak); add(bomb);
            Choice kolor = new Choice();
            kolor.add("zielona"); kolor.add("czerwona"); kolor.add("niebieska"); add(kolor);
            List lista = new List(2, false);
            lista.add("mała"); lista.add("malutka"); lista.add("wielka");
            lista.add("duża"); lista.add("ogromna"); lista.add("gigant"); add(lista);
            TextField param = new TextField("Podaj parametry"); add(param);
  }//koniec init()
            public static void main (String [] a){
                        PulpitJedi f=new PulpitJedi();
                        f.setLayout(new FlowLayout());//o tym później
                        f.init(); f.setSize(800,150); f.setVisible(true);
            }//koniec main()
}//koniec class PulpitJedi
```

Budowa GUI: komponenty

Pakiet javax.swing dostarcza szeregu zmodyfikowanych i nowych kontenerów i komponentów. Podstawowe komponenty tej biblioteki są podobne jak dla AWT z tym, że oznaczane z dodatkową literą J. Wprowadzono również nowe i przeredagowane elementy jak np.:

- JColorChooser – pole wyboru koloru
- JFileChooser – pole wyboru pliku
- JPasswordField – pole hasła,
- JProgressBar – pasek stanu
- JRadioButton – element wyboru
- JScrollBar - suwak
- JTable - tabela
- JTabbedPane - pole zakładek
- JToggleButton –przycisk dwu stanowy
- JToolBar – pasek narzędzi
- JTree – lista w postaci drzewa

```
import java.awt.*;
import javax.swing.*;
public class SwingJedi extends JFrame{
  public void init(){
            JToolBar jtb = new JToolBar("STEROWANIE");
            jtb.add(new JButton("|<")); jtb.add(new JButton(">")); jtb.add(new JButton(">|"));
            add(jtb);
            JPasswordField jpf = new JPasswordField(30); add(jpf);
            JButton jb = new JButton(new ImageIcon("ikona.gif")); add(jb);
            JColorChooser jc=new JColorChooser();              add(jc);
            JTextArea jta = new JTextArea(10, 60);
            JScrollPane jsp = new JScrollPane(jta);   add(jsp);
  }//koniec init()
  public static void main(String []a){
            SwingJedi sj=new SwingJedi();  sj.setLayout(new FlowLayout());
            sj.setUndecorated(true);  sj.init();
            sj.setSize(800,600);  sj.setLocation(100,50);
            sj.setVisible(true);
  }//koniec main()
}//koniec class SwingJedi
```

Budowa GUI: kontenery

Podstawowe klasy kontenerów w bibliotece AWT:

- Frame, Panel, Applet, Dialog

Odpowiednie klasy kontenerów w bibliotece SWING:

- JFrame, JWindow, JDialog, JPanel, JApplet, JFileDialog.

Znając charakterystykę kontenerów warto zapoznać się z możliwością ich wykorzystania. Zasada jest bardzo prosta - kontenery tworzy się dla:

- aplikacji – główny kontener to Frame, JFrame lub JWindow,
- apletu – główny kontener to Applet lub JApplet.

Nie oznacza to jednak, że nie można używać okien w apletach i apletów w oknach!

Klasa Panel/JPanel umożliwia utworzenie konteneru w kontenerze (w oknie kilka paneli, w każdym z nich …)

```
import java.awt.*;
import javax.swing.*;
public class KonteneryJedi extends JFrame{
        public KonteneryJedi(String tytul){
                super(tytul);
        }//koniec KonteneryJedi()
        public static void main(String []a){
                KonteneryJedi kj=new KonteneryJedi("Moje okno !!!");
                kj.setSize(800,600);
                kj.add(new JLabel("JESTEM TYLKO ETYKIETĄ..."));
                kj.setVisible(true);
                JDialog d;
                for(int i=0; i<40; i++){
                        d = new JDialog(kj,"OKNO NR "+i, false);
                        d.setSize(200,200);
                        d.add(new JButton(new ImageIcon("ikona.gif")));
                        d.setLocation(i*15,i*15);
                        d.setVisible(true);
                }//for
        }//koniec main()
}//koniec class KonteneryJedi
```

Budowa GUI: kontenery

Tworząc okno graficzne należy kierować się następującą procedurą:

deklaracja i zainicjowanie okna, np.: Frame okno = new Frame("Program");

- ustawienie parametrów okna, np. okno.setSize(300,300);
- dodanie komponentów do okna, np.: okno.add(new Button("Ognia"));
- ustawienie okna jako aktywnego, np.: okno.setVisible(true);

Brak ostatniego kroku spowoduje, że zdefiniowane okno będzie niewidoczne dla użytkownika.

Praktycznie interfejs składany jest z klocków (kontenery i komponenty). Korzystając ponadto z szeregu klas narzędziowych (np. Toolkit czy GraphicsEnvironment) projektowanie interfejsu może być bardzo proste. Przykładowo wywołanie metody *Toolkit.getDefaultToolkit().getScreenSize()* zwróci obiekt klasy Dimension przechowujący rozmiar ekranu w pikselach (szerokość i wysokość).

