|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Simple Calendar App**  **리포트** | | | | | | 결  재 | 작 성 | | | 검 토 | | 승 인 |
|  | | |  | |  |
| 작성일자 | 2020.03.10 – 2020.4.13 | | | 제품명 | | | | | - | | | |
| 작성부서 | 기술연구소 | | | 작성자 | | | | | 이효은 연구원 | | | |
| 참조부서 | □ 관리부 | □ 생산부 | □ 품질부 | | ■ 연구부 | | | □ 영업부 | | | 🞎 기타 ( ) | |
| [ 내 용 정 의 ] |  | | | | | | | | | | | |

< 목적 >

* **Simple Calendar App 에 대한 산출물 보고서 정리**

< 내용 >

**1. Simple Calendar App에 대한 개념정의서**

1) 개요

- 달력이란 본디 1년 가운데 달, 일, 요일, 공휴일 등을 날짜에 따라 기록한 것으로 이를 간략한 앱(App)으로 구현하여 안드로이드 OS환경에서 구동시켜보고자 한다.

2) 개발 목표 : 간단한 달력 기능의 Simple Calendar App 제작

3) 개발 항목 :

- 월간 달력 출력 기능

- 이전, 다음 버튼 클릭 시 달력 이동 기능

4) 개발 환경 :

- window10 os, Android studio 4.0 Canary 9(ide 193.6137316(64bit))

5) 상세 개발 내용

5-1) 월간 달력 출력 기능 :

(1) 상단에 연도/ 월 출력, 중단에 요일, 일 출력

(2) 주말은 평일과 다른 컬러로 출력

(3) 오늘의 날짜는 다른 표시

~~(4) 공휴일 표시~~

~~(5) 음력 표시~~

5-2) 이전, 다음 버튼 클릭 시 달력 이동 기능 :

(1) 이전달, 다음달로 이동 가능한 버튼 달기

(2) 버튼 클릭 시 이전달, 다음달로 이동

6) 개발 일정 : (다음장)

6) **개발 일정 :**

- 자세한 개발 일정 및 실제 개발내용, 일정은 Angie\_WBS\_SimpleCalendar .xlsx에 WBS 서식에 맞춰 작성함.

- 개발 일정과 단계별 상세한 내용은 아래와 같다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1일자 | 2일자 | 3일자 | 4일자 | 5일자 | 6일자 | 7일자 | 8일자 | 9일자 | 10일자 | 11일자 | 12일자 | 13일자 | 14일자 |
| 1단계 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2단계 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3단계 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4단계 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5단계 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4,5-1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4,5-2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4,5-3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4,5-4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4,5-5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4,5-6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4,5-7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6단계 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7단계 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

\* Simple Calendar App에 대한 개발 일정 단계별 상세 내용

1 단계 : 개념정의서 작성

2 단계 : 기능정의서 작성

3 단계 : UI,UX 구조서 작성

4 단계 : 기능별 코드 작성

5 단계 : 기능에 대한 단위테스트

4,5단계 : 월간 달력 출력 기능

(1) 상단에 연도/ 월 출력, 중단에 요일, 일 출력

(2) 주말은 평일과 다른 컬러로 출력

(3) 오늘의 날짜는 다른 표시

~~(4) 공휴일 표시~~

~~(5) 음력 표시~~

이전, 다음 버튼 클릭 시 달력 이동 기능 :

