응용 애플리케이션 개발을 위한 자바 프로그래밍

컨테이너와 레이아웃 활용하기

한국기술교육대학교
온라인평생교육원

학습내용

- 컨테이너 활용하기
- 레이아웃 활용하기

학습목표

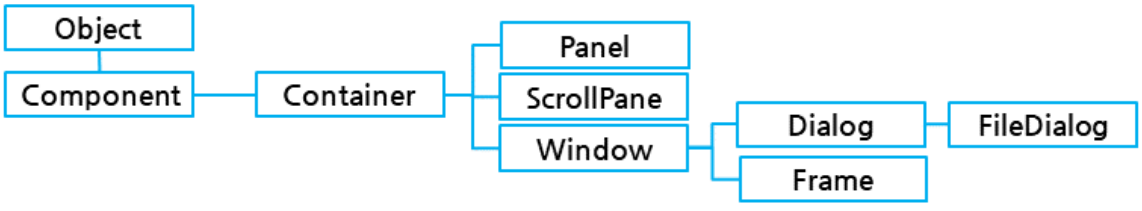
- 컨테이너의 개념을 이해하고 컨테이너를 활용하여 GUI를 구성할 수 있다.
- 레이아웃의 개념을 이해하고 레이아웃을 활용하여 GUI를 구성할 수 있다.

컨테이너 활용하기

1 컨테이너(Container)의 개요

① 컨테이너란?

② java.awt.Container 클래스의 하위 클래스들

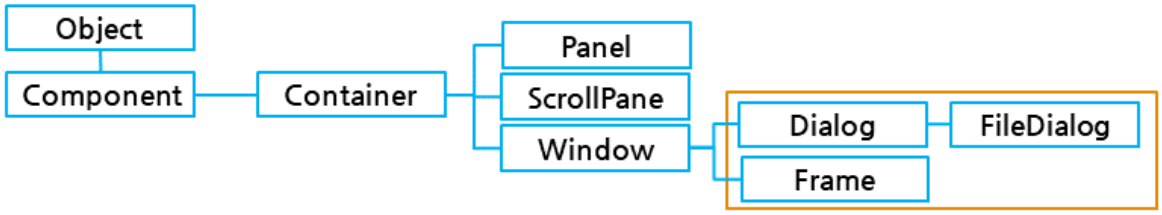


1 컨테이너(Container)의 개요

① 컨테이너란?

② java.awt.Container 클래스의 하위 클래스들

③ 단독으로 실행 가능한 클래스 : Frame, Dialog, FileDialog



컨테이너 활용하기

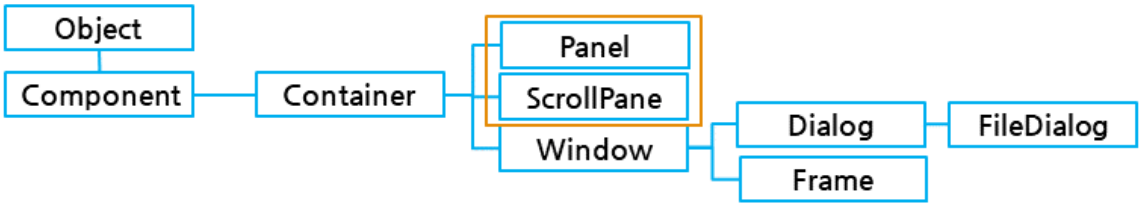
1 컨테이너(Container)의 개요

① 컨테이너란?

2 java.awt.Container 클래스의 하위 클래스들

3 단독으로 실행 가능한 클래스 : Frame, Dialog, FileDialog

4 단독으로 실행 불가능한 클래스 : Panel, ScrollPane



1 컨테이너(Container)의 개요

② Container 클래스

1 컨테이너 클래스들의 상위 클래스

컨테이너에 컴포넌트를 추가하고 삭제

주요 메소드	설 명
Component add(Component comp)	컨테이너에 컴포넌트를 추가
void remove(Component comp)	컨테이너에서 컴포넌트를 삭제
void removeAll()	컨테이너에서 모든 컴포넌트를 삭제

컨테이너 활용하기

1 컨테이너(Container)의 개요

〈3〉 Window 클래스

1 Container 클래스의 하위 클래스

컨테이너의 크기, 위치, 화면표시 여부 등

주요 메소드	설 명
void setVisible(boolean b)	컨테이너의 화면 표시 여부 지정
void setSize(Dimension d) void setSize(int w, int h)	• 컨테이너의 크기를 Dimension 객체로 설정 • 컨테이너의 크기를 폭과 넓이로 설정

1 컨테이너(Container)의 개요

〈3〉 Window 클래스

1 Container 클래스의 하위 클래스

컨테이너의 크기, 위치, 화면표시 여부 등

주요 메소드	설 명
void setLocation(Point d) void setLocation(int x, int y)	• 컨테이너의 위치를 Point 객체로 설정 • 컨테이너의 위치를 x,y 좌표값으로 설정
void dispose()	컨테이너의 리소스를 해제함

컨테이너 활용하기

1 컨테이너(Container)의 개요

〈3〉 Window 클래스

1 Container 클래스의 하위 클래스

컨테이너의 크기, 위치, 화면표시 여부 등

주요 메소드	설 명
void pack( )	컨테이너의 크기를 최적화
boolean isActive( )	컨테이너의 활성화 여부를 반환
boolean isShowing( )	컨테이너의 화면 표시 여부를 반환

2 컨테이너(Container) 클래스

〈1〉 Frame 클래스

1 윈도우 애플리케이션을 작성할 때 사용하는 컨테이너

생성자 메소드	설 명
Frame( )	타이틀이 없는 Frame 객체 생성
Frame(String title)	타이틀이 있는 Frame 객체 생성

컨테이너 활용하기

2 컨테이너(Container) 클래스

① Frame 클래스

1 윈도우 애플리케이션을 작성할 때 사용하는 컨테이너

주요 메소드	설 명
void setMenubar(MenuBar mb)	풀다운 메뉴를 Frame에 설정
void setResizable(boolean resizable)	프레임의 크기 조절 여부 설정(true : 크기 조정가능)
void setTitle(String title)	프레임의 타이틀 지정

2 컨테이너(Container) 클래스

① Frame 클래스

```
Frame f = new Frame("프레임 테스트");
f.setSize(300,300);
f.setLocation(500, 100);
f.setResizable(false);
f.setTitle("Frame Test");
f.setVisible(true);
```

컨테이너 활용하기

2 컨테이너(Container) 클래스

① Frame 클래스

```
Frame f = new Frame("프레임 테스트");
f.setSize(300,300);
f.setLocation(500, 100);
f.setResizable(false);
f.setTitle("Frame Test");
f.setVisible(true);
```

← 타이틀을 지정하여 Frame 객체 생성

2 컨테이너(Container) 클래스

① Frame 클래스

```
Frame f = new Frame("프레임 테스트");
f.setSize(300,300);
f.setLocation(500, 100);
f.setResizable(false);
f.setTitle("Frame Test");
f.setVisible(true);
```

← Frame의 크기 지정

컨테이너 활용하기

2 컨테이너(Container) 클래스

① Frame 클래스

```
Frame f = new Frame("프레임 테스트");
f.setSize(300,300);
f.setLocation(500, 100);
f.setResizable(false);
f.setTitle("Frame Test");
f.setVisible(true);
```

Frame의 위치 지정

2 컨테이너(Container) 클래스

① Frame 클래스

```
Frame f = new Frame("프레임 테스트");
f.setSize(300,300);
f.setLocation(500, 100);
f.setResizable(false);
f.setTitle("Frame Test");
f.setVisible(true);
```

Frame의 크기 조정 불가 설정

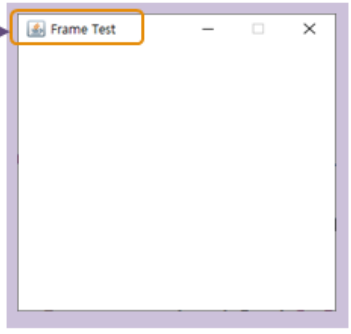
컨테이너 활용하기

2 컨테이너(Container) 클래스

〈1〉 Frame 클래스

```
Frame f = new Frame("프레임 테스트");
f.setSize(300,300);
f.setLocation(500, 100);
f.setResizable(false);
f.setTitle("Frame Test");
f.setVisible(true);
```

