Java프로그래밍

13강. AWT 컨트롤 클래스 (교재 12장)

컴퓨터과학과 김희천 교수

한국방송통신대학교

오늘의 학습목차

1. 윈도 프로그램 만들기
2. GUI 컴포넌트
3. 메뉴
4. 배치 관리자의 사용

1. 윈도 프로그램 만들기

1) 그래픽 사용자 인터페이스(GUI)

- 그래픽 요소를 이용하여 사용자가 프로그램과 대화하는 방식의 인터페이스
  - ✓ 텍스트 기반 or 명령 행 인터페이스(CLI)와 비교됨

GUI 프로그래밍을 위해 필요한 것

- GUI 컴포넌트
  - ✓ 윈도우, 메뉴, 버튼, 레이블, 콤보박스, 체크박스, 텍스트필드, 스크롤바, 대화상자 등
- 컨트롤, 이벤트 발생과 처리
  - ✓ 사용자 상호작용
- 컨테이너와 배치 관리자

2) AWT

- JFC(Java Foundation Class)
  - ✓ GUI를 만들거나 그래픽 처리를 위한 클래스 라이브러리
  - ✓ AWT, Swing, Java2D, 룩앤필 지원 API 등을 제공
    - JavaFX, 3D, Sound, Image 관련 API도 있음

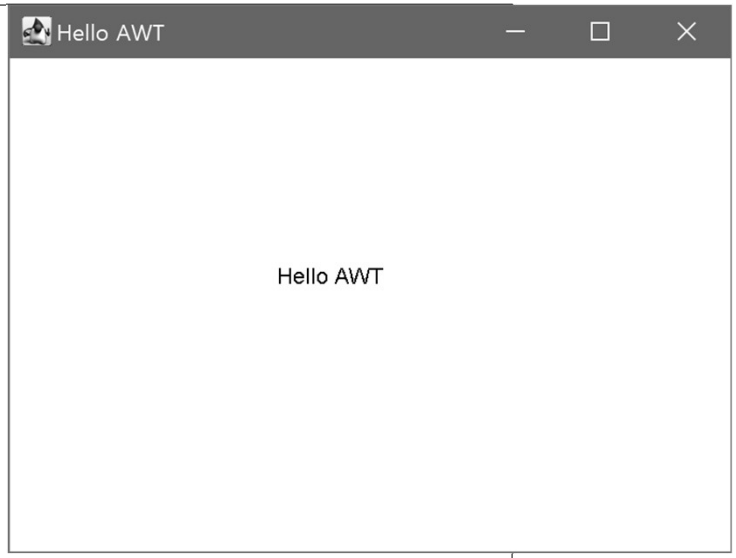
AWT(Abstract Window Toolkit)

- AWT는 자바에서 처음 제공한 GUI용 API
- 주 패키지는 java.awt
- 운영체제의 윈도우 시스템을 사용함
  - ✓ 중량 컴포넌트로 외양이 운영체제마다 다름

1. 윈도 프로그램 만들기

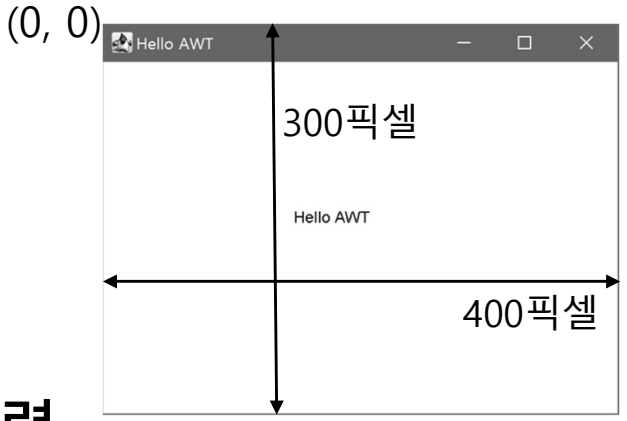
3) 윈도 프로그램 예

```
import java.awt.*;
class MyFrame extends Frame {
      public MyFrame(String title) {
         super(title);
         this.setSize(400, 300);
         this.setVisible(true);
      }
      public void paint(Graphics g) {
         g.drawString("Hello AWT", 150, 150);
      }
}

public class HelloAWT2 {
    public static void main(String args[ ]) {
         MyFrame myFrame = new MyFrame("Hello AWT");
    }
}
```