(6) 이전달, 다음달로 이동 가능한 버튼 달기

(7) 버튼 클릭 시 이전달, 다음달로 이동

6 단계 : 통합테스트

7 단계 : 결과물 산출

**2. Simple Calendar App에 대한 기능 정의서**

1) Simple Calendar App에 대한 간략한 기능 정의는 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | Depth1 | Depth2 | Depth3 | 작업요소 | 기능정의 | 비고 |
| 1 | 월간달력출력 | 연도/월 출력 |  | 디자인/개발 | 상단에 연도, 월 출력 |  |
| 2 |  | 요일/일 출력 |  | 디자인/개발 | 중단에 요일, 일 출력 |  |
| 3 |  | 주말 표시 |  | 디자인/개발 | 주말은 평일과 구분되는 다른 컬러로 출력(예시: 일-빨강) |  |
| 4 |  | 오늘날짜 표시 |  | 디자인/개발 | 오늘의 날짜 위에 점을 찍는 등의 표시 |  |
| ~~5~~ |  | ~~공휴일 표시~~ |  | ~~디자인/개발~~ | ~~공휴일 날짜를 빨간색으로 표시~~ |  |
| ~~6~~ |  | ~~음력 표시~~ |  | ~~디자인/개발~~ | ~~음력 날짜를 날짜 아래에 표시~~ |  |
| 7 | 버튼 클릭시, 달력 이동 기능 | 이전달, 다음달로 이동 가능한 버튼 생성 | 생성한 버튼 클릭시, 각각 이전달, 다음달 월간달력 출력 | 디자인/개발 | 하단에 이전달, 다음달을 의미하는 버튼 생성 및 클릭시 각각 이전달, 다음달 월간 달력 출력 |  |
| 8 |  |  |  |  |  |  |

**3.Simple Calendar App에 대한 UI / UX 구조(1)**

- 위의 기능에 따라 Simple Calendar App에 대한 ui, ux구조는 아래 (1),(2)와 같다.

🡪 월간달력출력

1) 상단 – 연 / 월

2) 중상단 – 요일

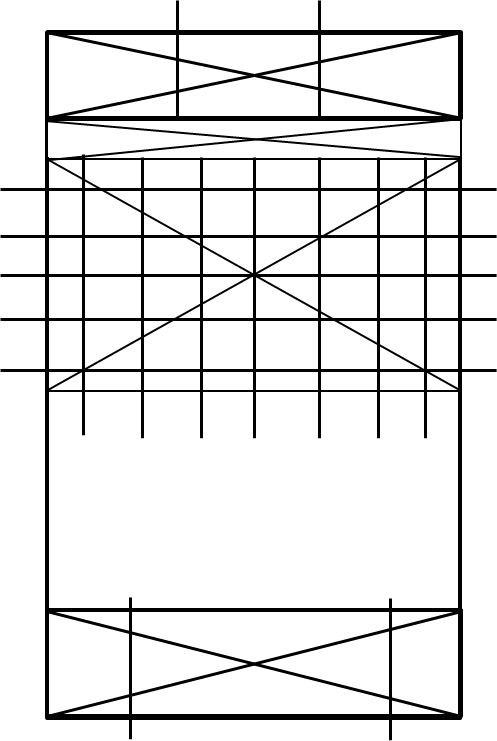
3) 중하단 - 월간 달력 날짜

(요일아래 표에 맞춰 날짜를 위치)

🡪 월간달력 이동버튼 위치

4) 하단: < 이전달, > 다음달

위의 대략적인 위치설계에 따른 화면 디자인

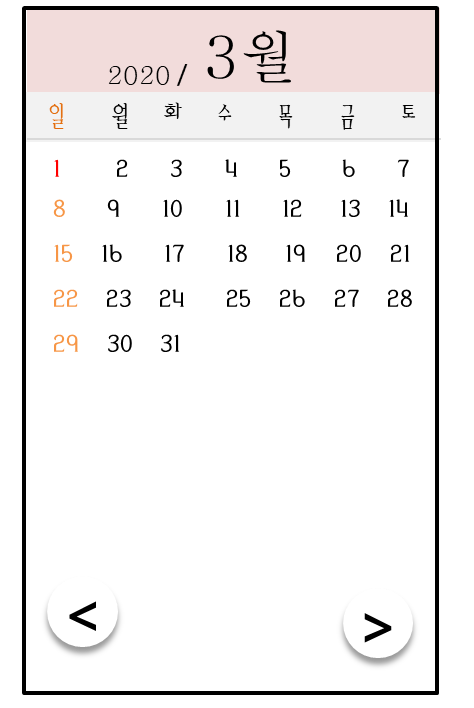


**4**

**3**

**2**

**1**



**3. Simple Calendar App에 대한 UI / UX 구조(2)**

**< 화면 설명 >**

1) 상단 (분홍배경)