← Frame의 타이틀 설정

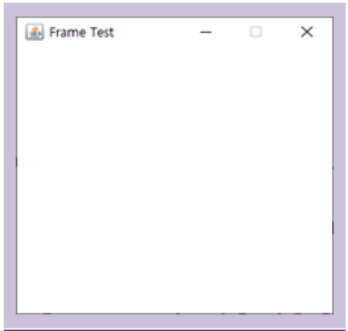


2 컨테이너(Container) 클래스

〈1〉 Frame 클래스

```
Frame f = new Frame("프레임 테스트");
f.setSize(300,300);
f.setLocation(500, 100);
f.setResizable(false);
f.setTitle("Frame Test");
f.setVisible(true);
```

← Frame 화면 표시



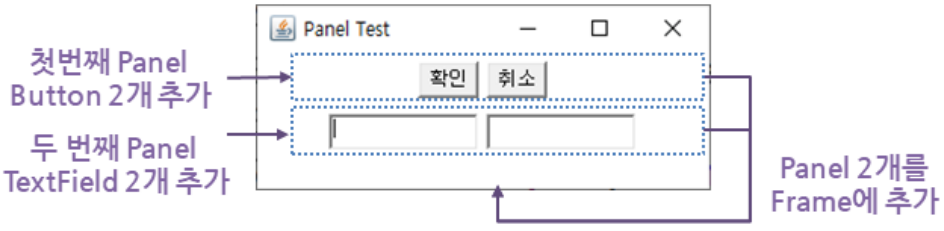
실행결과

컨테이너 활용하기

2 컨테이너(Container) 클래스

〈2〉 Panel 클래스

1. 비워져 있는 공간을 제공하는 컨테이너
2. 또 다른 Panel을 가질 수도 있음
3. Panel 객체는 다른 컨테이너(Frame, Dialog등)에 추가되어 화면에 표시됨



2 컨테이너(Container) 클래스

〈2〉 Panel 클래스

```
Frame f = new Frame( " Label Test " );
Panel north = new Panel();
north.add(new Button("확인"));
north.add(new Button("취소"));

Panel center = new Panel();
center.add(new TextField(10));
center.add(new TextField(10));

f.add(north, BorderLayout.NORTH );
f.add(center, BorderLayout.CENTER);
```

컨테이너 활용하기

2 컨테이너(Container) 클래스

〈2〉 Panel 클래스

```
Frame f = new Frame( " Label Test " );
Panel north = new Panel();
north.add(new Button("확인"));
north.add(new Button("취소"));

Panel center = new Panel();
center.add(new TextField(10));
center.add(new TextField(10));

f.add(north, BorderLayout.NORTH );
f.add(center, BorderLayout.CENTER);
```

첫 번째 Panel 객체 생성

2 컨테이너(Container) 클래스

〈2〉 Panel 클래스

```
Frame f = new Frame( " Label Test " );
Panel north = new Panel();
north.add(new Button("확인"));
north.add(new Button("취소"));

Panel center = new Panel();
center.add(new TextField(10));
center.add(new TextField(10));

f.add(north, BorderLayout.NORTH );
f.add(center, BorderLayout.CENTER);
```

첫 번째 Panel에
2개의 Button 객체 추가

컨테이너 활용하기

2 컨테이너(Container) 클래스

〈2〉 Panel 클래스

```
Frame f = new Frame( " Label Test " );
Panel north = new Panel();
north.add(new Button("확인"));
north.add(new Button("취소"));

Panel center = new Panel();
center.add(new TextField(10));
center.add(new TextField(10));

f.add(north, BorderLayout.NORTH );
f.add(center, BorderLayout.CENTER);
```

두 번째 Panel 객체 생성

2 컨테이너(Container) 클래스

〈2〉 Panel 클래스

```
Frame f = new Frame( " Label Test " );
Panel north = new Panel();
north.add(new Button("확인"));
north.add(new Button("취소"));

Panel center = new Panel();
center.add(new TextField(10));
center.add(new TextField(10));

f.add(north, BorderLayout.NORTH );
f.add(center, BorderLayout.CENTER);
```

두 번째 Panel에
2개의 TextField 객체 추가

컨테이너 활용하기

2 컨테이너(Container) 클래스

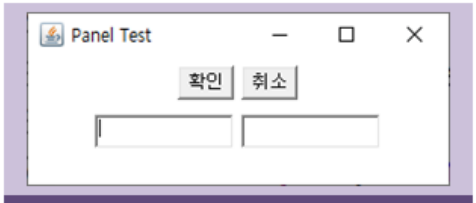
〈2〉 Panel 클래스

```
Frame f = new Frame( " Label Test " );
Panel north = new Panel();
north.add(new Button("확인"));
north.add(new Button("취소"));

Panel center = new Panel();
center.add(new TextField(10));
center.add(new TextField(10));

f.add(north, BorderLayout.NORTH );
f.add(center, BorderLayout.CENTER);
```

2개의 Panel 객체를
Frame 객체에 추가



실행결과

2 컨테이너(Container) 클래스

〈3〉 ScrollPane 클래스

1 스크롤되는 Panel 컨테이너

2 1개의 컴포넌트가 추가 가능

Panel 에 컴포넌트들을 추가하고,
Panel 객체를 ScrollPane 객체에 추가

생성자 메소드	설 명
ScrollPane()	기본 생성자 메소드
ScrollPane(int scrollbarDisplayPolicy)	스크롤바 생성 방법을 지정하여 객체 생성

컨테이너 활용하기

2 컨테이너(Container) 클래스

〈3〉 ScrollPane 클래스

1 스크롤되는 Panel 컨테이너

2 1개의 컴포넌트가 추가 가능

Panel 에 컴포넌트들을 추가하고,
Panel 객체를 ScrollPane 객체에 추가

스크롤바 생성 방법	설 명
ScrollPane.SCROLLBARS_ALWAYS	항상 수직/수평 스크롤바 표시
ScrollPane.SCROLLBARS_AS_NEEDED	필요한 경우 수직/수평 스크롤바 표시
ScrollPane.SCROLLBARS_NEVER	스크롤바 표시하지 않음

2 컨테이너(Container) 클래스

〈3〉 ScrollPane 클래스

```
Frame f = new Frame( " ScrollPane Test " );

ScrollPane sp = new ScrollPane(  스크롤바 표시 방법 지정  );
Panel p = new Panel();
for(int i=0; i<5; i++) p.add(new Button("버튼"+i));
sp.add(p);

f.add(sp);
```

컨테이너 활용하기

2 컨테이너(Container) 클래스

〈3〉 ScrollPane 클래스

```
Frame f = new Frame( " ScrollPane Test " );

ScrollPane sp = new ScrollPane( 스크롤바 표시 방법 지정 );
Panel p = new Panel();
for(int i=0; i<5; i++) p.add(new Button("버튼"+i));
sp.add(p);

f.add(sp);
```

ScrollPane 객체 생성

2 컨테이너(Container) 클래스

〈3〉 ScrollPane 클래스

```
Frame f = new Frame( " ScrollPane Test " );

ScrollPane sp = new ScrollPane( 스크롤바 표시 방법 지정 );
Panel p = new Panel();
for(int i=0; i<5; i++) p.add(new Button("버튼"+i));
sp.add(p);

f.add(sp);
```

Panel 객체 생성후
Panel에 Button 5개 추가

컨테이너 활용하기

2 컨테이너(Container) 클래스

〈3〉 ScrollPane 클래스

```
Frame f = new Frame( " ScrollPane Test " );

ScrollPane sp = new ScrollPane(  스크롤바 표시 방법 지정  );
Panel p = new Panel();
for(int i=0; i<5; i++) p.add(new Button("버튼"+i));
sp.add(p);
f.add(sp);
```

← Panel 객체를 ScrollPane 객체에 추가

2 컨테이너(Container) 클래스

〈3〉 ScrollPane 클래스

```
Frame f = new Frame( " ScrollPane Test " );

ScrollPane sp = new ScrollPane(  스크롤바 표시 방법 지정  );
Panel p = new Panel();
for(int i=0; i<5; i++) p.add(new Button("버튼"+i));
sp.add(p);

f.add(sp);
```

← ScrollPane 객체를 Frame에 추가

컨테이너 활용하기

2 컨테이너(Container) 클래스

〈3〉 ScrollPane 클래스

```
Frame f = new Frame( " ScrollPane Test " );

ScrollPane sp = new ScrollPane( 스크롤바 표시 방법 지정 );
Panel p = new Panel();
for(int i=0; i<5; i++) p.add(new Button("버튼"+i));
sp.add(p);
f.add(sp);
```