4) 윈도 프로그램 만들기

- Frame 클래스를 상속받아 클래스를 정의
  - ✓ 프레임은 제목표시줄, 경계, 최소/최대/종료 버튼과 같은 윈도우 장식을 가짐
- 생성자에서 윈도우의 주요 속성을 지정
  - ✓ 제목을 인자로 받아 지정
    - super(title);
  - ✓ 가로와 세로 크기를 지정
    - setSize(400, 300);
  - ✓ 화면에 표시
    - setVisible(true);



- paint( ) 메소드에서 문자열을 출력
  - ✓ paint( )는 '다시 그리기' 이벤트가 발생할 때 자동 호출되는 메소드

2. GUI 컴포넌트

2. GUI 컴포넌트

1) AWT 패키지의 GUI 컴포넌트

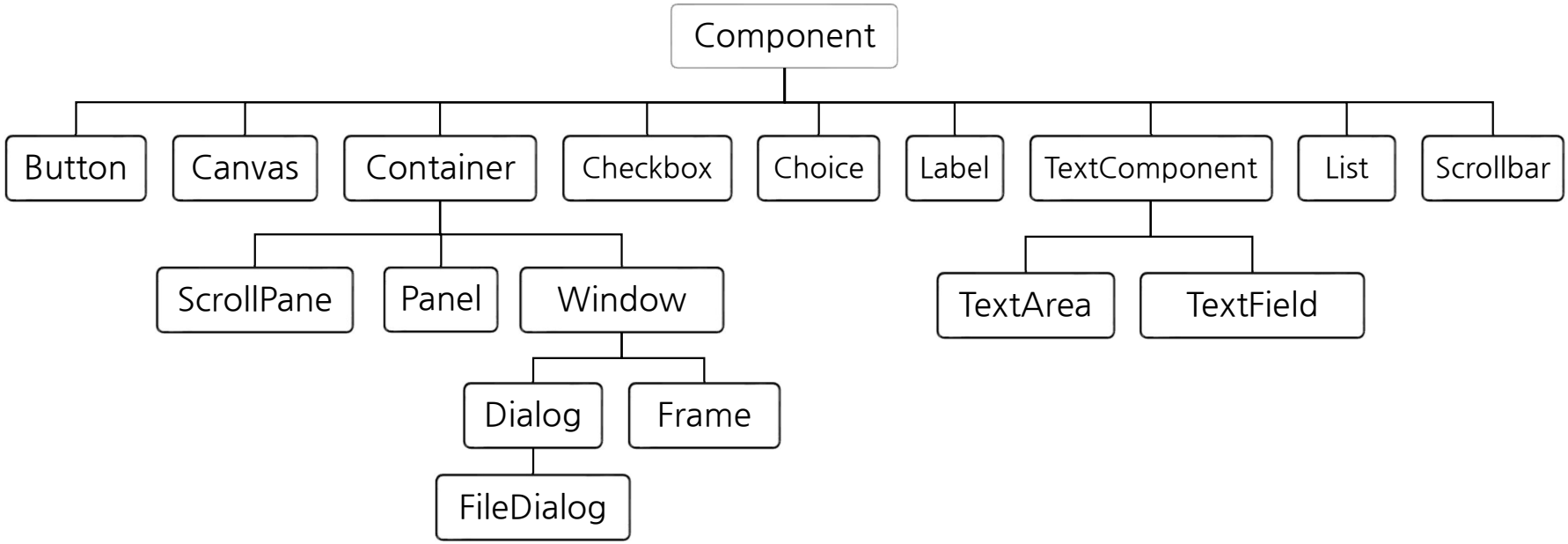
분류	의미와 클래스
컨트롤	사용자와 실제 의사소통하는 GUI 컴포넌트 Button, Label, Canvas, Choice, Checkbox, Menu 등
컨테이너	하나 이상의 컨트롤을 포함하고 레이아웃을 관리 Panel, Frame, Window, Dialog 등