: ( 2020 )년도 / ( 3 )월

2) 중단-상단 (회색배경)

: 요일 구성

3) 중단-하단 : 날짜 표시

- 오늘의 날짜표시

- 일요일 색깔표시,

~~- 공휴일 색깔표시~~

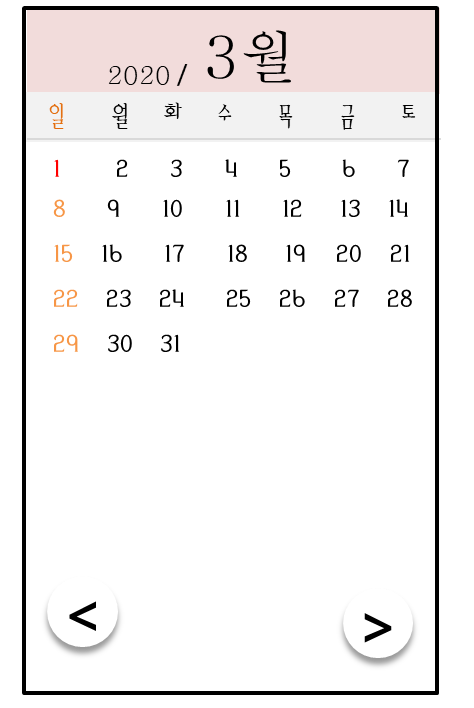
~~- 음력 표시~~

4) 하단

: 이동 버튼

(우측(이전달) / 좌측(다음달))

각각 버튼 클릭시 빨간 이정표를 따라 보이는 화면처럼 이전달, 다음달 화면으로 이동(화면은 1),2),3)과 구성이 동일)



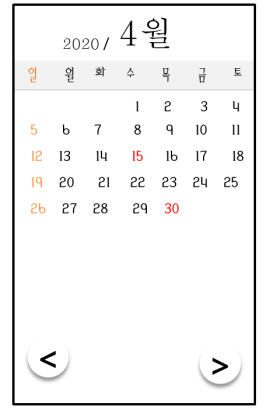
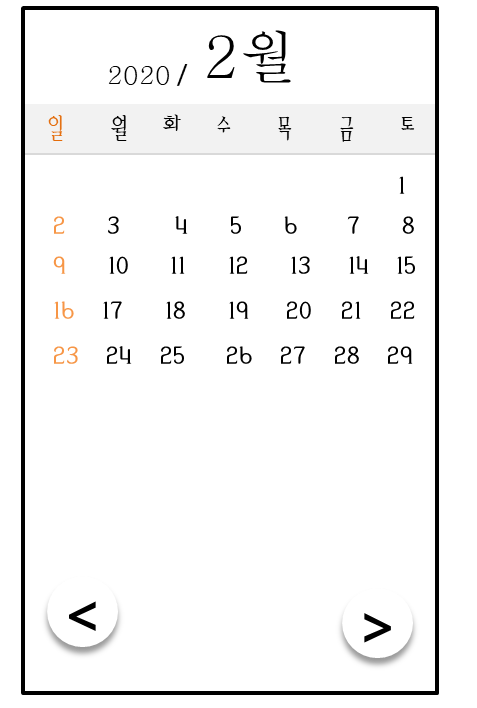
**3**

2.29

**21\**

**1**

**4**

****

**4. 실제로 구현한 Simple Calendar App**

**< 화면 설명 >**

1) 상단 (분홍배경)

: ( 2020 )년도 / ( 4 )월

2) 중단-상단 (회색배경)