ScrollPane.SCROLLBARS_ALWAYS
(스크롤바 항상 표시)

실행결과

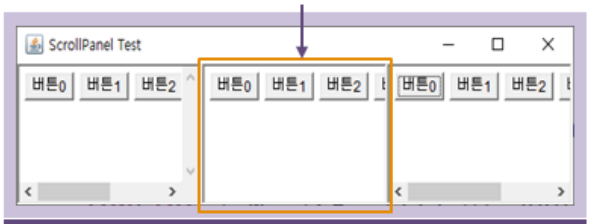
2 컨테이너(Container) 클래스

〈3〉 ScrollPane 클래스

```
Frame f = new Frame( " ScrollPane Test " );

ScrollPane sp = new ScrollPane( 스크롤바 표시 방법 지정 );
Panel p = new Panel();
for(int i=0; i<5; i++) p.add(new Button("버튼"+i));
sp.add(p);
f.add(sp);
```

ScrollPane.SCROLLBARS_NEVER
(스크롤바 표시 안함)



실행결과

컨테이너 활용하기

2 컨테이너(Container) 클래스

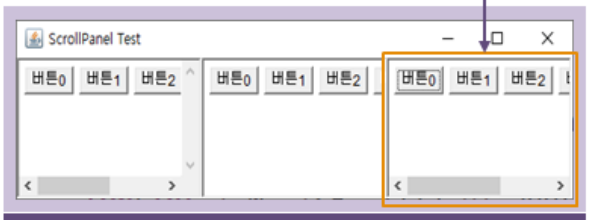
〈3〉 ScrollPane 클래스

```
Frame f = new Frame( " ScrollPane Test " );

ScrollPane sp = new ScrollPane( 스크롤바 표시 방법 지정 );
Panel p = new Panel();
for(int i=0; i<5; i++) p.add(new Button("버튼"+i));
sp.add(p);

f.add(sp);
```

ScrollPane.SCROLLBARS_AS_NEEDED
(필요시 스크롤바 표시)



실행결과

2 컨테이너(Container) 클래스

〈4〉 Dialog 클래스

1. 사용자로부터 입력을 받는데 사용하는 대화상자를 만드는 컨테이너
2. 객체 생성시 소유자를 지정해야 함

   소유자는 또 다른 Dialog, Frame, Window 객체가 됨
3. Frame 클래스의 사용법과 동일함
4. 종류

모달리스(Modaless)	대화상자를 종료하지 않고 다른 작업이 가능
모달(Modal)	대화상자를 종료하지 않고 다른 작업이 불가능

컨테이너 활용하기

2 컨테이너(Container) 클래스

④ Dialog 클래스

생성자 메소드	설 명
Dialog(Frame owner, String title, boolean modal) Dialog(Dialog owner, String title, boolean modal) Dialog(Window owner, String title, boolean modal)	소유자(Frame, Dialog, Window), 타이틀, 대화상자 종류를 지정하여 객체 생성 • modal : true (모달(Modal) 대화상자) false (모달(Modaless) 대화상자 - 기본)

2 컨테이너(Container) 클래스

④ Dialog 클래스

주요 메소드	설 명
void setModal(boolean modal)	대화상자의 종류를 지정
void setTitle(String Title)	대화상자의 타이틀을 지정
void setResizable(boolean resizable)	대화상자의 크기 조정 가능 여부를 지정

컨테이너 활용하기

2 컨테이너(Container) 클래스

④ Dialog 클래스

```
Frame f = new Frame("Dialog Test");
Dialog modal = new Dialog(f, "Modal", true);

Panel p = new Panel();
p.add(new Button("OK"));
p.add(new Button("CANCEL"));

modal.add(p);

f.setVisible(true);
modal.setVisible(true);
```

2 컨테이너(Container) 클래스

④ Dialog 클래스

```
Frame f = new Frame("Dialog Test");
Dialog modal = new Dialog(f, "Modal", true);
```

← 모달(Modal) 대화상자 객체 생성

```
Panel p = new Panel();
p.add(new Button("OK"));
p.add(new Button("CANCEL"));

modal.add(p);

f.setVisible(true);
modal.setVisible(true);
```

컨테이너 활용하기

2 컨테이너(Container) 클래스

④ Dialog 클래스

```
Frame f = new Frame("Dialog Test");
Dialog modal = new Dialog(f, "Modal", true);

Panel p = new Panel();
p.add(new Button("OK"));
p.add(new Button("CANCEL"));

modal.add(p);

f.setVisible(true);
modal.setVisible(true);
```

false 지정시
모달리스(Modaless) 대화상자 객체 생성

2 컨테이너(Container) 클래스

④ Dialog 클래스

```
Frame f = new Frame("Dialog Test");
Dialog modal = new Dialog(f, "Modal", true);

Panel p = new Panel();
p.add(new Button("OK"));
p.add(new Button("CANCEL"));

modal.add(p);

f.setVisible(true);
modal.setVisible(true);
```

2개의 Button 객체가 있는 Panel 객체 생성

컨테이너 활용하기

2 컨테이너(Container) 클래스

④ Dialog 클래스

```
Frame f = new Frame("Dialog Test");
Dialog modal = new Dialog(f, "Modal", true);

Panel p = new Panel();
p.add(new Button("OK"));
p.add(new Button("CANCEL"));

modal.add(p);
```

← Panel 객체를 Dialog 객체에 추가

```
f.setVisible(true);
modal.setVisible(true);
```

2 컨테이너(Container) 클래스

④ Dialog 클래스

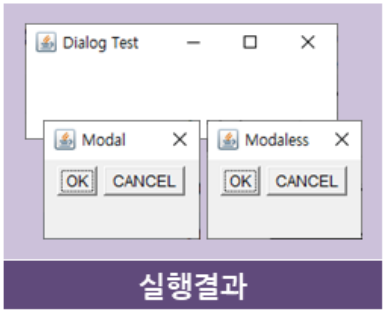
```
Frame f = new Frame("Dialog Test");
Dialog modal = new Dialog(f, "Modal", true);

Panel p = new Panel();
p.add(new Button("OK"));
p.add(new Button("CANCEL"));

modal.add(p);

f.setVisible(true);
modal.setVisible(true);
```

← Frame과 Dialog 객체를 화면에 표시



실행결과

컨테이너 활용하기

2 컨테이너(Container) 클래스

⑤ FileDialog 클래스

1. 파일을 열거나 저장할 때 사용하는 대화상자
2. 객체 생성시 소유자를 지정해야 함

   소유자는 또 다른 Dialog, Frame 객체가 됨

3. 모달(Modal) 대화상자

2 컨테이너(Container) 클래스

⑤ FileDialog 클래스

생성자 메소드	설 명
FileDialog(Frame owner) FileDialog(Dialog owner)	소유자(Frame, Dialog)를 지정하여 객체 생성
FileDialog(Frame owner, String title) FileDialog(Dialog owner, String title)	소유자(Frame, Dialog)와 타이틀을 지정하여 객체 생성

컨테이너 활용하기

2 컨테이너(Container) 클래스

⑤ FileDialog 클래스

생성자 메소드	설 명
FileDialog (Frame owner, String title, int mode) FileDialog(Dialog owner, String title, int mode)	소유자(Frame, Dialog), 타이틀, 대화상자 모드를 지정하여 객체 생성 • mode : FileDialog.LOAD (파일 불러오기) FileDialog.SAVE (파일 저장하기)

2 컨테이너(Container) 클래스

⑤ FileDialog 클래스

주요 메소드	설 명
String getDirectory( )	선택된 폴더명을 문자열로 반환
String getFile( )	선택된 파일명을 문자열로 반환
void setDirectory(String dir)	대화상자에 표시할 폴더명 지정

컨테이너 활용하기

2 컨테이너(Container) 클래스

〈5〉 FileDialog 클래스

주요 메소드	설 명
void setFile(String file)	대화상자에 표시할 파일명 지정
void setMode(int mode)	대화상자 모드를 지정
void setTitle(String title)	대화상자 타이틀 지정

2 컨테이너(Container) 클래스

〈5〉 FileDialog 클래스

```
Frame f = new Frame("FileDialog Test");
f.setVisible(true);

FileDialog load = new FileDialog(f, "파일 불러오기", FileDialog.LOAD);
load.setVisible(true);

System.out.println("Folder : "+ load.getDirectory());
System.out.println("File : "+ load.getFile());
```

