Programowanie obiektowe

Wykład 10: Grafika w Javie

dr inż. Marcin Luckner
mluckner@mini.pw.edu.pl

Wydział Matematyki i Nauk Informacyjnych

Wersja 1.3
5 marca 2021

Fundusze Europejskie Wiedza Edukacja Rozwój
Rzeczpospolita Polska
Politechnika Warszawska
Unia Europejska Europejski Fundusz Społeczny

Projekt „NERW 2 PW. Nauka – Edukacja – Rozwój – Współpraca” współfinansowany jest ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Zadanie 10 pn. „Modyfikacja programów studiów na kierunkach prowadzonych przez Wydział Matematyki i Nauk Informacyjnych”, realizowane w ramach projektu „NERW 2 PW. Nauka – Edukacja – Rozwój – Współpraca”, współfinansowanego jest ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Rzeczpospolita
Polska

Politechnika
Warszawska

Graficzny interfejs użytkownika

- Jeżeli nasze oprogramowanie nie jest przeznaczone do działania po stronie serwera to prędzej czy później musimy wyposażyć je w graficzny interfejs użytkownika.
- GUI zapewnia możliwość obsługi oprogramowania poprzez zestaw komponentów.
- Java posiada standardowy zestaw komponentów i dość oryginalny sposób zarządzania ich układem

Abstract Window Toolkit

- Na początku swego istnienia Java udostępniała bibliotekę o nazwie Abstract Window Toolkit (AWT).
- Biblioteka AWT przekazuje zadania dotyczące tworzenia interfejsu graficznego do natywnych metod systemu operacyjnego.
- Biblioteka zapewniała przenośność funkcjonalności interfejsu graficznego, ale ze względu na ograniczenia systemów i różnice między nimi nie pozwala na budowanie skomplikowanych i spójnych interfejsów.
- Wywoływanie natywnych operacji wymagało też testowania oprogramowania na każdej z platform.
- Ze względu na te ograniczenia postanowiono zmienić podejście do budowy interfejsu i stworzono bibliotekę Swing

Swing

- Biblioteka Swing nie zastępuje AWT tylko rozszerza jej działanie.
- Biblioteka AWT wywołuje systemową funkcję tworzenia pustego okna, które jest następnie pokrywane komponentami klasy Swing.
- Powoduje to pewne opóźnienie w stosunku do rysowania komponentów AWT, ale jest ono pomijalne.

Zalety biblioteki Swing

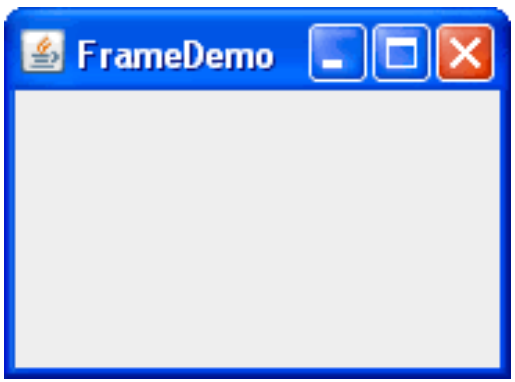
- Biblioteka Swing oferuje:
  - Bogaty i wygodny w użyciu zestaw kontrolek.
  - Niewielką zależność od systemu operacyjnego, co ułatwia testowanie i zwiększa niezawodność interfejsu.
  - Spójny sposób działania i wygląd elementów na różnych platformach systemowych.

Nazewnictwo

- Większość komponentów biblioteki Swing zaczyna się od litery J
  - `JFrame`,
  - `JButton`.
- Łatwo je pomylić z klasami biblioteki AWT.
  - `Frame`
  - `Button`
- Można łączyć ze sobą komponenty AWT i Swing, ale nie jest to potrzebne i zalecane.