: 요일 구성

3) 중단-하단 : 날짜 표시

- 오늘의 날짜표시

- 일요일 색깔표시,

~~- 공휴일 색깔표시~~

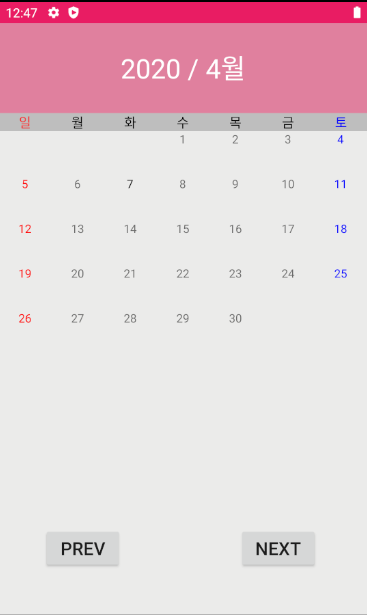
~~- 음력 표시~~

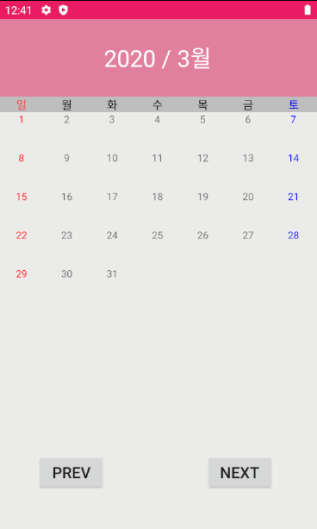
4) 하단

: 이동 버튼

(우측(이전달) / 좌측(다음달))

각각 버튼 클릭시 빨간 이정표를 따라 보이는 화면처럼 이전달, 다음달 화면으로 이동(화면은 1),2),3)과 구성이 동일)

****

** **

**5. Simple Calendar App에 대한 의사코드(pseudo code)**

1) Simple Calendar App에서 구현하고자 하는 기능은 아래와 같음.

(1) .xml 속성

- viewitem.xml 속성

: ViewText = Calendar의 내용을 채우는 TextView

- activity\_main.xml 속성

(1) tv\_date = 연 / 월 TextView

(2) LinearLayout-TextView(‘일’,’월’,’화’,’수’,’목’,’금’,’토’) = 주간 표시

(3) GridView-LinearLayout-Button-prevBtn = 이전버튼, -nextBtn = 다음 버튼

(4) btnClick = 버튼 클릭

(2) .java 속성

- MainActivity.java 속성(메서드)

(1) Oncreate

-LBtn, RBtn, tvDate

(2) btnClick

- prevBtn <

- nextBtn

(3) DateAdapter extends BaseAdapter

- getDatAdapter()

- Context()

- getCount()

- getItem()

(4) Data

- convertview

(3) .날짜 가져오기

: public class MainActivity extends Activity

- void onCreate(Bundle savedInstanceState) : LBtn, RBtn, tvDate

- void btnClick(View v

: class DateAdapter extends BaseAdapter = Calendar Gird에 data view적용

- Integer today mCal.get(Calendar.DAY\_OF\_MONTH);

- View getView(int position, View convertView, ViewGroup parent) : Calendar 뷰 메서드