```
import javax.swing.*;
import java.awt.*;
public class OknaJedi extends JFrame{

        public static void main(String []a){
                Dimension d=Toolkit.getDefaultToolkit().getScreenSize();
                OknaJedi oj =new OknaJedi();

                oj.setSize(d.width, d.height);
                oj.add(new JButton("Wymiary okna: "+d.width+", "+d.height));
                oj.setVisible(true);

        }//koniec main()
}//koniec class OknaJedi
```

Budowa GUI: kontenery

Z kontenerem typu Frame/JFrame związany jest zbiór komponentów menu. W celu stworzenia menu okna graficznego należy wykonać następujące kroki:

1. zainicjować obiekt klasy MenuBar reprezentujący pasek menu;
2. zainicjować obiekt klasy Menu reprezentujący jedną kolumnę wyborów,
3. zainicjować obiekt klasy MenuItem reprezentujący element menu w danej kolumnie
4. dodać element menu do Menu;
5. powtórzyć kroki 2,3,4 tyle razy ile ma być pozycji w kolumnie i kolumn
6. dodać pasek menu do okna graficznego.

```
import java.awt.*;
class Okno extends Frame{
        Okno(String nazwa){
                super(nazwa);  setResizable(false);  setSize(600,400);
        }//koniec Okno()  }// koniec class Okno
public class MenuJedi{
        public static void main(String args[]){
                Okno o = new Okno("Panel sterujący działem");
                MenuBar pasek = new MenuBar(); //najpierw pasek
                Menu plik = new Menu("Plik"); //pierwsza lista menu
                plik.add(new MenuItem("Ognia")); //pozycja na liście menu
                plik.add(new MenuItem("-")); //separator
                plik.add(new MenuItem("Stop"));
                pasek.add(plik);
                Menu edycja = new Menu("Edycja"); //druga lista menu
                edycja.add(new MenuItem("Pokaż dane działa"));
                pasek.add(edycja);
                o.setMenuBar(pasek);//dodaj pasek menu do okna NIE przez add()!
                o.add(new Button());//dodaj przycisk do okna
                o.setVisible(true);
        }//koniec main()   }// koniec class MenuJedi
```

Budowa GUI: układanie komponentów

Bardzo częstym dylematem programisty jest problem rozkładu komponentów w ramach tworzonego interfejsu graficznego. Problem rozdzielczości ekranu, liczenia pikseli to typowe zadania do rozwiązania przed przystąpieniem do projektu interfejsu.
Komponenty w kontenerze można ułożyć:
-bezwzględnie według położenia w pikselach (według górnego lewego narożnika kontenera),
- względnie według odniesienia do innych komponentów lub obszarów kontenera (np. góra, dół, środek).

W języku JAVA problemy te w znaczny sposób zredukowano wprowadzając tzw. rozkłady (layouts). Metody rozkładania komponentów w kontenerze oznaczają nic innego jak sposób układania komponentów na danej formie (np. jeden po drugim lub w komórkach rastra).

Budowa GUI: układanie komponentów

Dla kontenera definiuje się określoną formę (rozkład), a system zarządzający rozkładami (Layout Manager) umieszcza dany komponent zgodnie z przyjętym rozkładem.
Każdy kontener, jako własność, ma ustaloną domyślną metodę automatycznego rozkładania komponentów w kontenerze, co oczywiście można zmienić.

Możemy wyróżnić następujące metody układania komponentów:
- BorderLayout - (domyślny dla kontenerów: Window, Frame, Dialog, JWindow, JFrame, JDialog) komponenty są umieszczane i dopasowywane do pięciu regionów: północ, południe, wschód, zachód oraz centrum. Każdy z pięciu regionów jest identyfikowany przez stałą z zakresu: BorderLayout.NORTH, .SOUTH, .EAST, .WEST, oraz .CENTER.

Budowa GUI: układanie komponentów

Możemy wyróżnić następujące metody układania komponentów:

- CardLayout - każdy komponent w danej formie (kontenerze) jest rozumiany jako karta. W danym czasie widoczna jest tylko jedna karta, a forma jest rozumiana jako stos kart. Metody omawianej klasy umożliwiają zarządzanie przekładaniem tych kart.

-FlowLayout - (domyślny dla kontenerów*:* Panel, Applet, JPanel, JApplet) komponenty są umieszczane w ciągu "przepływu" od lewej do prawej (podobnie do kolejności pojawiania się liter przy pisaniu na klawiaturze).

- BoxLayout – komponenty układane są według danej osi (np. osi X – od lewej do prawej, osi Y – z góry na dół)

- SpringLayout – komponenty układane są według zasad określonych przez definiowane dodatkowo kryteria (np. pionowa/pozioma odległość pomiędzy brzegami komponentów). Odległość pomiędzy skrajnymi brzegami komponentu mogą określać jego rozmiar.

Budowa GUI: układanie komponentów

Możemy wyróżnić następujące metody układania komponentów:

- GridLayout - komponenty są umieszczane w elementach regularnej siatki (gridu, np. 3 wiersze i 3 kolumny). Forma (kontener) jest dzielona na równe pola, w których kolejno umieszczane są komponenty.

-GridBagLayout - komponenty umieszczane są w dynamicznie tworzonej siatce regularnych pól, przy czym komponent może zajmować więcej niż jedno pole (szczegółowe ustawienia dostępne są przez szereg ograniczeń – pól obiektu klasy GridBagConstraints).

-Null layout – brak metody automatycznego rozkładania komponentów w kontenerze – komponenty muszą mieć określone granice podane w pikselach (setBounds(x, y, szerokość, wysokość)).

-inne (są jeszcze inne metody dostępne w SWING oraz innych, opcjonalnych bibliotekach i środowiskach programistycznych.