Tworzenie ramki

- Głównym komponentem GUI jest ramka `JFrame`



Rysunek 1: Ramka jako podstawowy komponent GUI

Tworzenie ramki

```
1  JFrame frame = new JFrame("FrameDemo");
2  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
3  frame.getContentPane().add(new JLabel());
4  frame.pack();
5  frame.setVisible(true);
```

Poprawne tworzenie ramki

- System obsługi GUI jest asynchroniczny. Oznacza to, że nasz interfejs może otrzymać w dowolnym momencie zdarzenie do obsłużenia.
- W związku z tym krytyczne jest poprawne tworzenie elementów interfejsu, aby nie otrzymywały zdarzeń, zanim będą na to w pełni gotowe.
- Wykorzystujemy do tego metodę `invokeLater`.

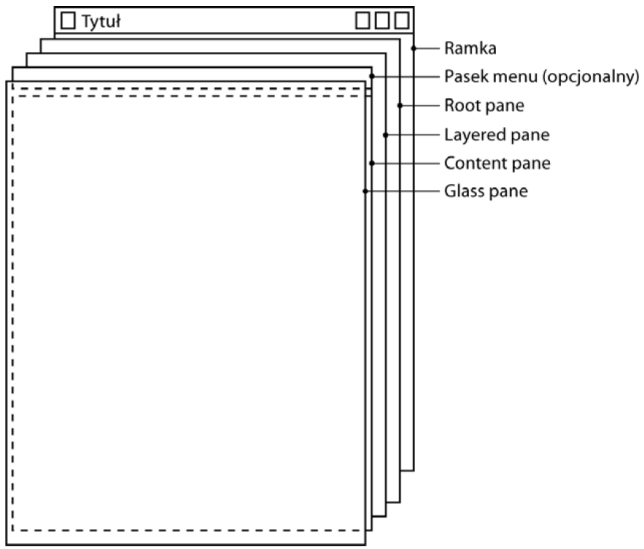
Wywołanie kodu tworzącego GUI

```
1  SwingUtilities.invokeLater(new Runnable() {
2    public void run() {
3      JFrame frame = new JFrame("FrameDemo");
4      frame.getContentPane().add(new JLabel());
5      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
6      frame.pack();
7      frame.setVisible(true);
8      }
9  });
```

Obsługa ramki

- Po utworzeniu ramki deklarujemy jaka akcja ma zostać wykonana w momencie jej zamknięcia.
  - W przypadku aplikacji jednookienkowej powinniśmy wywołać metodę `setDefaultCloseOperation`, aby ustalić, że zamknięcie ramki kończy program.
  - W przypadku aplikacji wielookienkowej zamknięcie ramki nie powinno kończyć programu, tylko ją ukryć. Jest to domyślne działanie przy zamykaniu ramki.

Struktura ramki



Rysunek 2: Struktura komponentu `JFrame` [Horstmann, 2016]

- Komponent `JFrame` składa się z kilku elementów.
- Listwa sterująca zawiera przyciski sterujące oknem.
- Pasek menu zawiera elementy menu.
- Element *Content Pane* tworzy kontener, w którym możemy umieszczać kontrolki.

```
1  frame.getContentPane().add(emptyLabel);
2  frame.add(emptyLabel)
       //alternatywnie
```

- Pozostałe elementy pozwalają nam na działania ponad kontrolkami np. na rysowanie kursora.

Rysowanie komponentów

- Komponenty są rysowane dzięki metodzie `paintComponent`.
  - Kontrolki pakietu Swing mają zaimplementowane odpowiednie metody.
  - W przypadku własnych kontrolek należy nadpisać metodę klasy `JComponent`.
- Metoda `paintComponent` przekazuje dostęp do obiektu `Graphics` pozwalającego na rysowanie figur, tekstu, obrazów.
- Metody `paintComponent` nie należy wywoływać! stanie się to automatycznie, gdy będzie wymagane przerysowanie komponentu.
- Jeżeli chcemy wymusić przerysowanie komponentu to wywołujemy metodę `repaint`.

Tworzenie własnych komponentów

- Komponent rozszerza klasę `JComponent` oraz nadpisuje metody `paintComponent` i `getPreferredSize`.