- long getItemId(int position) : Calendar grid position 메서드

- Objects getItem(int position) : Calendar grid position에 data 매칭 메서드

- int getCount() : Calendar grid position에 data size count 메서드

(4) CalData.java

: public class CalData

- public CalData(int d, int h) : Calculate calendar data 메서드

- int getDay() : Day date data 메서드

- int getDayofWeek() : Day of the week data 메서드

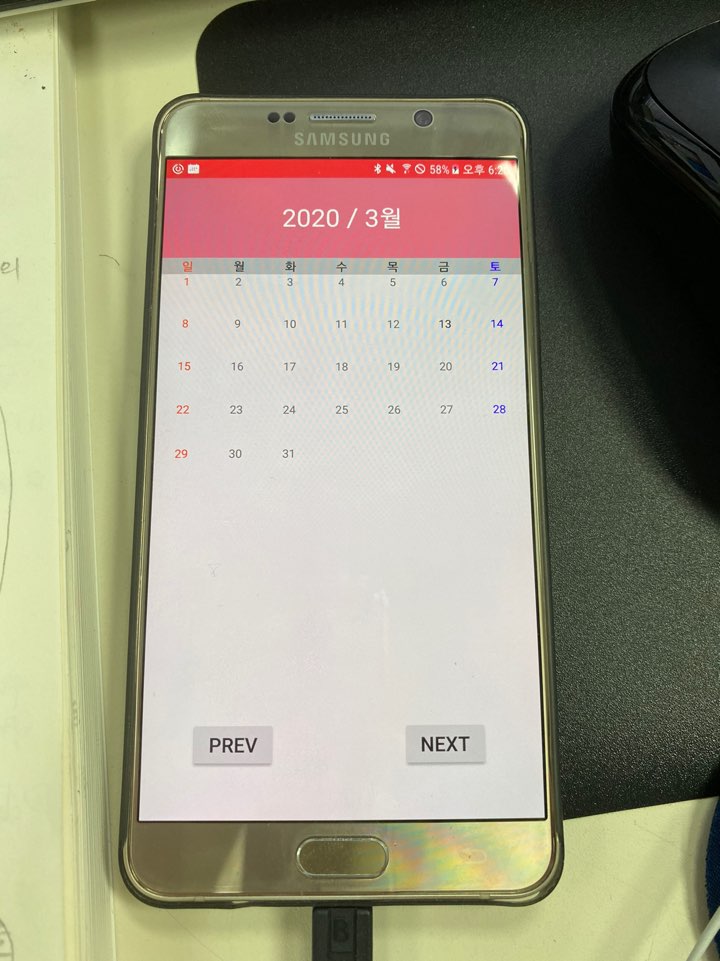
**(5) line by line 소스 리뷰**  
**package** com.example.myapplication;  
*/\*\*package (패키지이름)  
 \* 클래스와 인터페이스의 집합  
 \* 안드로이드상의 자바 패키지 이름(내부 소스코드 작성)이자 APP id(myapplication)(앱의 실별자 역할)  
 \* 패키지 이름과 APP id은 독립적. 서로 영향x, 변경o  
 \* https://jms0707.tistory.com/2\*/  
  
/\*\*import(키워드) 패키지이름.클래스이름  
 \* 각 패키지의 해당클래스(모듈)를 사용  
 \* http://cris.joongbu.ac.kr/course/java/api/allclasses-noframe.html  
 \* https://developer.android.com/reference/packages\*/  
//자바패키지***import** java.text.SimpleDateFormat; *//시간을 원하는 포맷으로 출력***import** java.util.ArrayList; *//List인터페이스의 사이즈 변경 가능한 배열 구현 및 리스트 포함을 위해 내부적으로 사용되는 배열의 사이즈 조작가능***import** java.util.Calendar; *//달력 필드간의 변환 및 날짜와 시간들의 취득 등 달력 필드 조작을 행하기 위함(추상클래스: 구체적이지 않아 상속을 강제하는 추상메소드가 포함된 클래스)***import** java.util.Date; *//연,월,일,시,분,초의 값을 받아 돌려주는 클래스***import** java.util.List; *//순서를 갖는 인터페이스, 리스트내 요소 위치,삽입 정밀 제어 가능  
  
//안드로이드패키지***import** android.app.Activity; *//UI를 배치할 수 있는 창생성***import** android.content.Context; *//장치에서 data를 액세스하고 게시하기 위한 클래스를 포함***import** android.graphics.Color; *//graphic 도구를 제공하여 화면에 색상생성, 변환 및 조작을 위한 메소드를 제공***import** android.os.Bundle; *//장치에서 기본 os서비스, 메시지전달 및 프로세스간 통신을 제공하며 Bundle클래스는 문자열 키에서 다양한 Parcelable값으로의 매핑(parcelable: 인스턴스를 작성하고 복원할 수 있는 클래스의 인터페이스)***import** android.view.LayoutInflater; *//XML에 정의된 resources을 view객체로 봔환***import** android.view.View; *//화면 레이아웃 및 사용자와의 상호 작용을 처리하는 기본 사용자 인터페이스 클래스를 제공하는 클래스를 제공***import** android.view.View.OnClickListener; *//button click event에 대한 인터페이스 구현***import** android.view.ViewGroup; *//n개의 View를 담는 클래스***import** android.widget.BaseAdapter; *//Adapter(ListView, SpinnerView에 출력할 data를 보관하는 장소) 기본클래스 구현***import** android.widget.Button; *//사용자가 탭하거나 클릭하여 작업을 수행 할 수 있는 사용자 인터페이스 요소***import** android.widget.GridView; *//격자보기. 격자의 항목은 ListAdapter와 연관됨***import** android.widget.TextView; *//사용자에게 텍스트를 표시하는 사용자 인터페이스 요소***import** com.example.myapplication.CalData; *//myapplication패키지의 CalData클래스. day, dayofweek 파라미터를 받아서 반환  
  