```
import javax.swing.*;
import java.awt.*;
public class UkladJedi{
  public static void main(String []a){
            JFrame jf = new JFrame("Test ukladania komponentow.");
            JButton [] b=new JButton[5];
            for(int i=0; i<b.length;i++)      b[i]=new JButton("P"+i);
            //Kolejne przekłady demonstrują różne rozkłady
            /*          //1 domyślny rozkład dla JFrame - BorderLayout
            for(int i=0; i<b.length;i++)
               jf.add(b[i]);  //nie podajemy gdzie dodajemy - domyślnie do CENTER    */
            /* //2 domyślny rozkład z podaniem lokalizacji
            jf.add(b[0],BorderLayout.EAST);
            jf.add(b[1],BorderLayout.WEST);
            jf.add(b[2],BorderLayout.NORTH);
            jf.add(b[3],BorderLayout.SOUTH);
            jf.add(b[4],BorderLayout.CENTER);       */
            /* //3 Rozkład flow layout
                        jf.setLayout(new FlowLayout());
                        for(int i=0; i<b.length;i++)      jf.add(b[i]);
            */
```

c.d.n.

```
        /* //4 Rozkład grid layout
        jf.setLayout(new GridLayout(3,2));      //3 wiersze 2 kolumny
        for(int i=0; i<b.length;i++)      jf.add(b[i]);             */
        /* //5 Rozkład card layout
        CardLayout cl=new CardLayout();
        jf.setLayout(cl);
        for(int i=0; i<b.length;i++)      jf.add(b[i],""+i);
        cl.show(jf.getContentPane(),"2");      //co to ContentPane – później     */
        /*  //6 Rozkład BoxLayout
        jf.setLayout(new BoxLayout(jf.getContentPane(), BoxLayout.Y_AXIS));
        for(int i=0; i<b.length;i++){
                b[i].setAlignmentX(Component.CENTER_ALIGNMENT);
                b[i].setMinimumSize(new Dimension(50, 50));
                b[i].setPreferredSize(new Dimension(100, 100));
                b[i].setMaximumSize(new Dimension(150,150));                jf.add(b[i]);
                jf.add(Box.createRigidArea(new Dimension(0,10)));//wstaw przerwę
        }   */
        jf.setSize(600,600);
        jf.setVisible(true);
  }//koniec main()
}//koniec class UkladJedi
```

Budowa GUI: układanie komponentów

Znając różne metody rozkładania komponentów w kontenerze możemy budować złożone interfejsy użytkownika poprzez:

1. Określenie głównego rozkładu dla kontenera,
2. Dodanie kontenerów (to też komponenty, np. JPanel) do poszczególnych miejsce w kontenerze głównym,
3. Dla dodanych kontenerów określić wymagane rozkłady (może być każdy inny),
4. Do każdego kontenera dodać komponenty lub kolejne kontenery, itd.

Przykładowo dla JFrame wybieramy BorderLayout (domyślny). Do BorderLayout.SOUTH dodajemy JPanel. W dodanym panelu ustawiamy FlowLayout (domyślny) i do niego dodajemy zestaw przycisków….

Język JAVA – GUI