기타 클래스

- ◆ 컴포넌트 배치
  - ✓ BorderLayout, FlowLayout, GridLayout, CardLayout 등
- ◆ 그래픽 출력
  - ✓ Color, Font, Point, Graphics, Image 등

2. GUI 컴포넌트

2) 컴포넌트 클래스 계층 구조



3) Component 클래스

- 메뉴를 제외한 AWT 컴포넌트들의 최상위 추상 클래스
- 이름, 기준 좌표, 크기, 배경색/전경색, 폰트, visible 속성, Graphics 객체와 같은 속성을 가짐
- 컴포넌트의 기본 메소드들을 정의
  - ✓ void paint(Graphics)
  - ✓ Container getParent( )
  - ✓ void setVisible(boolean b)
  - ✓ void setSize(int, int)
  - ✓ void setBackground(Color c)
  - ✓ … …

4) 컨테이너(1)

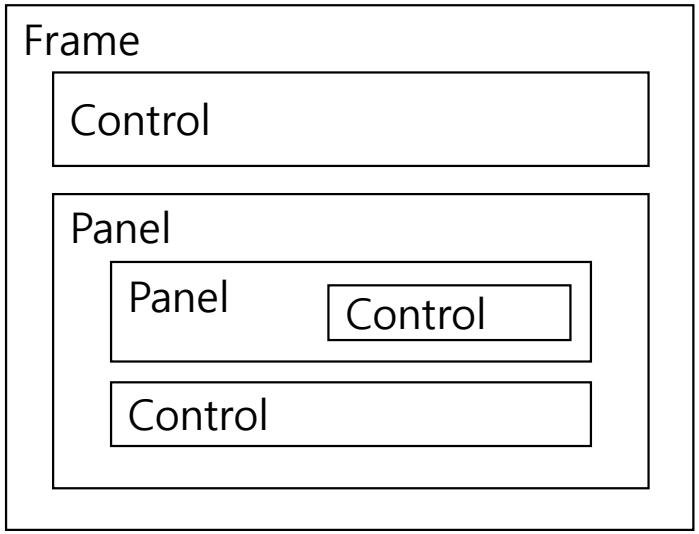
- 다른 컴포넌트를 포함하는 컴포넌트
  - ✓ 컨트롤은 컨테이너에 포함되어야 함
- 최상위 클래스는 추상 클래스인 Container
  - ✓ 하위 컨테이너를 위한 기본 메소드를 제공
- 자식 컴포넌트들의 배치(위치와 크기)를 담당
  - ✓ 기본 배치 관리자를 가짐
    - Frame의 경우 BorderLayout, Panel은 FlowLayout
  - ✓ 배치 관리자를 변경할 수 있음
    - void setLayout(LayoutManager)

4) 컨테이너(2)

- 자식 컴포넌트를 리스트 형태로 관리함
  - ✓ 컨테이너에 추가되는 컴포넌트들은 순서지정이 없다면 맨 뒤에 들어감
- Container 클래스에 자식 컴포넌트를 추가하는 메소드
  - ✓ Component add(Component comp)
    - 마지막에 추가
  - ✓ Component add(Component comp, int index)
    - 지정된 위치(index)에 추가
  - ✓ void add(Component comp, Object constraints)
    - 마지막에 추가, 두번째 인자는 배치 관리자에게 주는 정보

5) 최상위 수준 컨테이너

- 최상위 수준 컨테이너
  - ✓ 컴포넌트 간 포함 관계에서 루트가 되는 컨테이너
    - 모든 GUI 컴포넌트는 1개의 컨테이너에 포함됨
  - ✓ Frame, Window, Dialog 등
- 일반 컨테이너
  - ✓ Panel, ScrollPane