/\*\*  
 \* MainActivity 메인액티비티 (기본클래스, extends 문법으로 Activity 클래스를 상속)  
 \* public : 어떤 클래스에서도 접근, 사용 가능  
 \* protected : 이 클래스에서 상속받은 자식 클래스에서만 접근하여 사용 가능  
 \* private : 같은 클래스에서만 접근하여 사용가능  
 \* \*/***public class** MainActivity **extends** Activity {  
*/\*\* 필드(정보, 상태) \*/  
 //변수초기화* **private** TextView **tvDate**; *//텍스트뷰 연월* **private** GridView **mGridView**; *//그리드뷰* **private** DateAdapter **adapter**; *//그리드뷰날짜 어댑터* **private** ArrayList<CalData> **arrData**; *//일자 저장 리스트* **private** Calendar **mCalToday**; *//오늘날짜구하기 변수* **private** Calendar **mCal**; *//캘린더 변수* **private** Button **LBtn**; *//이전버튼* **private** Button **RBtn**; *//다음버튼* **private int thisYear**; *//이번년* **private int thisMonth**; *//이번달* **private int thisStartday**; *//이번년이번달시작요일정하기  
  
 /\*\* 메서드(동작, 기능) \*/  
 /\*\* OnCreate() 메서드 : Activity 클래스에서 기본적인 액티비티가 만들어지는 메서드  
 \** ***@Override로*** *이를 오버라이딩(상속, 하위클래스에서 클래스간 같은 이름의 메소드 재정의 가능)  
 \* 메서드 호출시 저장한 savedInstanceState Bundle객체(액티비티 중단시 임시적으로 데이터 저장 및 이전 데이터 호출)를 처리  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) { *//  
 //super.~ 상위클래스인 Activity클래스의 onCreate()메소드 처리  
 //savedInstancerState : 현재상태저장* **super**.onCreate(savedInstanceState);  
  
 *//Activity를 위한 레이아웃 UI를 설정* setContentView(R.layout.***activity\_main***);  
  
 *//Activity안의 멤버 변수를 초기화* **LBtn** = (Button) findViewById(R.id.***prevBtn***); *//prevBtn 버튼 = LBtn* **RBtn** = (Button) findViewById(R.id.***nextBtn***); *//nextBtn 버튼 = RBtn* **tvDate** = (TextView)findViewById(R.id.***tv\_date***); *//tv\_date 텍스트뷰 = tvDate  
  
 /\*\* getInstance() 싱글톤패턴적용 객체생성(객체얻어오기), 자체 생성자를 이용하여 객체생성 불가능  
 \* 싱글톤패턴: 해당클래스의 인스턴스가 하나만 만들어지고 어디서든지 그 인스턴스에 접근, 구현시 private생성자, 정적메소드, static변수를 사용  
 \* (다른 클래스에서 이 클래스의 인스턴스를 new를 이용해 생성하지 못하게하고 항상 getInstance()를 사용해서 인스턴스를 갖도록 제한)  
 \* Calendar 추상클래스는 객체를 생성해주는 메서드 getInstance()를 제공하므로 객체생성이 가능  
 \* cf)new 키워드를 통해 객체생성 : 클래스 인스턴스변수 = new 호출할메서드(파라미터)  
 \* \*/* **mCalToday** = Calendar.*getInstance*(); *//mCalToday 객체생성* **mCal** = Calendar.*getInstance*(); *//mCal 객체생성* **thisYear** = **mCal**.get(Calendar.***YEAR***); *//thisYear 객체생성, get()메서드로 mCal의 객체 호출* **thisMonth** = **mCal**.get(Calendar.***MONTH***)+1; *//thisMonth 객체생성, get()메서드로 mCal의 객체 호출(+1: Month value is 0-based. e.g., 0 for January.)* setCalendarDate(**thisMonth**); *//thisMonth 파라미터를 받아 setCalendarDate()함수 호출* }  
  