컨테이너 활용하기

2 컨테이너(Container) 클래스

⑤ FileDialog 클래스

```
Frame f = new Frame("FileDialog Test");
f.setVisible(true);

FileDialog load = new FileDialog(f, "파일 불러오기", FileDialog.LOAD);
load.setVisible(true);

System.out.println("Folder : "+ load.getDirectory());
System.out.println("File : "+ load.getFile());
```

파일 불러오기 FileDialog 객체 생성하고 화면에 표시

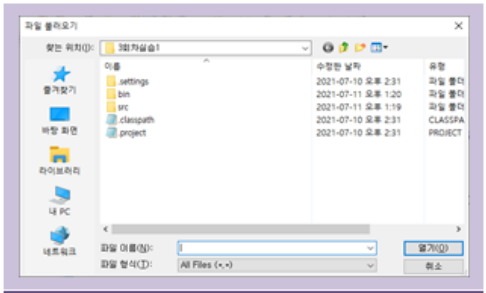
2 컨테이너(Container) 클래스

⑤ FileDialog 클래스

```
Frame f = new Frame("FileDialog Test");
f.setVisible(true);

FileDialog load = new FileDialog(f, "파일 불러오기", FileDialog.LOAD);
load.setVisible(true);

System.out.println("Folder : "+ load.getDirectory());
System.out.println("File : "+ load.getFile());
```



실행결과

컨테이너 활용하기

2 컨테이너(Container) 클래스

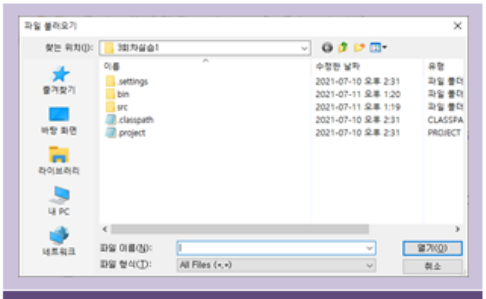
⑤ FileDialog 클래스

```
Frame f = new Frame("FileDialog Test");
f.setVisible(true);

FileDialog load = new FileDialog(f, "파일 불러오기", FileDialog.LOAD);
load.setVisible(true);

System.out.println("Folder : "+ load.getDirectory());
System.out.println("File : "+ load.getFile());
```

선택된 폴더명과 파일명을 화면에 출력



실행결과

2 컨테이너(Container) 클래스

⑤ FileDialog 클래스

```
Frame f = new Frame("FileDialog Test");
f.setVisible(true);

FileDialog save = new FileDialog(f, "파일 저장하기", FileDialog.SAVE);

save.setDirectory("C:\\Program Files");
save.setFile("java.txt");

save.setVisible(true);
```

컨테이너 활용하기

2 컨테이너(Container) 클래스

⑤ FileDialog 클래스

```
Frame f = new Frame("FileDialog Test");
f.setVisible(true);

FileDialog save = new FileDialog(f, "파일 저장하기", FileDialog.SAVE);

save.setDirectory("C:\\Program Files");
save.setFile("java.txt");

save.setVisible(true);
```

파일 저장하기 FileDialog 객체 생성

2 컨테이너(Container) 클래스

⑤ FileDialog 클래스

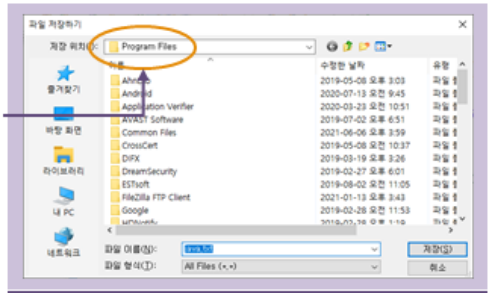
```
Frame f = new Frame("FileDialog Test");
f.setVisible(true);

FileDialog save = new FileDialog(f, "파일 저장하기", FileDialog.SAVE);

save.setDirectory("C:\\Program Files");
save.setFile("java.txt");

save.setVisible(true);
```

대화상자에 표시할 폴더 지정



실행결과

컨테이너 활용하기

2 컨테이너(Container) 클래스

⑤ FileDialog 클래스

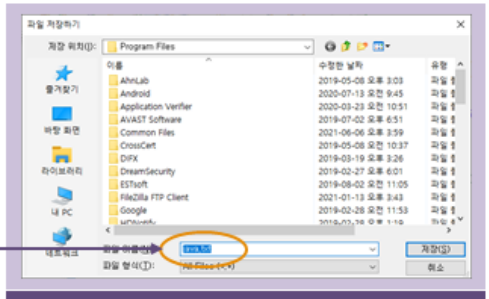
```
Frame f = new Frame("FileDialog Test");
f.setVisible(true);

FileDialog save = new FileDialog(f, "파일 저장하기", FileDialog.SAVE);

save.setDirectory("C:\\Program Files");
save.setFile("java.txt");

save.setVisible(true);
```

대화상자에 표시할 파일명 지정



실행결과

2 컨테이너(Container) 클래스

⑤ FileDialog 클래스

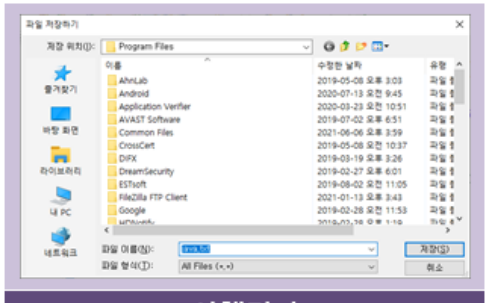
```
Frame f = new Frame("FileDialog Test");
f.setVisible(true);

FileDialog save = new FileDialog(f, "파일 저장하기", FileDialog.SAVE);

save.setDirectory("C:\\Program Files");
save.setFile("java.txt");

save.setVisible(true);
```

저장하기 대화상자를 화면에 표시



실행결과

컨테이너 활용하기

컨테이너와 레이아웃 활용하기 | 컨테이너 활용하기

실습하기

컨테이너와 레이아웃 활용하기 | 컨테이너 활용하기

실습순서

1. 컨테이너(Container) 클래스 실습
   1) Frame 클래스 실습
   2) Panel 클래스 실습
   3) ScrollPane 클래스 실습
   4) Dialog 클래스 실습
   5) FileDialog 클래스 실습

유의사항

- JDK와 이클립스를 설치한 후 실습이 가능함
- 본인이 원하는 작업 폴더를 미리 정해 놓은 다음 실습하기
- 작업 폴더는 C드라이브에 지정하기 보다는 D드라이브나 외장하드디스크를 활용하는 것을 추천함

※ 제공되는 실습 코드를 다운받아 실습해보시기 바랍니다.

레이아웃 활용하기

1 레이아웃(Layout) 이해하기

〈1〉 레이아웃이란?