6) Window 클래스

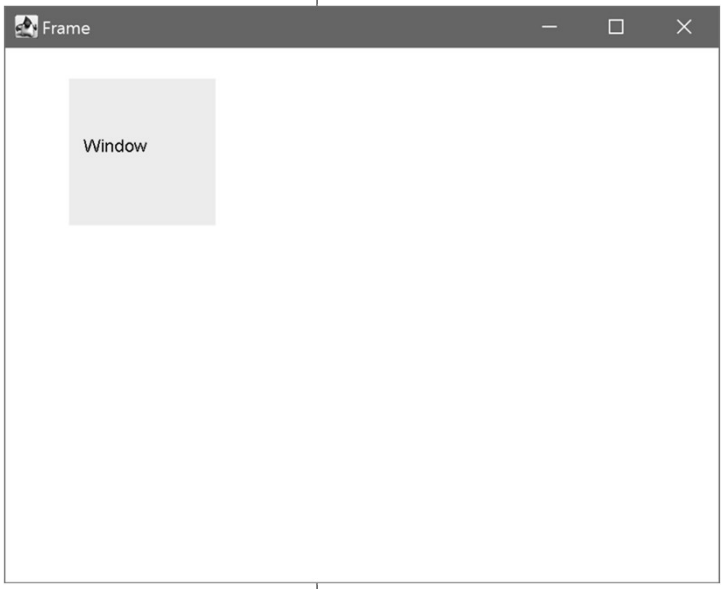
- 최상위 수준의 컨테이너
  - ✓ 다른 컨테이너의 사각영역에 포함될 수 없음
  - ✓ 제목이나 테두리가 없으며 메뉴바를 가지지 않음
- 생성자
  - ✓ Window(Frame owner), Window(Window owner)
- 기본 레이아웃은 BorderLayout

7) Frame 클래스

- 최상위 수준의 컨테이너
  - ✓ 부모 컴포넌트를 가지지 못함
  - ✓ 제목과 테두리가 있으며 메뉴바를 가질 수 있음
  - ✓ 계층 구조에서 Window의 서브 클래스
- 생성자
  - ✓ Frame( ), Frame(String title)
- 기본 레이아웃은 BorderLayout

8) Window와 Frame 클래스 예

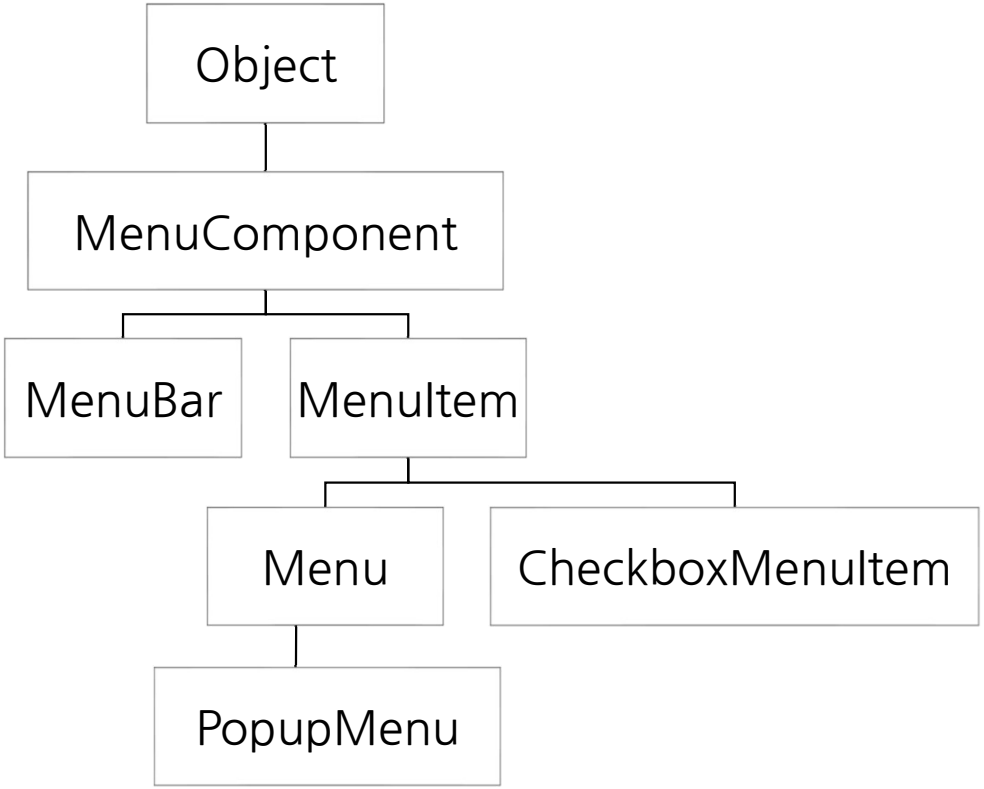
```
public class WindowFrame {
   public static void main(String args[]) {
       Frame f = new Frame("Frame");
       f.setSize(500, 400);
       f.setBackground(Color.white);
       f.setVisible(true);

       Window w = new Window(f) {
          public void paint(Graphics g) {
              g.drawString("Window", 10, 50);
          }
       };
       Rectangle r = new Rectangle(50, 50, 100, 100);
       w.setBounds(r);
       w.setBackground(Color.yellow);
       w.setVisible(true);
       w.toFront();
   }
}
```