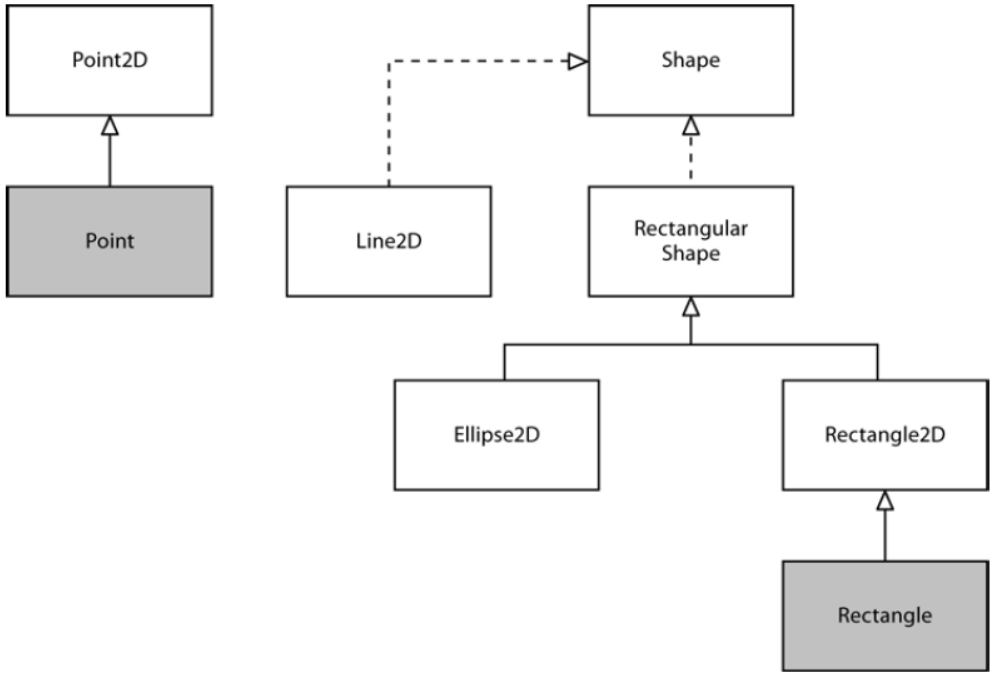
```
1  public class MyComponent extends JComponent {
2      static int WIDTH = 500;
3      static int HEIGHT = 50;
4      static int OFFSET = 50;
5      static int FONTSIZE = 24;
6      String text = "This is a demo of JFrame component."
7      public void paintComponent(Graphics g) {
8          g.setColor(Color.CYAN);
9          g.fillOval(0,0,WIDTH,HEIGHT);
10         g.setColor(Color.BLUE);
11         g.drawOval(0,0,WIDTH,HEIGHT);
12         g.setFont(new Font("Arial",Font.ITALIC,FONTSIZE));
13         g.drawString(text,OFFSET,OFFSET);
14     }
15     public Dimension getPreferredSize(){
16         return new Dimension(WIDTH,HEIGHT);
17     }
18 }
```

Rysunek 3: Stworzony komponent

Większe możliwości rysowania

- Możemy rozszerzyć możliwości rysowania korzystając z klasy `Graphics2D`.
- Otrzymujemy do niej dostęp rzutując obiekt `Graphics` otrzymywany w metodzie `paintComponent`.
- Klasa `Graphics2D` pozwala rysować różnorodne kształty, ustalać rodzaj pędzla i wykonywać przekształcenia afiniczne.
- W przeciwieństwie do klasy `Graphics` klasa `Graphics2D` posługuje się współrzędnymi rzeczywistymi typu `float`. Jest to wystarczająca precyzja dla określania współrzędnych ekranu.
- Ponieważ w Javie najczęściej posługujemy się dokładnością `double` to wprowadzono także możliwość stosowania tego typu.