1 컴포넌트를 컨테이너에 배치하는 방식

2 레이아웃 관리자 클래스

컨테이너에 배치된 컴포넌트들의 배치를 자동으로 관리하는 클래스

종류 : FlowLayout, BorderLayout, GridLayout,
GridbagLayout, CardLayoout

3 컨테이너에서 레이아웃 관리자 클래스를 사용하지 않아도 됨

Null 레이아웃

1 레이아웃(Layout) 이해하기

〈2〉 레이아웃 관리자 클래스를 이용한 컴포넌트의 배치 순서

1 컨테이너 클래스의 객체 생성

2 레이아웃 관리자 클래스의 객체 생성

3 컨테이너에 레이아웃 관리자를 설정 : setLayout( ) 메소드

4 컨테이너에 컴포넌트 추가

레이아웃 활용하기

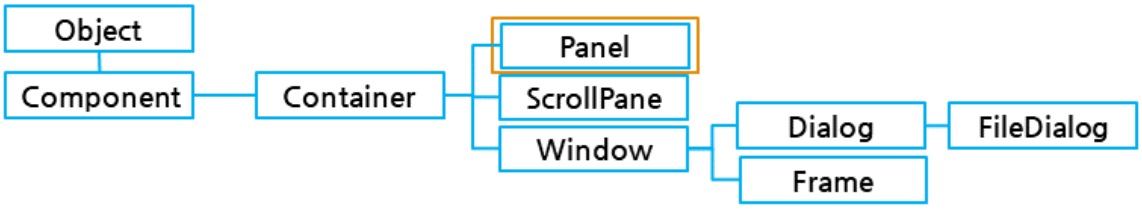
1 레이아웃(Layout) 이해하기

〈3〉 기본 레이아웃 관리자

1 컨테이너는 기본으로 사용하는 레이아웃 관리자가 정해져 있음

2 레이아웃 관리자를 지정하지 않으면, 기본 레이아웃 관리자가 적용됨

Panel	FlowLayout



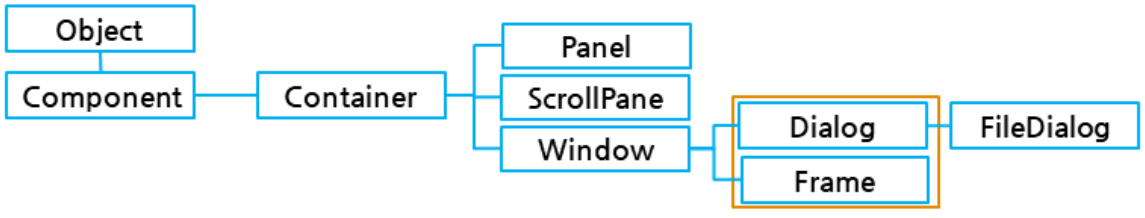
1 레이아웃(Layout) 이해하기

〈3〉 기본 레이아웃 관리자

1 컨테이너는 기본으로 사용하는 레이아웃 관리자가 정해져 있음.

2 레이아웃 관리자를 지정하지 않으면, 기본 레이아웃 관리자가 적용됨.

Panel	FlowLayout
Frame, Dialog	BorderLayout



레이아웃 활용하기

2 레이아웃 관리자 이해하기

〈1〉 FlowLayout 클래스

1 컴포넌트가 추가되는 순서대로 왼쪽에서 오른쪽으로 추가

오른쪽으로 더 이상 배치 할 곳이 없으면 아래 줄의 왼쪽으로 추가

생성자 메소드	설 명
FlowLayout()	중앙정렬, 수평/수직 간격 : 5픽셀
FlowLayout(int align)	정렬방식 지정, 수평/수직 간격 : 5픽셀 • align : FlowLayout.LEFT (왼쪽 정렬) FlowLayout.RIGHT (오른쪽 정렬) FlowLayout.CENTER (중앙 정렬)

2 레이아웃 관리자 이해하기

〈1〉 FlowLayout 클래스

1 컴포넌트가 추가되는 순서대로 왼쪽에서 오른쪽으로 추가

오른쪽으로 더 이상 배치 할 곳이 없으면 아래 줄의 왼쪽으로 추가

생성자 메소드	설 명
FlowLayout(int align, int hgap, int vgap)	정렬방식 지정 • hgap : 수평 간격 지정 • vgap : 수직 간격 지정

레이아웃 활용하기

2 레이아웃 관리자 이해하기

〈1〉 FlowLayout 클래스

```
Frame f = new Frame("FlowLayout Test");


for(int i=0; i<10; i++) f.add(new Button("버튼 "+i));

f.setSize(300,200);
f.setVisible(true);
```

2 레이아웃 관리자 이해하기

〈1〉 FlowLayout 클래스

```
Frame f = new Frame("FlowLayout Test");
```

레이아웃 관리자 지정

```
for(int i=0; i<10; i++) f.add(new Button("버튼 "+i));

f.setSize(300,200);
f.setVisible(true);


f.setLayout(new FlowLayout());
f.setLayout(new FlowLayout(FlowLayout.LEFT));
f.setLayout(new FlowLayout(FlowLayout.RIGHT, 10, 10));
```

레이아웃 활용하기

2 레이아웃 관리자 이해하기

〈1〉 FlowLayout 클래스

```
Frame f = new Frame("FlowLayout Test");
```

레이아웃 관리자 지정

10개의 Button 객체를 Frame 객체에 추가

```
for(int i=0; i<10; i++) f.add(new Button("버튼 "+i));

f.setSize(300,200);
f.setVisible(true);

f.setLayout(new FlowLayout());
f.setLayout(new FlowLayout(FlowLayout.LEFT));
f.setLayout(new FlowLayout(FlowLayout.RIGHT, 10, 10));
```

2 레이아웃 관리자 이해하기

〈1〉 FlowLayout 클래스

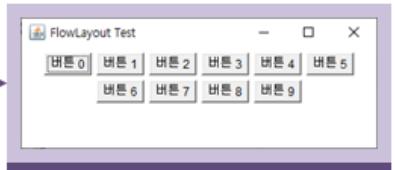
```
Frame f = new Frame("FlowLayout Test");
```

레이아웃 관리자 지정

```
for(int i=0; i<10; i++) f.add(new Button("버튼 "+i));

f.setSize(300,200);
f.setVisible(true);

f.setLayout(new FlowLayout());
f.setLayout(new FlowLayout(FlowLayout.LEFT));
f.setLayout(new FlowLayout(FlowLayout.RIGHT, 10, 10));
```



실행결과

레이아웃 활용하기

2 레이아웃 관리자 이해하기

〈1〉 FlowLayout 클래스

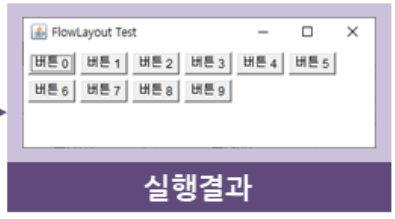
```
Frame f = new Frame("FlowLayout Test");
```

레이아웃 관리자 지정

```
for(int i=0; i<10; i++) f.add(new Button("버튼 "+i));

f.setSize(300,200);
f.setVisible(true);

f.setLayout(new FlowLayout());
f.setLayout(new FlowLayout(FlowLayout.LEFT));
f.setLayout(new FlowLayout(FlowLayout.RIGHT, 10, 10));
```



실행결과

2 레이아웃 관리자 이해하기

〈1〉 FlowLayout 클래스

```
Frame f = new Frame("FlowLayout Test");
```

레이아웃 관리자 지정

```
for(int i=0; i<10; i++) f.add(new Button("버튼 "+i));

f.setSize(300,200);
f.setVisible(true);

f.setLayout(new FlowLayout());
f.setLayout(new FlowLayout(FlowLayout.LEFT));
f.setLayout(new FlowLayout(FlowLayout.RIGHT, 10, 10));
```

실행결과

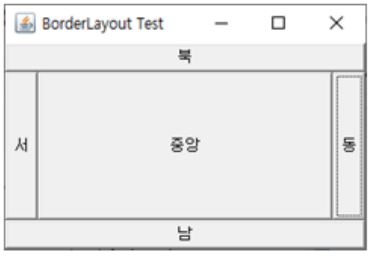
레이아웃 활용하기

2 레이아웃 관리자 이해하기

〈2〉 BorderLayout 클래스

1 동, 서, 남, 북, 중앙을 지정하여 컴포넌트를 배치

생성자 메소드	설 명
BorderLayout()	기본 생성자 메소드 수평/수직 간격 없음

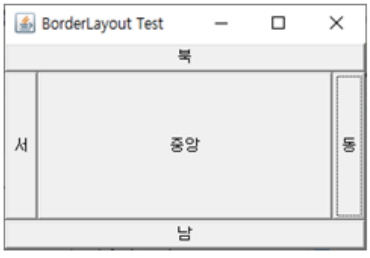


2 레이아웃 관리자 이해하기

〈2〉 BorderLayout 클래스

1 동, 서, 남, 북, 중앙을 지정하여 컴포넌트를 배치

생성자 메소드	설 명
BorderLayout()	기본 생성자 메소드 수평/수직 간격 없음
BorderLayout(int vgap, int hgap)	수평/수직 간격을 지정하여 객체 생성 • hgap : 수평 간격 지정 • vgap : 수직 간격 지정



레이아웃 활용하기

2 레이아웃 관리자 이해하기

⑵ BorderLayout 클래스

2 컴포넌트 배치

Container 클래스의 add() 메소드를 이용하여 컴포넌트를 배치

Container 클래스의 메소드	설 명
Component add(Component comp, int index)	컨테이너에 컴포넌트를 추가하는 메소드 • index : 컴포넌트가 추가될 위치 지정 상수
Component add(String name, Component comp)	컨테이너에 컴포넌트를 추가하는 메소드 • name : 컴포넌트가 추가될 위치 지정 문자열

