제27장
AWT-GUI의 기초_Part1

자바 어플리케이션(Application)

- 콘솔 Application

  - Dos와 같은 텍스트 기반의 어플리케이션

- 윈도우 Application

  - GUI 기반의 어플리케이션
  - GUI 프로그래밍 도구

    AWT(Abstract Window Toolkit)

    → **Swing**

비교

▶ AWT

- GUI프로그래밍(윈도우 프로그래밍)을 위한 도구
- GUI프로그래밍에 필요한 다양한 컴포넌트를 제공한다.
- 자바로 구현하지 않고,OS(윈도우,리눅스,솔라리스 등) 제공하는 컴포넌트를 그대로 사용한다.

▶ Swing

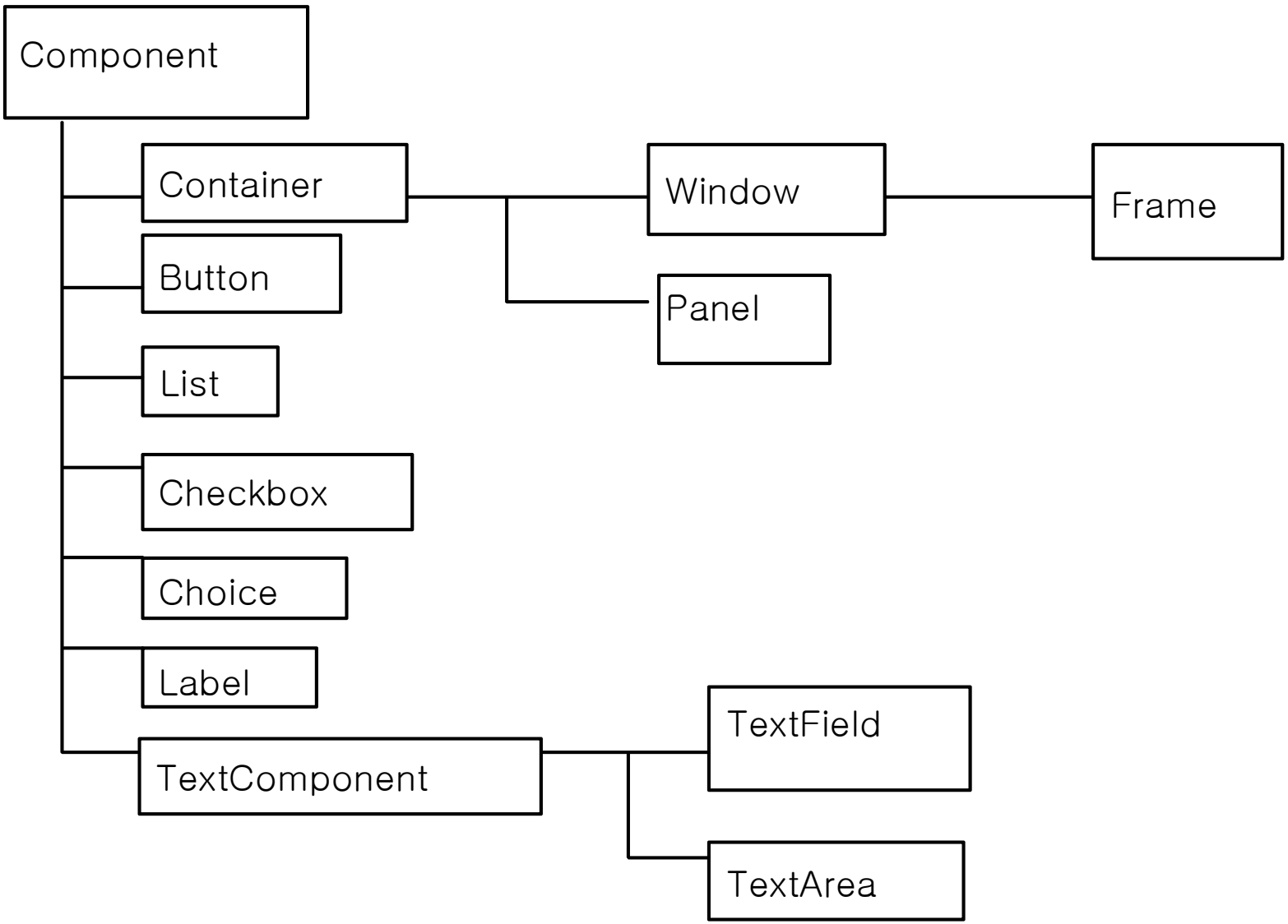
- AWT를 확장한 GUI프로그래밍 도구
- AWT보다 더 많은 종류의 컴포넌트를 제공한다.
- OS의 컴포넌트를 사용하지 않고, 순수한 Java로 구현하였다.
- Swing 의 단점은 자바가 직접 컴포넌트를 생성하기 때문에 AWT에 비해 CPU와 메모리를 상대적으로 많이 사용한다