 */\*\* void btnClick(View v) : 버튼을 클릭했을 떄 실행되는 메서드(버튼의 클릭 이벤트 처리)  
 \* 이전달버튼 0월<->1월 및 다음달버튼 13월<->1월 처리  
 \* switch() ~case 문 : 여러가지 경우에 따라 어떤 작업을 할 것인지 결정  
 \* \*/* **public void** btnClick(View v){  
 **switch**(v.getId()){  
 **case** R.id.***prevBtn***: *//prevBtn 버튼 클릭시 0월, 13월 처리* **if**(**thisMonth**>1){  
 **thisMonth**--;  
 setCalendarDate(**thisMonth**);  
 }  
 **else**{  
 **thisMonth** = 0;  
 setCalendarDate(**thisMonth**);  
 }  
 **break**;  
  
 **case** R.id.***nextBtn***: *//nextBtn 버튼 클릭시 0월, 13월 처리* **if**(**thisMonth**<12){  
 **thisMonth**++;  
 setCalendarDate(**thisMonth**);  
 }  
 **else**{  
 **thisMonth** = 13;  
 setCalendarDate(**thisMonth**);  
 }  
 **break**;  
  
 **default**:  
 **break**;  
 }  
 }  
  
 */\*\* void setCalendarDate(int month) : 해당 월에 표시할 일 수를 구하는 메서드  
 \* if()~else if()~else 문 : 조건문  
 \* for(초기;조건;증감;){실행문} : 반복문  
 \* \*/* **public void** setCalendarDate(**int** month){  
 **arrData** = **new** ArrayList<CalData>(); *//변수 arr = 일자 저장 리스트* **if**(month == 0){ *//Month value is 0-based. e.g., 0 for January.* **thisMonth** = 12; *//이전해에 대한 연,월 처리* --**thisYear**;  
 **mCal**.set(**thisYear**,11,1);  
 }  
 **else if** (month == 13){ *//다음해에 대한 연,월 처리* **thisMonth** = 1;  
 ++**thisYear**;  
 **mCal**.set(**thisYear**,0,1);  
 }  
 **else**{  
 **mCal**.set(**thisYear**, month-1, 1); *//1일에 맞는 요일 설정* }  
  
 **thisStartday** = **mCal**.get(Calendar.***DAY\_OF\_WEEK***);  
  
 **if**(**thisStartday** != 1){ *//시작요일이 일요일이 아니면 공백으로 띄우기* **for**(**int** i=0; i<**thisStartday**-1; i++){  
 **arrData**.add(**null**);  
 }  
 }  
 **if**(month == 13){ *//1~12월까지 월처리(0~11)* month = 0;  
 }**else if**(month == 0){  
 month = 11;  
 }  
 **else**{  
 --month;  
 }  
  
 **for** (**int** i = 0; i < **mCal**.getActualMaximum(Calendar.***DAY\_OF\_MONTH***); i++) { *// 월간 최대일수 처리* **mCalToday**.set(**thisYear**, month, (i+1));  
 **arrData**.add(**new** CalData((i+1), **mCalToday**.get(Calendar.***DAY\_OF\_WEEK***)));  
 }  
  
 **adapter** = **new** DateAdapter(**this**, **arrData**); *//this, arrData를 DataAdapter로 보냄* **mGridView** = (GridView)findViewById(R.id.***calGrid***); *//calGrid(캘린더) 그리드뷰 = mGridView* **mGridView**.setAdapter(**adapter**); *//mGridView에 Data 셋팅* **tvDate**.setText(**thisYear** + **" / "** + **thisMonth** + **"월"** ); *// tvDate 텍스트뷰에 연 / "월" 셋팅* }  
}  
  