Kształty



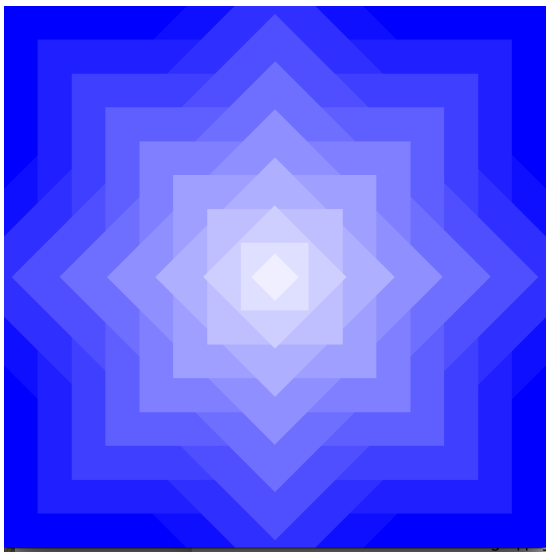
Rysunek 4: Hierarchia klasy `Shape` [Horstmann, 2016]

- Figury geometryczne są pochodnymi klasy `Shape`.
- Można nadpisać klasę `Shape` by stworzyć własne figury.
- Można także dokonywać przekształceń afinicznych układu odniesienia używanego podczas rysowania.

Korzystanie z klasy Graphics2D

Komponent MyComponent2D

```
1  public class MyComponent2D extends
       MyComponent {
2   static int STEP = 16;
3   public void paintComponent(Graphics g) {
4    Graphics2D g2 = (Graphics2D) g;
5    for(int i=0;i<STEP;i++) {
6     double m = 0.5*i*WIDTH/STEP;
7     double w = WIDTH-2*m;
8     Rectangle2D rect = new
          Rectangle2D.Double(m,m,w,w);
9     int value =
          (int)(i/((double)STEP)*255);
10    g2.setPaint(new
          Color(value,value,255));
11    g2.fill(rect);
12    g2.translate(WIDTH/2.0, WIDTH/2.0);
13    g2.rotate(Math.PI/4.0); //Kąt w
          radianach
14    g2.translate(-WIDTH/2.0, -WIDTH/2.0);
15   }
16  }
17  public Dimension getPreferredSize(){
18   return new Dimension(WIDTH,WIDTH);
19  }
20 }
```



Rysunek 5: Stworzony komponent

Wczytywanie obrazów

- Możemy wczytywać obrazy z plików graficznych typu GIF, PNG, JPEG i BMP.
- Klasa `ImageIcon` oferuję metodę `getImage` odczytującą pliki określone przez ścieżkę lub adres URL.
- Ponieważ pozyskanie grafiki z pliku nie jest operacją natychmiastową umożliwiono nadzór nad podstępem wczytywania.
- Rolę nadzorcy może pełnić obiekt implementujący interfejs `ImageObserver`, w tym dowolny komponent.
- Mechanizm jest używany przy śledzeniu postępu wczytywania i informowaniu o konieczności przerysowania komponentu.

Wczytywanie fontów i obrazów

Komponent MINIComponent

```
1  public class MINIComponent extends MyComponent2D {
2      Image logoMiNI;
3
4      public MINIComponent() {
5          super();
6          logoMiNI = new ImageIcon("LogoMiNI.png").getImage();
7          FONTSIZE = 19;
8      }
9      public void paintComponent(Graphics g) {
10         Graphics2D g2 = (Graphics2D) g;
11         g2.drawImage(logoMiNI,0,0,HEIGHT,HEIGHT,this);
12         Font f = new Font("Radikal WUT", Font.BOLD, FONTSIZE);
13         g2.setFont(f);
14         g2.drawString("Faculty of Mathematics and Information
               Science",(int)(1.1*HEIGHT),(int)(0.6*HEIGHT));
15     }
16 }
```

Rysunek 6: Stworzony komponent

Bibliografia

[Horstmann, 2016] Horstmann, C. S. (2016).
Java. Podstawy.
Helion.