AWT의 구성

- AWT관련 패키지는 모두 'java.awt' 로 시작한다.
- 'java.awt' 패키지와 'java.awt.event' 패키지가 AWT의 핵심이다.

- 모든 AWT컴포넌트의 최고 조상은 java.awt.Component클래스이다.
  (메뉴관련 컴포넌트 제외 - java.awt.MenuComponent가 최고조상)
- Container는 다른 컴포넌트를 담을 수 있는 컴포넌트이다.

AWT Class 계층도



컨테이너와 컴포넌트

- 컨테이너
  - 다른 GUI 컴포넌트를 포함할 수 있는 컴포넌트
  - java.awt.Container 상속
  - 다른 컨테이너에 포함될 수 있음
  - 종류들
    - AWT 컨테이너 : Panel, Frame, Applet, Dialog, Window
    - Swing 컨테이너 : JPanel JFrame, JApplet, JDialog, JWindow

- 최상위 컨테이너
  - 다른 컨테이너에 속하지 않고 독립적으로 출력가능한 컨테이너
    - JFrame, JDialog, JApplet
  - 모든 컴포넌트는 컨테이너에 포함되어야 화면에 출력 가능

- 컴포넌트
  - 컨테이너에 포함되어야 화면에 출력될 수 있는 순수 컴포넌트
  - 모든 컴포넌트는 java.awt.Component를 상속받음
  - 모든 스윙 컴포넌트는 javax.swing.JComponent를 상속받음

Panel

- 빈 평면 공간만 가지고 있는 종속적인 컨테이너.
- Panel안에 Panel을 넣을 수 있어서 컴포넌트의 다양한 배치에 유용하다.
- Panel은 프레임과 달리 titlebar나 버튼도 없고, 단지 비어있는 공간만 제공
- 컨테이너이기 때문에 레이아웃을 가질 수 있다.(기본값 : FlowLayout매니저)

레이아웃 매니저(Layout Manager)

레이아웃 매니저란?

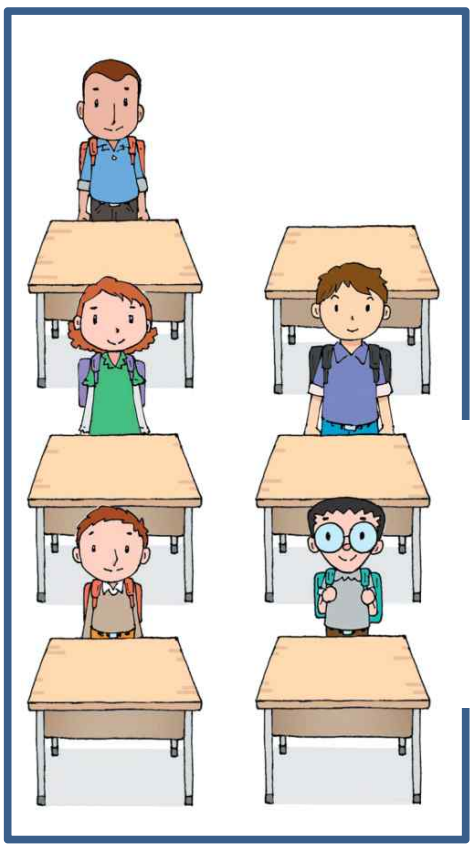
- 레이아웃 매니저는 컨테이너에 포함된 컴포넌트의 배치를 자동관리
- 레이아웃 매니저를 사용하면 컨테이너의 크기가 변경되거나 새로운 컴포넌트가 추가될 때, 컴포넌트를 재배치하는 코드를 작성할 필요가 없다.
- AWT에서는 아래와 같이 5개의 레이아웃 매니저를 제공한다.