```
import java.awt.*;
import javax.swing.*;
public class UkladWieluJedi extends JFrame{
  public void initGUI(){
            JPanel panelPrzyciskow=new JPanel();
            JButton pOtworz=new JButton("OTWÓRZ");JButton pZapisz=new JButton("ZAPISZ");
            JButton pWyslij=new JButton("WYŚLIJ");
            panelPrzyciskow.add(pOtworz); panelPrzyciskow.add(pZapisz);
            panelPrzyciskow.add(pWyslij);
            //Proszę spróbować różne położenia panelu
            //this.add(panelPrzyciskow,BorderLayout.SOUTH);
            this.add(panelPrzyciskow,BorderLayout.NORTH);
            JTextArea jta=new JTextArea();   JScrollPane jsp=new JScrollPane(jta);
            this.add(jsp,BorderLayout.CENTER);
  }//koniec initGUI()
  public void wyswietl(){
            this.setSize(500,500);  this.setResizable(false);  this.setVisible(true);
  }//koniec wyswietl();
  public static void main(String []a){
            UkladWieluJedi uwj=new UkladWieluJedi(); uwj.initGUI();   uwj.wyswietl();
  }//koniec main()   }//koniec class UkladWieluJedi
```

Budowa GUI: układanie komponentów

A co jeśli nie chcemy wykorzystywać automatycznego rozkładania komponentów w kontenerze? Wówczas ustawiamy

```
setLayout(null);
```

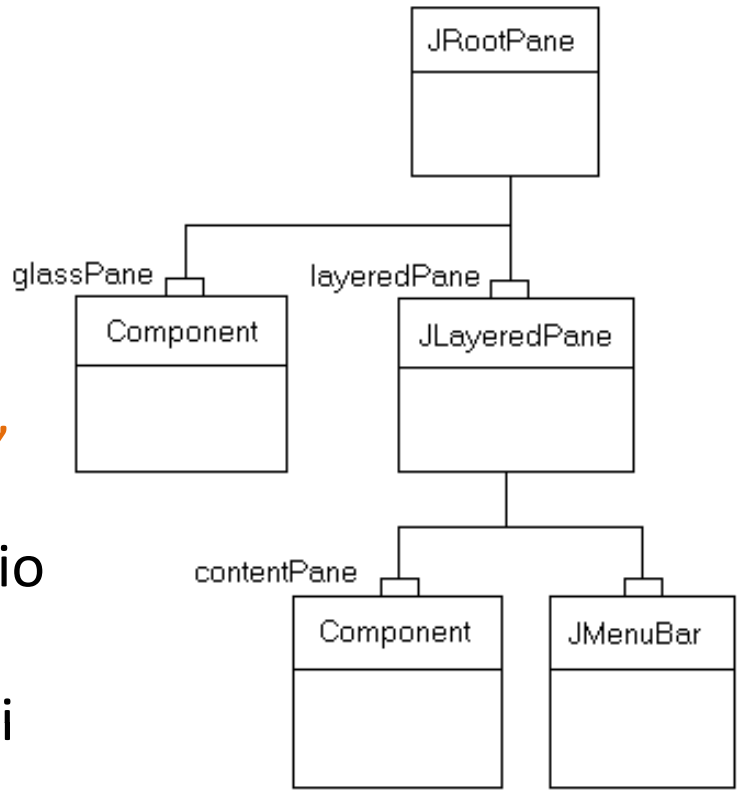
I możemy dodawać komponenty (używając metody add()), a następnie musimy określić położenie w pikselach KAŻDEGO komponentu:

```
JFrame jf=new JFrame("Moja ramka");
JButton jb=new JButton("Przycisk");
jf.add(jb);
jb. setBounds(0, 0, 140, 70); //x,y,szerokość,wysokość
```

i tak dla każdego komponentu.

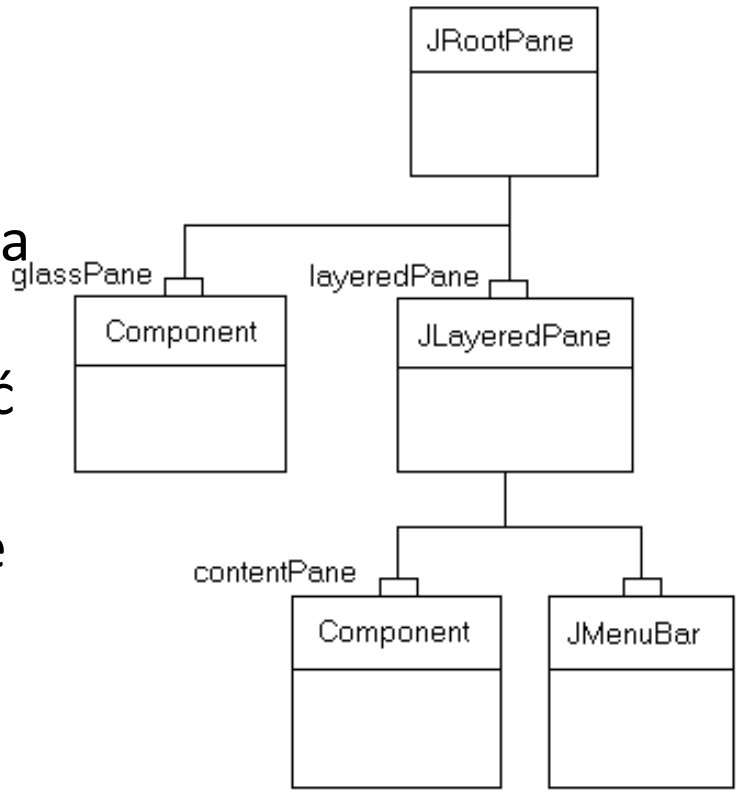
UWAGA! Tworzenie interfejsu graficznego znacznie ułatwiają zintegrowane środowiska programistyczne (np. NetBeans), które umożliwiają graficzną kompozycję komponentów (składamy z klocków). Dla kompozycji automatycznie generowany jest kod.

Pakiet **Swing** wprowadza nową budowę konteneru. Kontener (forma) zawiera podstawowe komponenty zwane PANE (płyty).
Podstawowym komponentem jest tu płyta główna (korzeń) - JRootPane. Płyta ta jest fundamentalną częścią kontenerów w **Swing** takich jak JFrame, JDialog, JWindow, JApplet. Oznacza to, że umieszczenie komponentów nie odbywa się bezpośrednio przez odwołanie do tych kontenerów lecz poprzez JRootPane. Dla potrzeb konstrukcji GUI z wykorzystaniem elementów pakietu Swing istotna jest znajomość struktury JRootPane, pokazanej poniżej.



JRootPane składa się z **glassPane** (szyba, płyta szklana) i **layeredPane** (warstwę płyt). Ta ostatni złożona jest z : opcjonalnego obektu ***menuBar*** (pasek menu) oraz ***contentPane*** (płyta robocza, zawartość). Płyta ***glassPane*** jest umieszczona zawsze na wierzchu wszystkich elementów (stąd szyba). Ponieważ ***glassPane*** może stanowić dowolny komponent możliwe jest więc rysowanie w tym komponencie. Domyślnie ***glassPane*** jest niewidoczna. Płyta ***contentPane*** stanowi główną roboczą (domyślną) część kontenera gdzie rozmieszczamy komponenty. Domyślne dodanie komponentu add() działa jak:

rootPane.getContentPane().add(child);



Zastosowanie GlassPane

Typowe zastosowanie GlassPane to utworzenie prezentacji wyświetlanej w pierwszym planie w czasie wykonywania długotrwałej operacji, np. pasek postępu (progress bar) pokazujący postęp operacji.

//Utwórz komponent, czyli class PasekPostepu extends JComponent
//Utwórz obiekt klasy Cos, np. PasekPostepu pasekWglassPane;
setGlassPane(glassWglassPane);

DEMO: GlassPaneTest (projekt NetBeans)

Plan wykładu:

1. Wprowadzenie do grafiki w Javie
2. Budowa GUI: komponenty, kontenery i układanie komponentów
3. Budowa GUI: obsługa zdarzeń
4. Grafika wektorowa – porysujmy sobie
5. Reprezentacja koloru
6. Grafika rastrowa - obrazy
7. Obsługa czcionek