2 레이아웃 관리자 이해하기

⑵ BorderLayout 클래스

2 컴포넌트 배치

위치를 지정하는 상수와 문자열

위치 지정 상수	위치 지정 문자열	설 명
BorderLayout.EAST	"East"	우측 영역의 위치 지정
BorderLayout.WEST	"West"	좌측 영역의 위치 지정
BorderLayout. SOUTH	"South"	하단 영역의 위치 지정
BorderLayout. NORTH	"North"	상단 영역의 위치 지정
BorderLayout. CENTER	"Center"	중앙 영역의 위치 지정

레이아웃 활용하기

2 레이아웃 관리자 이해하기

〈2〉 BorderLayout 클래스

```
Frame f = new Frame("BorderLayout Test");
f.setLayout(new BorderLayout());

f.add(new Button("동"),BorderLayout.EAST);
f.add(new Button("서"),BorderLayout.WEST);
f.add(new Button("남"),BorderLayout.SOUTH);
f.add(new Button("북"),BorderLayout.NORTH);
f.add(new Button("중앙"),BorderLayout.CENTER);
```

2 레이아웃 관리자 이해하기

〈2〉 BorderLayout 클래스

```
Frame f = new Frame("BorderLayout Test");
f.setLayout(new BorderLayout());

f.add(new Button("동"),BorderLayout.EAST);
f.add(new Button("서"),BorderLayout.WEST);
f.add(new Button("남"),BorderLayout.SOUTH);
f.add(new Button("북"),BorderLayout.NORTH);
f.add(new Button("중앙"),BorderLayout.CENTER);
```

Frame에 BorderLayout 지정
- Frame은 기본 레이아웃이 BorderLayout이므로 지정하지 않아도 됨

레이아웃 활용하기

2 레이아웃 관리자 이해하기

〈2〉 BorderLayout 클래스

```
Frame f = new Frame("BorderLayout Test");
f.setLayout(new BorderLayout());

f.add(new Button("동"),BorderLayout.EAST);
f.add(new Button("서"),BorderLayout.WEST);
f.add(new Button("남"),BorderLayout.SOUTH);
f.add(new Button("북"),BorderLayout.NORTH);
f.add(new Button("중앙"),BorderLayout.CENTER);
```

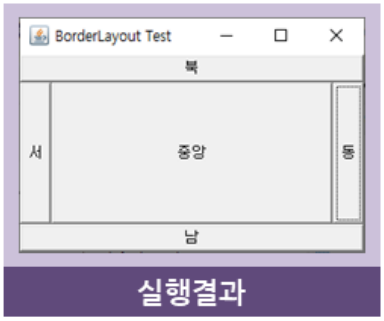
동, 서, 남, 북, 중앙 위치에 각각 Button 객체를 추가

2 레이아웃 관리자 이해하기

〈2〉 BorderLayout 클래스

```
Frame f = new Frame("BorderLayout Test");
f.setLayout(new BorderLayout());

f.add(new Button("동"),BorderLayout.EAST);
f.add(new Button("서"),BorderLayout.WEST);
f.add(new Button("남"),BorderLayout.SOUTH);
f.add(new Button("북"),BorderLayout.NORTH);
f.add(new Button("중앙"),BorderLayout.CENTER);
```



실행결과

레이아웃 활용하기

2 레이아웃 관리자 이해하기

〈3〉 GridLayout 클래스

1 행과 열의 수를 지정하여 표와 같은 영역을 만들어 컴포넌트를 배치

2 컴포넌트 추가 순서

좌측에서 우측, 우측이 없으면 다음 행의 첫 칸

생성자 메소드	설 명
GridLayout()	1행 1열의 객체를 생성
GridLayout(int rows, int cols)	행과 열의 수를 지정하여 객체를 생성 • rows : 행의 수 • cols : 열의 수

2 레이아웃 관리자 이해하기

〈3〉 GridLayout 클래스

1 행과 열의 수를 지정하여 표와 같은 영역을 만들어 컴포넌트를 배치

2 컴포넌트 추가 순서

좌측에서 우측, 우측이 없으면 다음 행의 첫 칸

생성자 메소드	설 명
GridLayout(int rows, int cols, int hgap, int vgap)	행과 열의 수, 수평/수직 간격을 지정하여 객체를 생성 • rows : 행의 수 • cols : 열의 수 • hgap : 수평 간격 • vgap : 수직 간격

레이아웃 활용하기

2 레이아웃 관리자 이해하기

〈3〉 GridLayout 클래스

```
Frame f = new Frame( " GridLayout Test " );

f.setLayout(new GridLayout(3,4));

for(int i=1; i<=12; i++) f.add(new Button("버튼"+i));
```

2 레이아웃 관리자 이해하기

〈3〉 GridLayout 클래스

```
Frame f = new Frame( " GridLayout Test " );

f.setLayout(new GridLayout(3,4));   ← 3행 4열의 GridLayout을 Frame에 지정

for(int i=1; i<=12; i++) f.add(new Button("버튼"+i));
```

레이아웃 활용하기

2 레이아웃 관리자 이해하기

〈3〉 GridLayout 클래스

```
Frame f = new Frame( " GridLayout Test " );

f.setLayout(new GridLayout(3,4));

for(int i=1; i<=12; i++) f.add(new Button("버튼"+i));
```

12개의 버튼을 Frame에 추가

2 레이아웃 관리자 이해하기

〈3〉 GridLayout 클래스

```
Frame f = new Frame( " GridLayout Test " );

f.setLayout(new GridLayout(3,4));

for(int i=1; i<=12; i++) f.add(new Button("버튼"+i));
```

실행결과

레이아웃 활용하기

2 레이아웃 관리자 이해하기

〈4〉 GridBagLayout 클래스

1 GridLayout 클래스가 확장된 레이아웃

2 컴포넌트의 배치 속성 설정

GridBagConstraints 클래스

생성자 메소드	설 명
GridBagLayout()	기본 생성자 메소드

2 레이아웃 관리자 이해하기

〈4〉 GridBagLayout 클래스

1 GridLayout 클래스가 확장된 레이아웃

2 컴포넌트의 배치 속성 설정

GridBagConstraints 클래스

주요 메소드	설 명
void setConstraints(Component comp, GridBagConstraints constraints)	추가될 컴포넌트와 배치 속성을 설정한 GridBagConstraints 객체를 지정

레이아웃 활용하기

2 레이아웃 관리자 이해하기

〈4〉 GridBagLayout 클래스

3 GridBagConstraints 클래스

생성자 메소드	설 명
GridBagConstraints( )	기본 생성자 메소드
주요 변수	**설 명**
int gridx	컴포넌트가 추가될 셀의 x 좌표 값
int gridy	컴포넌트가 추가될 셀의 y 좌표 값
int gridwidth	컴포넌트가 차지할 열의 개수

2 레이아웃 관리자 이해하기

〈4〉 GridBagLayout 클래스

3 GridBagConstraints 클래스

주요 변수	설 명
int gridheight	컴포넌트가 차지할 행의 개수
int fill	셀 크기와 컴포넌트 크기의 조정 방법 지정
double weightx	컨테이너 크기와 전체 컴포넌트 크기의 좌우 크기 조정 방법 지정
double weighty	컨테이너 크기와 전체 컴포넌트 크기의 상하 크기 조정 방법 지정

레이아웃 활용하기

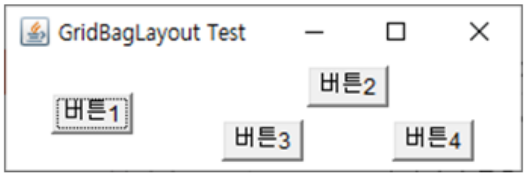
2 레이아웃 관리자 이해하기

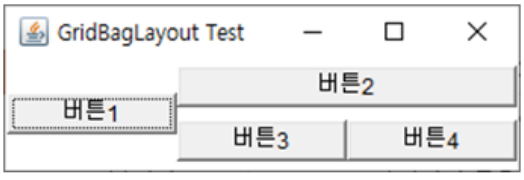
〈4〉 GridBagLayout 클래스

3 GridBagConstraints 클래스

fill 변수에 지정할 수 있는 주요 상수

주요 상수	설 명
GridBagConstraints.NONE	셀의 크기에 따라 컴포넌트의 크기를 조정하지 않음
GridBagConstraints.HORIZONTAL	셀의 크기에 따라 컴포넌트의 좌우 크기만 자동으로 조정