  BorderLayout, FlowLayout, GridLayout, CardLayout

컨테이너와 기본레이아웃
FlowLayout – Panel, Applet
BorderLayout – Window, Dialog, Frame

컨테이너와 배치 개념

컨테이너(Container)



배치관리자
(Layout Manager)

컴포넌트
(Component)

1.컨테이너마다 하나의 배치관리자가 존재하며, 삽입되는 모든 컴포넌트의 위치와 크기를 결정하고 적절히 배치한다.

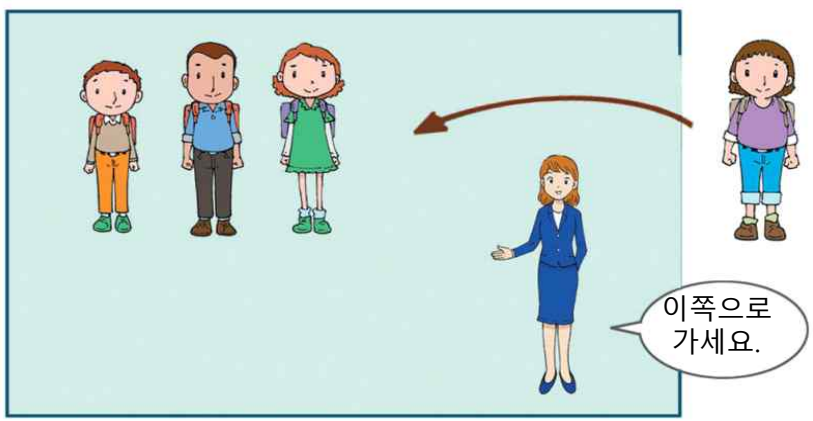
2. 컨테이너의 크기가 변하면 내부 컴포넌트들의 위치와 크기를 모두 재조절하고 재배치한다.