3. 메뉴

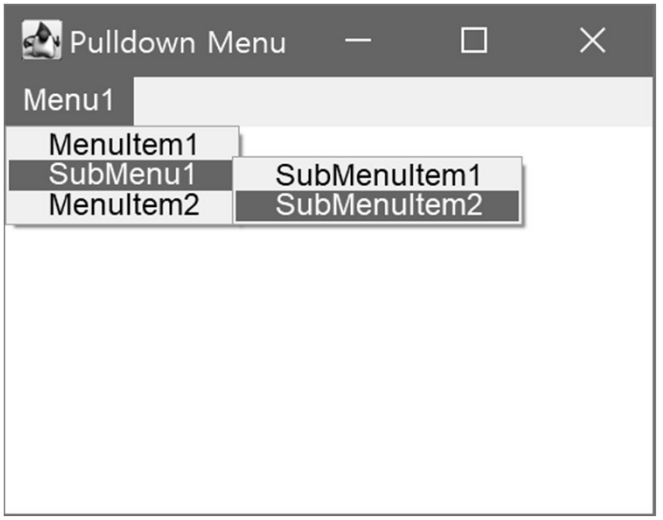
3. 메뉴

1) 메뉴 클래스 계층 구조



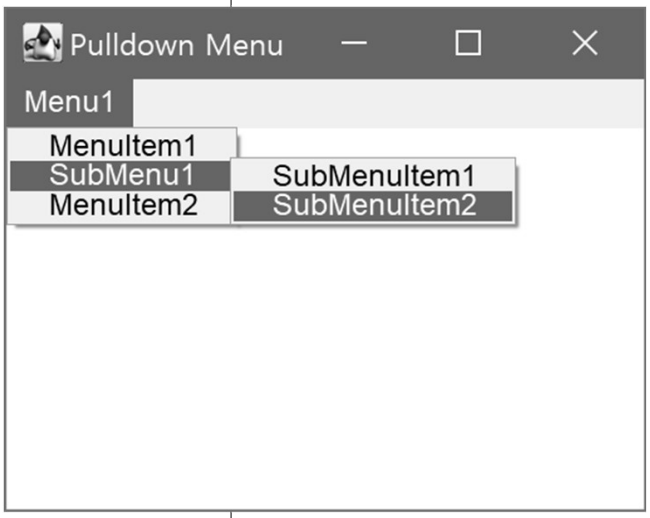
2) 풀다운 메뉴

- 제목 표시줄 밑의 메뉴바를 가짐
- 메뉴 만들기 과정
  - ✓ MenuBar 객체 생성
  - ✓ MenuBar에 추가할 Menu객체를 생성
  - ✓ Menu에 추가할 또다른 서브 Menu객체나 MenuItem 객체를 생성하고 Menu에 붙임 - add( )
  - ✓ 생성한 Menu를 Menubar에 추가 – add( )
  - ✓ 프레임에 MenuBar를 붙임 - setMenuBar( )