*/\*\*  
 \* DateAdapter 클래스 (extends 문법으로 BaseAdapter 클래스를 상속) : 그리드뷰 어댑터 적용  
 \* ListView는 사용자에게 표시할 데이터를 제공하기 위해서 Adapter를 생성해서 ListView에 연결  
 \*/***class** DateAdapter **extends** BaseAdapter {  
 **private** Context **context**; *//Context 변수(Context: 어플리케이션에 관하여 시스템이 관리하고 있는 정보에 접근, Context 인터페이스가 제공하는 API호출)* **private** LayoutInflater **inflater**; *//LayoutInflater 변수(LayoutInflater: XML에 미리 정의해둔 틀을 실제 메모리에 올려줌)* **private** ArrayList<CalData> **arrData**; *//CalData 배열 변수* **private** Calendar **mCal**; *//캘린더 변수  
  
 /\*\* DateAdapter 메서드 (param : Context c, ArrayList<CalData> arr)  
 \* this.~ : 현재 클래스의 인스턴스의 특정 필드를 지정하여 선언  
 \*/* **public** DateAdapter(Context c, ArrayList<CalData> arr) {  
 **this**.**context** = c; *//지금 메서드(DateAdapter)의 파라미터인 c값이 아닌 클래스의 필드인 context를 의미* **this**.**arrData** = arr; *//위와 같은 의미로 클래스의 필드인 arr를 의미* **inflater** = (LayoutInflater)**context**.getSystemService(Context.***LAYOUT\_INFLATER\_SERVICE***); *//inflater 변수 초기화. context api인 LAYOUT\_INFLATER\_SERVICE 호출)* }  
  
 */\*\* 접근자 메소드 : 멤버변수의 값을 리턴하고 설정하는 메소드  
 \* position : 리스트뷰 adapter에 연결된 AllayList의 특정 위치  
 \* getCount() : arrData의 개수를 리턴  
 \* getItem(position) : position의 Item을 리턴  
 \* getItemId(position) : Item의 id를 세팅하여 리턴  
 \* getView(int position, View convertView, ViewGroup parent) : Adapter가 가지고 있는 data를 converView로 리턴  
 \* converView :실제 화면에 그려지는 item을 converView라는 배열로 관리, convertView는 Adapter의 getView()를 통해서 관리됨  
 \* \*/* @Override  
 **public int** getCount() {  
 **return arrData**.size();  
 }  
  
 @Override  
 **public** Object getItem(**int** position) {  
 **return arrData**.get(position);  
 }  
  
 @Override  
 **public long** getItemId(**int** position) {  
 **return** position;  
 }  
  
 @Override  
 **public** View getView(**int** position, View convertView, ViewGroup parent) {  
 **if** (convertView == **null**) { *//파라미터에서 넘어온 converView가 아닌 매번 새로운 뷰를 inflate하여 메모리가 낭비되는것을 방지  
 //(안드로이드에서 inflate를 사용하면 xml에 정의되어 있는 view를 실제 view객체로 만드는 역할)* convertView = **inflater**.inflate(R.layout.***viewitem***, parent, **false**);  
 }  
  
 TextView ViewText = (TextView) convertView.findViewById(R.id.***ViewText***); *//convertView.ViewText 텍스트뷰 = ViewText* **if** (**arrData**.get(position) == **null**)  
 ViewText.setText(**""**);  
 **else** {  
 ViewText.setText(**arrData**.get(position).getDay() + **""**);  
  
 **if** (**arrData**.get(position).getDayofweek() == 1) *//일요일의 색깔 : 빨강* ViewText.setTextColor(Color.***RED***);  
  
 **else if**(**arrData**.get(position).getDayofweek() == 7) *//토요일의 색깔 : 파랑* ViewText.setTextColor((Color.***BLUE***));  
  
 **else** {  
 **mCal** = Calendar.*getInstance*();  
 Integer today = **mCal**.get(Calendar.***DAY\_OF\_MONTH***);  
  
 **if** (**arrData**.get(position).getDay() == today) *//오늘일자의 색깔 : 검정(평일색깔은 xml에서 회색으로 셋팅)* ViewText.setTextColor(Color.***BLACK***);  
 }  
 }  
 **return** convertView;  
 }  
}

**5. Simple Calendar App에 대한 단위 테스트\_보고서 생략**

**6. Simple Calendar App에 대한 통합 테스트\_보고서 생략**

**7. Simple Calendar App에 대한 결과 보고**

****



**끝.**