*

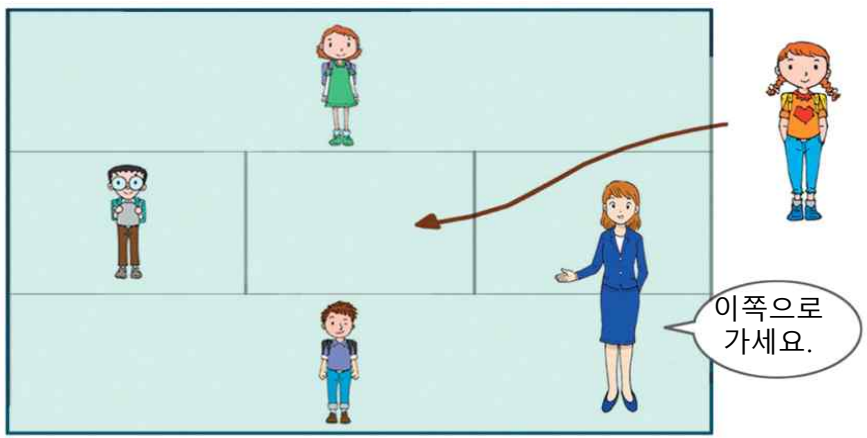
배치 관리자 대표 유형 4 가지

- java.awt 패키지에 구현되어 있음

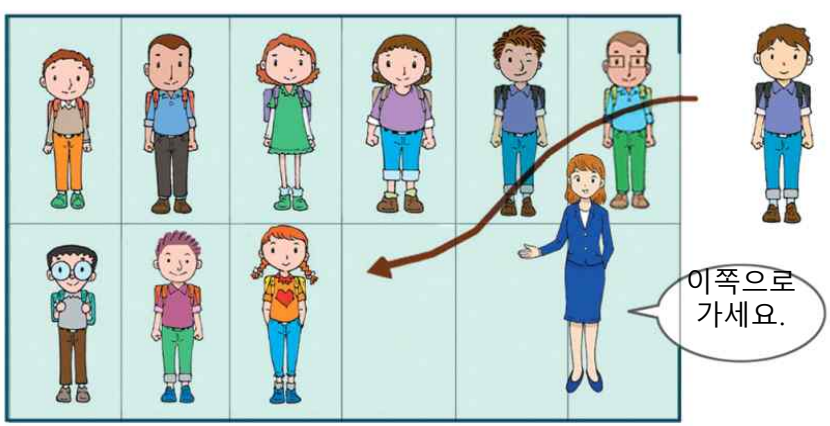
FlowLayout



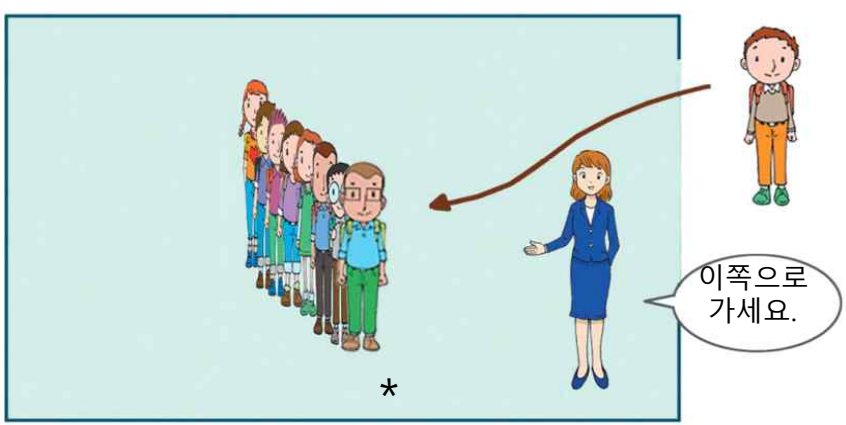
BorderLayo



GridLayout



CardLayout



컨테이너와 배치관리자

- 컨테이너의 디폴트 배치관리자
  - 컨테이너는 생성시 디폴트 배치관리자 설정

AWT와 스윙 컨테이너	디폴트 배치관리자
Window, JWindow	BorderLayout
Frame, JFrame	BorderLayout
Dialog, JDialog	BorderLayout
Panel, JPanel	FlowLayout
Applet, JApplet	FlowLayout

- 컨테이너에 새로운 배치관리자 설정
  - Container.setLayout(LayoutManager lm)
    - lm을 새로운 배치관리자로 설정

```
// JPanel 패널에 BorderLayout 배치관리자 설정

JPanel p = new JPanel();
p.setLayout(new BorderLayout());
```

```
// 컨텐트팬의 배치 관리자를 FlowLayout 으로 변경

Container c = frame.getConentPane(); // 컨텐트팬
c.setLayout(new FlowLayout());
```

*

이벤트처리(Event handling)

이벤트(Event) 란?

- 사용자 또는 프로그램에 의해 발생할 수 있는 하나의 사건.

이벤트 처리를 위한 기본 용어

- Event Source : 컴포넌트, 컨테이너 ex) Button, Pannel

- Event : 이벤트 소스별 각각의 이벤트가 존재한다.

- Event Handler : 이벤트를 처리해 주는 클래스를 의미한다.
  : 이벤트가 발생했을 때 실행될 코드를 구현해 놓은 클래스

- Event Listener : 이벤트 소스와 이벤트 핸들러를 연결

- 사용자 또는 프로그램에 의해 발생할 수 있는 하나의 사건.

종류	설명
이벤트 소스 (Event Source, 이벤트 발생지)	이벤트가 발생한 컴포넌트. 사용자가 Button을 눌렀을 때 이벤트가 발생하고, Button은 이 이벤트의 이벤트 소스가 된다.
이벤트 핸들러 (Event Handler, 이벤트 처리기)	이벤트가 발생했을 때 실행될 코드를 구현해 놓은 클래스
이벤트 리스너 (Event Listener, 이벤트 감지기)	이벤트를 감지하고 처리한다. 이벤트 핸들러를 이벤트 리스너로 이벤트 소스에 연결해야 이벤트가 발생했을 때 이벤트가 처리된다.

감사합니다!