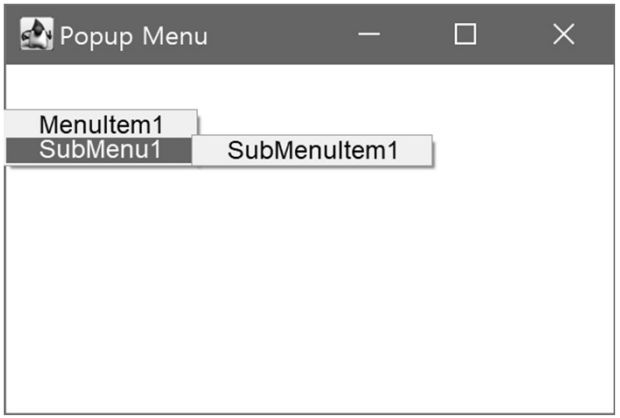
3) 풀다운 메뉴 예

```
import java.awt.*;

public class PulldownMenu {
    public static void main(String args[]) {
       Frame f = new Frame("Pulldown Menu");
       MenuBar mb = new MenuBar();
       Menu m = new Menu("Menu1");
       m.add(new MenuItem("MenuItem1"));
       Menu sm = new Menu("SubMenu1");
       sm.add(new MenuItem("SubMenuItem1"));
       sm.add(new MenuItem("SubMenuItem2"));
       m.add(sm);
       m.add(new MenuItem("MenuItem2"));
       mb.add(m);
       f.setMenuBar(mb);
       f.setSize(200, 200);
       f.setBackground(Color.white);
       f.setVisible(true);
    }
}
```



4) 팝업 메뉴

- 컨테이너 내부에서 어디든 나타남
- 메뉴 만들기
  - ✓ PopupMenu 객체를 생성한다
  - ✓ PopupMenu에 MenuItem이나 서브 Menu객체를 추가 – add( )
  - ✓ PopupMenu를 Frame에 추가 – add( )
  - ✓ PopupMenu를 보이게 함 – show( )



4. 배치 관리자의 사용

4. 배치 관리자의 사용

1) 배치 관리자

- 일관성 있는 배치 관리 방법을 제공하는 클래스
- 모든 컨테이너는 기본 배치 관리자를 속성으로 가짐
  - ✓ 배치 관리자를 변경할 수 있음
- 자식 컴포넌트의 크기와 위치를 자동으로 조정함
  - ✓ 컨테이너의 크기가 바뀌면 자식 컴포넌트들의 크기와 위치가 자동 재조정됨

자식 컴포넌트의 수동 배치

- 배치 관리자를 제거해야 함
  - ✓ myContainer.setLayout(null)
- 이후 크기와 위치를 수동 설정함
  - ✓ setLocation(x, y), setSize(width, height)

2) 배치 관리자의 종류

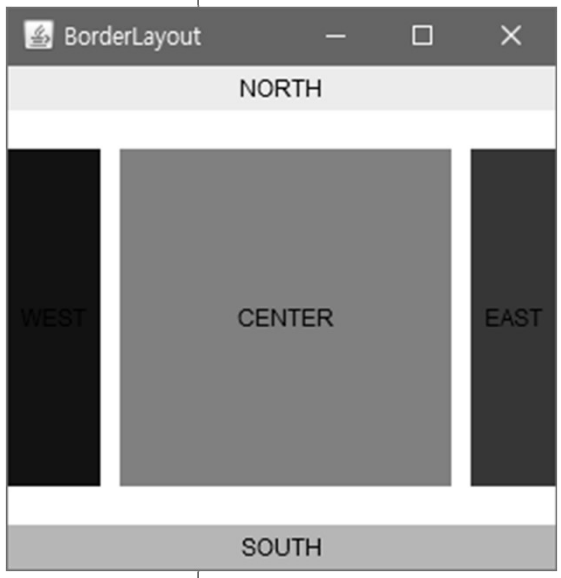
배치 관리자	설명
BorderLayout	중앙, 동, 서, 남, 북에 배치함 중앙이 크게 설정되고 나머지는 최소한으로 설정 Frame, Window, Dialog의 기본 배치 관리자
FlowLayout	한 행에 순서대로 수평으로 배치(모자라면 다음 행에) 선호 크기로 배치함 Panel의 기본 배치 관리자
GridLayout	바둑판 모양으로, 격자 형식으로 배치 컴포넌트 크기가 동일함
CardLayout	한 번에 한 장의 카드(컴포넌트)를 보여줌