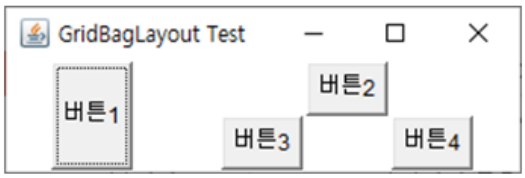
2 레이아웃 관리자 이해하기

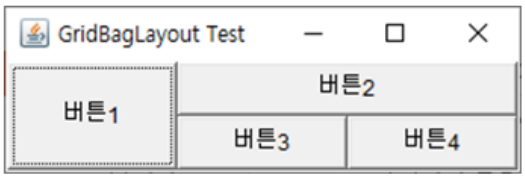
〈4〉 GridBagLayout 클래스

3 GridBagConstraints 클래스

fill 변수에 지정할 수 있는 주요 상수

주요 상수	설 명
GridBagConstraints.VERTICAL	셀의 크기에 따라 컴포넌트의 상하 크기만 자동으로 조정
GridBagConstraints.BOTH	셀의 크기에 따라 컴포넌트의 상하, 좌우 크기 모두 자동으로 조정





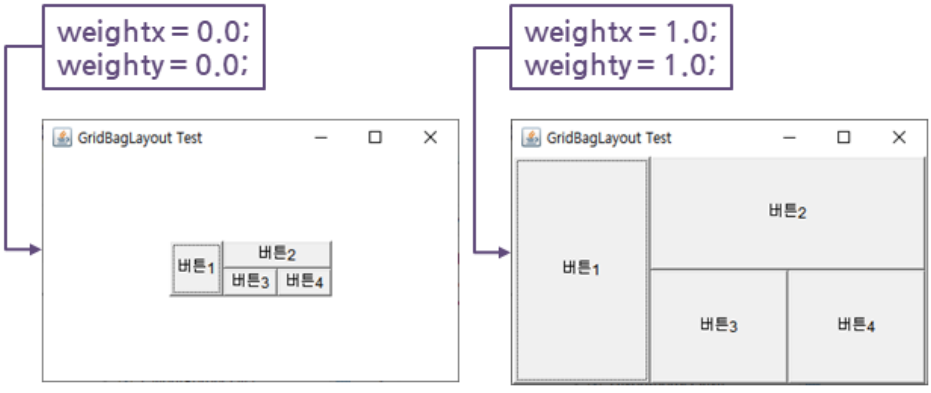
레이아웃 활용하기

2 레이아웃 관리자 이해하기

⟨4⟩ GridBagLayout 클래스

3 GridBagConstraints 클래스

weightx, weighty 변수에 지정할 수 있는 값 (0.0~1.0)



2 레이아웃 관리자 이해하기

⟨4⟩ GridBagLayout 클래스

4 GridBagLayout 프로그래밍 방법

```
Frame f = new Frame("GridBagLayout Test");
GridBagLayout gbl = new GridBagLayout();
f.setLayout(gbl);

GridBagConstraints gbc = new GridBagConstraints();

gbc.fill = GridBagConstraints.BOTH;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 1;
gbc.gridheight = 2;

Button b1 = new Button("버튼1");
gbl.setConstraints(b1, gbc);
f.add(b1);
```

레이아웃 활용하기

2 레이아웃 관리자 이해하기

④ GridBagLayout 클래스

4 GridBagLayout 프로그래밍 방법

1. GridBagLayout 객체를 생성하고 Frame에 레이아웃 설정

```
Frame f = new Frame("GridBagLayout Test");
GridBagLayout gbl = new GridBagLayout();
f.setLayout(gbl);

GridBagConstraints gbc = new GridBagConstraints();

gbc.fill = GridBagConstraints.BOTH;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 1;
gbc.gridheight = 2;

Button b1 = new Button("버튼1");
gbl.setConstraints(b1, gbc);
f.add(b1);
```

2 레이아웃 관리자 이해하기

④ GridBagLayout 클래스

4 GridBagLayout 프로그래밍 방법

```
Frame f = new Frame("GridBagLayout Test");
GridBagLayout gbl = new GridBagLayout();
f.setLayout(gbl);

GridBagConstraints gbc = new GridBagConstraints();

gbc.fill = GridBagConstraints.BOTH;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 1;
gbc.gridheight = 2;

Button b1 = new Button("버튼1");
gbl.setConstraints(b1, gbc);
f.add(b1);
```

2. GridBagConstraints 객체 생성

레이아웃 활용하기

2 레이아웃 관리자 이해하기

⟨4⟩ GridBagLayout 클래스

4 GridBagLayout 프로그래밍 방법

```
Frame f = new Frame("GridBagLayout Test");
GridBagLayout gbl = new GridBagLayout();
f.setLayout(gbl);

GridBagConstraints gbc = new GridBagConstraints();

gbc.fill = GridBagConstraints.BOTH;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 1;
gbc.gridheight = 2;

Button b1 = new Button("버튼1");
gbl.setConstraints(b1, gbc);
f.add(b1);
```

3. GridBagConstraints 객체의 속성 지정

2 레이아웃 관리자 이해하기

⟨4⟩ GridBagLayout 클래스

4 GridBagLayout 프로그래밍 방법

```
Frame f = new Frame("GridBagLayout Test");
GridBagLayout gbl = new GridBagLayout();
f.setLayout(gbl);

GridBagConstraints gbc = new GridBagConstraints();

gbc.fill = GridBagConstraints.BOTH;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 1;
gbc.gridheight = 2;

Button b1 = new Button("버튼1");
gbl.setConstraints(b1, gbc);
f.add(b1);
```

4. 추가할 컴포넌트 생성

레이아웃 활용하기

2 레이아웃 관리자 이해하기

⟨4⟩ GridBagLayout 클래스

4 GridBagLayout 프로그래밍 방법

```
Frame f = new Frame("GridBagLayout Test");
GridBagLayout gbl = new GridBagLayout();
f.setLayout(gbl);

GridBagConstraints gbc = new GridBagConstraints();

gbc.fill = GridBagConstraints.BOTH;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 1;
gbc.gridheight = 2;

Button b1 = new Button("버튼1");
gbl.setConstraints(b1, gbc);
f.add(b1);
```

5. 추가할 컴포넌트와 GridBagConstraints 객체를 GridBagLayout 객체에 설정

2 레이아웃 관리자 이해하기

⟨4⟩ GridBagLayout 클래스

4 GridBagLayout 프로그래밍 방법

```
Frame f = new Frame("GridBagLayout Test");
GridBagLayout gbl = new GridBagLayout();
f.setLayout(gbl);

GridBagConstraints gbc = new GridBagConstraints();

gbc.fill = GridBagConstraints.BOTH;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 1;
gbc.gridheight = 2;

Button b1 = new Button("버튼1");
gbl.setConstraints(b1, gbc);
f.add(b1);
```

6. 컴포넌트를 Frame에 추가

레이아웃 활용하기

2 레이아웃 관리자 이해하기

5 CardLayout 클래스

1. 한 개의 화면에 여러 개의 컨테이너를 중복해서 표시할 수 있는 레이아웃 관리자
2. 일반적으로 컨테이너는 Panel을 사용
3. 각각의 컨테이너를 Card라고 함

생성자 메소드	설 명
CardLayout()	기본 생성자 메소드, 컨테이너와 Card의 간격은 0
CardLayout(int hgap, int vgap)	컨테이너와 Card와의 간격을 지정하여 객체 생성

2 레이아웃 관리자 이해하기

5 CardLayout 클래스

주요 메소드	설 명
void first(Container parent)	처음 추가 Card를 화면에 표시
void last(Container parent)	마지막 추가된 Card를 화면에 표시
void next(Container parent)	다음 Card를 화면에 표시
void previous(Container parent)	이전 Card를 화면에 표시
void show(Container parent, String name)	지정된 이름의 Card를 화면에 표시

레이아웃 활용하기

2 레이아웃 관리자 이해하기

〈5〉 CardLayout 클래스

```
Frame f = new Frame("CardLayout Test");

CardLayout card = new CardLayout();
f.setLayout(card);

Panel[] panel = new Panel[5];
for(int i=0; i<panel.length; i++) {
    panel[i] = new Panel();
    panel[i].add(new Button("Card"+(i+1)));
    f.add(panel[i], "Card"+(i+1));
}

card.first(f);
card.last(f);
card.next(f);
card.previous(f);
card.show(f, "Card3");
```