3) BorderLayout

- Center, East, West, South, North 영역에 배치
  - ✓ 한 영역에 컴포넌트 1개만 배치함
  - ✓ 컴포넌트 사이의 수평/수직 간격을 지정할 수 있음
  - ✓ 컨테이너 크기를 확장하면 남북의 컴포넌트는 수평으로 동서는 수직으로 확장됨. 중앙이 가장 커짐
- 생성자
  - ✓ BorderLayout(int hgap, int vgap)
- 예
  - ✓ aFrame.setLayout(new BorderLayout(30, 20));
  - ✓ aFrame.add(btn1, “North”);
  - ✓ aFrame.add(btn2, BorderLayout.SOUTH);

4) BorderLayout 사용 예

```
import java.awt.*;
public class BorderLayoutTest extends Frame {
public BorderLayoutTest() {
        super("BorderLayout");
        setSize(300, 300);
        setLayout(new BorderLayout(10, 20));
        Label l_east = new Label("EAST", Label.CENTER);
        l_east.setBackground(Color.red);
        … …
        Label l_center = new Label("CENTER", Label.CENTER);
        l_center.setBackground(Color.gray);
        add(l_east, BorderLayout.EAST);
        add(l_south, BorderLayout.SOUTH);
        add(l_west, BorderLayout.WEST);
        add(l_north, BorderLayout.NORTH);
        add(l_center, BorderLayout.CENTER);
    }
    public static void main(String[ ] args) {
        Frame f = new BorderLayoutTest();
        f.setVisible(true);
    }
}
```



5) FlowLayout

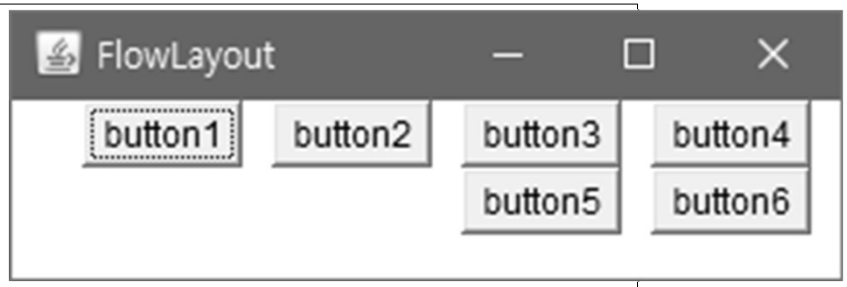
- 자식 컴포넌트를 한 줄에 차례로 배치
  - ✓ 공간이 모자라면 다음 줄에 배치함
  - ✓ 적정 크기(preferred size)로 배치함
  - ✓ 중앙/좌/우로 정렬, 수평/수직 간격을 지정할 수 있음

생성자

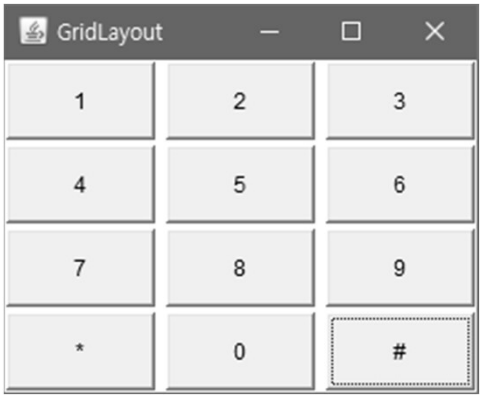
- FlowLayout( )
- FlowLayout(int align, int hgap, int vgap)
  - ✓ align은 FlowLayout.LEFT, FlowLayout.RIGHT, FlowLayout.CENTER, FlowLayout.LEADING, FlowLayout.TRAILING 중 하나

6) FlowLayout 사용 예

```
import java.awt.*;

public class FlowLayoutTest extends Frame {
public FlowLayoutTest( ) {
      super("FlowLayout");
      setSize(300, 100);
      setLayout(new FlowLayout(FlowLayout.RIGHT, 10, 0));
      add(new Button("button1"));  add(new Button("button2"));
      add(new Button("button3"));  add(new Button("button4"));
      add(new Button("button5"));  add(new Button("button6"));
      }
    public static void main(String[ ] args) {
      Frame f = new FlowLayoutTest( );
      f.setVisible(true);
    }
}
```