2 레이아웃 관리자 이해하기

〈5〉 CardLayout 클래스

```
Frame f = new Frame("CardLayout Test");

CardLayout card = new CardLayout();
f.setLayout(card);

Panel[] panel = new Panel[5];
for(int i=0; i<panel.length; i++) {
    panel[i] = new Panel();
    panel[i].add(new Button("Card"+(i+1)));
    f.add(panel[i], "Card"+(i+1));
}

card.first(f);
card.last(f);
card.next(f);
card.previous(f);
card.show(f, "Card3");
```

CardLayout 객체를 생성후 Frame에 레이아웃 설정

레이아웃 활용하기

2 레이아웃 관리자 이해하기

5 CardLayout 클래스

```
Frame f = new Frame("CardLayout Test");

CardLayout card = new CardLayout();
f.setLayout(card);

Panel[] panel = new Panel[5];
for(int i=0; i<panel.length; i++) {
    panel[i] = new Panel();
    panel[i].add(new Button("Card"+(i+1)));
    f.add(panel[i], "Card"+(i+1));
}

card.first(f);
card.last(f);
card.next(f);
card.previous(f);
card.show(f, "Card3");
```

Panel 배열 선언(5개)

2 레이아웃 관리자 이해하기

5 CardLayout 클래스

```
Frame f = new Frame("CardLayout Test");

CardLayout card = new CardLayout();
f.setLayout(card);

Panel[] panel = new Panel[5];
for(int i=0; i<panel.length; i++) {
    panel[i] = new Panel();
    panel[i].add(new Button("Card"+(i+1)));
    f.add(panel[i], "Card"+(i+1));
}

card.first(f);
card.last(f);
card.next(f);
card.previous(f);
card.show(f, "Card3");
```

Panel 객체 생성

레이아웃 활용하기

2 레이아웃 관리자 이해하기

〈5〉 CardLayout 클래스

```
Frame f = new Frame("CardLayout Test");

CardLayout card = new CardLayout();
f.setLayout(card);

Panel[] panel = new Panel[5];
for(int i=0; i<panel.length; i++) {
    panel[i] = new Panel();
    panel[i].add(new Button("Card"+(i+1)));
    f.add(panel[i], "Card"+(i+1));
}

card.first(f);
card.last(f);
card.next(f);
card.previous(f);
card.show(f, "Card3");
```

Panel 객체에 Button 객체 추가

2 레이아웃 관리자 이해하기

〈5〉 CardLayout 클래스

```
Frame f = new Frame("CardLayout Test");

CardLayout card = new CardLayout();
f.setLayout(card);

Panel[] panel = new Panel[5];
for(int i=0; i<panel.length; i++) {
    panel[i] = new Panel();
    panel[i].add(new Button("Card"+(i+1)));
    f.add(panel[i], "Card"+(i+1));
}

card.first(f);
card.last(f);
card.next(f);
card.previous(f);
card.show(f, "Card3");
```

Frame에 Panel 객체 추가

레이아웃 활용하기

2 레이아웃 관리자 이해하기

⑤ CardLayout 클래스

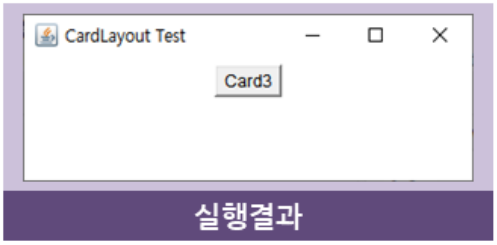
```
Frame f = new Frame("CardLayout Test");

CardLayout card = new CardLayout();
f.setLayout(card);

Panel[] panel = new Panel[5];
for(int i=0; i<panel.length; i++) {
    panel[i] = new Panel();
    panel[i].add(new Button("Card"+(i+1)));
    f.add(panel[i], "Card"+(i+1));
}

card.first(f);
card.last(f);
card.next(f);
card.previous(f);
card.show(f, "Card3");
```

CardLayout 객체를 이용하여 Card를 화면에 표시



실행결과

2 레이아웃 관리자 이해하기

⑥ Null 레이아웃

1. 레이아웃 관리자를 지정하지 않음
2. 레이아웃 관리자가 컴포넌트의 배치에 관여하지 않음
3. 컴포넌트의 xy 좌표와 폭, 넓이를 지정하여 배치함

레이아웃 활용하기

2 레이아웃 관리자 이해하기

⑥ Null 레이아웃

1. 레이아웃 관리자를 지정하지 않음
2. 레이아웃 관리자가 컴포넌트의 배치에 관여하지 않음
3. 컴포넌트의 xy 좌표와 폭, 넓이를 지정하여 배치함

```
Frame f = new Frame("Null Layout Test");
f.setLayout(null);

Button ok = new Button("OK");
ok.setBounds(75, 75, 150, 150);

f.add(ok);
```

Frame의 레이아웃 관리자를 null로 지정함

2 레이아웃 관리자 이해하기

⑥ Null 레이아웃

1. 레이아웃 관리자를 지정하지 않음
2. 레이아웃 관리자가 컴포넌트의 배치에 관여하지 않음
3. 컴포넌트의 xy 좌표와 폭, 넓이를 지정하여 배치함

```
Frame f = new Frame("Null Layout Test");
f.setLayout(null);

Button ok = new Button("OK");
ok.setBounds(75, 75, 150, 150);

f.add(ok);
```

Button 객체를 생성

레이아웃 활용하기

2 레이아웃 관리자 이해하기

⑥ Null 레이아웃

1. 레이아웃 관리자를 지정하지 않음
2. 레이아웃 관리자가 컴포넌트의 배치에 관여하지 않음
3. 컴포넌트의 xy 좌표와 폭, 넓이를 지정하여 배치함

```
Frame f = new Frame("Null Layout Test");
f.setLayout(null);

Button ok = new Button("OK");
ok.setBounds(75, 75, 150, 150);

f.add(ok);
```

Button 객체의 x, y 좌표, 폭, 넓이를 지정

2 레이아웃 관리자 이해하기

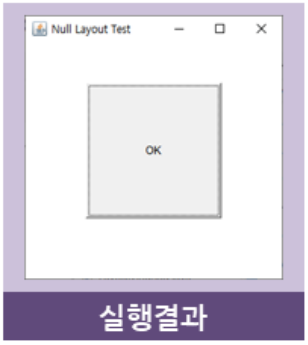
⑥ Null 레이아웃

1. 레이아웃 관리자를 지정하지 않음
2. 레이아웃 관리자가 컴포넌트의 배치에 관여하지 않음
3. 컴포넌트의 xy 좌표와 폭, 넓이를 지정하여 배치함

```
Frame f = new Frame("Null Layout Test");
f.setLayout(null);

Button ok = new Button("OK");
ok.setBounds(75, 75, 150, 150);

f.add(ok);
```



실행결과

레이아웃 활용하기

컨테이너와 레이아웃 활용하기 | 레이아웃 활용하기

실습하기

컨테이너와 레이아웃 활용하기 | 레이아웃 활용하기

실습순서

1. 레이아웃 관련 실습
   1) FlowLayout, BorderLayout, GridLayout, GridBagLayout, CardLayout, Null Layout

유의사항

- JDK와 이클립스를 설치한 후 실습이 가능함
- 본인이 원하는 작업 폴더를 미리 정해 놓은 다음 실습하기
- 작업 폴더는 C드라이브에 지정하기 보다는 D드라이브나 외장하드디스크를 활용하는 것을 추천함

※ 제공되는 실습 코드를 다운받아 실습해보시기 바랍니다.

응용문제

컨테이너와 레이아웃 활용하기 응용문제

다음 실행화면과 조건에 맞게 프로그램을 작성하시오.

조건

1. Frame 클래스를 상속하여 구현하시오.
2. Frame 전체 레이아웃은 BorderLayout으로 지정한다.
3. 북쪽은 FlowLayout으로 지정한다.
   : Label, TextField, Choice 컴포넌트("Male", "Female") 사용
4. 중앙은 GridLayout으로 지정한다.
   : Checkbox 컴포넌트와 CheckboxGroup 클래스 사용
5. 남쪽은 GridBagLayout으로 지정한다.
   : Button 컴포넌트 사용

클래스명 : LayoutTotal

제공되는 실습 소스코드를 다운받아 실습해보시기 바랍니다.

컨테이너와 레이아웃 활용하기 응용문제

실행화면