7) GridLayout



- 바둑판 모양의 격자에 차례로 배치함
  - ✓ 좌에서 우로(변경 가능), 위에서 아래로
- 자식 컴포넌트들의 크기는 동일

생성자

- GridLayout(int rows, int cols, int hgap, int vgap)
  - ✓ rows 또는 cols 중 하나가 0인 경우 임의 개수 가능

8) CardLayout

- 자식 컴포넌트를 카드로 간주
- 한번에 하나의 카드만 보여줌
- 생성자
  - ✓ CardLayout(int hgap, int vgap)

메소드

- void first(Container parent)
  - ✓ 컨테이너의 첫 번째 카드를 보여줌
- void next(Container parent)
- void show(Container parent, String name)
  - ✓ add( )할 때 이름을 지정한 경우, 해당 이름의 카드를 보여줌

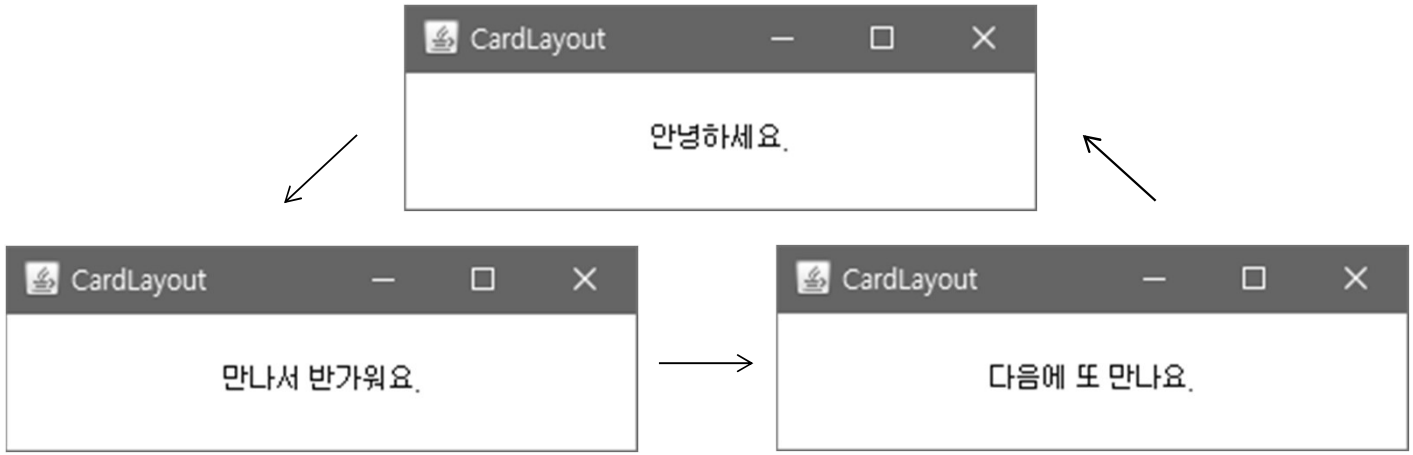
4. 배치 관리자의 사용

9) CardLayout 사용 예(1)

```
import java.awt.*;

public class CardLayoutTest extends Frame {
    CardLayout cl;
    public CardLayoutTest( ) {
        super("CardLayout");
        setSize(300, 100);
        cl = new CardLayout();
        setLayout(cl);
        add( new Label("안녕하세요.", Label.CENTER));
        add( new Label("만나서 반가워요.", Label.CENTER));
        add( new Label("다음에 또 만나요.", Label.CENTER));
     }
    public void rotate( ) throws Exception{
        cl.first(this);
        Thread.sleep(1000);
        while(true) {
                cl.next(this);
                Thread.sleep(1000);
          }
     }
```

9) CardLayout 사용 예(2)

```
public static void main(String[ ] args)
                throws Exception {
      CardLayoutTest f = new CardLayoutTest( );
      f.setVisible(true);
      f.rotate( );
    }
}
```



Java프로그래밍

다음시간안내

14강. AWT 이벤트 처리하기 (교재 13장)