**Embedded System Software HW#4**

**Document**

**20131540**

**김준호**

**1. 목표**

**실습 시간 때 배운 내용을 활용하여 간단한 java application 프로그램을 작성한다.**

**2. 구현**

**(1) 보드에서 실행시킬 수 있는 application을 구현한다.**

(1)번의 구현 순서는 다음과 같았다.

1. MainActivity 구현

2. MainActivity2 구현

3. service를 이용한 timer 구현

먼저 MainActivity에 대한 구현이다.

MainActivity의 뷰 구성은 간단하다. puzzle 게임을 시작하는 button과 학번을 띄워주는 text view 2개이다. 이 2가지를 linear layout으로 생성하였고, button을 눌렸을 때 게임화면인 MainActivity2를 실행시키도록 만들었다.

...

Button btn=(Button)findViewById(R.id.button\_startPuzzle);

OnClickListener listener=new OnClickListener(){

public void onClick(View v){

Intent intent=new Intent(MainActivity.this, MainActivity2.class);

startActivity(intent);

}

};

...

위의 코드처럼 MainActivity에 있는 button의 onclick listener에 startActivity함수를 이용하여 MainActivity2를 실행시키는 코드를 추가했다.

다음으로 mainActivity2에 대한 구현이다. mainActivity2의 뷰 구성은 puzzle의 크기를 입력받는 editText와 진행 시간을 나타내는 textView, 각 버튼을 만들어내는 button과 이 버튼을 클릭했을 때 생성되는 button들이다. 우선 사용자가 editText에 puzzle의 행과 열을 입력한 후 make button을 클릭했을 때 수행하는 일에 대한 설명이다. make button의 onclick listener에선 맨 처음에 puzzle 버튼들이 생성되어 있었는지를 검사한다. 이는 linear layout의 getChildAt라는 함수를 통해 검사하는데 getChildAt(3)이 Null이 아닌지를 검사한다.

인자로 넘겨주는 3의 의미는 linear layout의 4번째 view를 나타내는 것으로 위에서 설명했듯 puzzle button들은 4번째 view로 만들어지게 된다. 따라서 getChildAt(3)이 Null이 아닌 경우는 사용자가 행과 열을 입력한 후에 make button을 누르고 게임 진행 도중 make button을 다시 누르는 경우이다. 이 때 clearButtons라는 함수를 호출하는데 이 함수에서 하는 일은 간단하게 row만큼 for loop을 돌면서 View를 지워주는 것이다. clearButtons 후에는 진행 시간을 나타내주는 textview를 00:00 으로 설정하고 timer service를 unbind해준다. timer service에 대한 설명은 구현의 (3)에서 설명하겠다. 다시 돌아가서 앞선 진행과정을 마치면 edit text에 있는 data를 getText 함수를 통해 가져오고 파싱해준 후에 integer로 변환하여 validation 검사를 해준다. 만약 valid한 값이라면 timer service를 bind해준 후에 새로운 thread를 생성하여 timer를 돌려준다. 이에 대한 자세한 설명도 구현 (3)에서 설명하겠다. 이후에 랜덤한 숫자 조합을 생성해주는데 이과정에 대해 설명하자면 먼저 0부터 len ( 행 X 열 )까지 수를 1부터 len - 1까지 수를 numList라는 integer 배열에 넣어 준 후 에 끝에 0을 넣어준다. (0이 blank 칸을 의미한다.) 다음으로 0부터 3까지의 난수를 생성하여 그에 맞는 방향에 위치 한 수와 swap해준다. 이 과정을 for loop을 통해 100번 해주는데 처음부터 난수를 입력하지 않는 이유는 풀 수 없는 puzzle이 생성되는 것을 방지하기 위함이다.

for(int i = 0; i < len - 1; i++) numList[i] = i + 1;

numList[len - 1] = 0;

int zr = (len - 1) / col, zc = (len - 1) % col;

for(int i = 0; i < 100; i++){

int dir = r.nextInt(4);

int tr = zr + dr[dir], tc = zc + dc[dir];

if(tr < 0 || tc < 0 || tr >=row || tc >= col) continue;

numList[zr \* col + zc] = numList[tr \* col + tc];

numList[tr \* col + tc] = 0;

zr = tr; zc = tc;

}

위는 난수 배열을 만들어 내는 부분의 코드이다.

다음으로는 생성된 난수 배열들이 이미 풀려버린 puzzle인지를 간단하게 검사해준 후에 refreshButtons라는 함수를 호출하여 puzzle 버튼들을 생성해주면 make button의 onclick listener 역할이 끝나게 된다. refreshButtons라는 함수에서 하는 일은 다음과 같다.

for(int i = 0; i < row; i++){

LinearLayout subLin = new LinearLayout(this);

subLin.setLayoutParams(new LayoutParams(width, height / row));

subLin.setTag("Buttons");

for(int j = 0; j < col; j++){

Button newBtn = new Button(this);

newBtn.setLayoutParams(new LayoutParams(width / col, height / row));

newBtn.setText( String.valueOf(numList[i \* col + j]) );

if(numList[i \* col + j] > 0)

newBtn.setOnClickListener(this);

else

newBtn.setBackgroundColor(Color.BLACK);

subLin.addView(newBtn);

}

lin.addView(subLin);

}

2중 loop를 이용하여 dynamic하게 button들을 view에 달아주는 함수인데 먼저 i loop에서는 row만큼 linear layout을 만들어주고 j loop에서 i loop에서 만든 linear layout에 버튼들을 달아준다. 이후 i loop 마지막엔 mainActivity2의 linear layout에 i loop에서 생성한 linear layout을 달아준다.

다음으로는 puzzle 버튼들이 눌렸을 때 수행되는 onClick 함수에 대한 설명이다. 먼저 눌린 버튼의 값을 getText라는 함수를 통해 가져와서 integer로 변환한다. 이후 눌린 값이 numList의 몇번째 index에 위치했는지 찾아주고 이후 그 idx가 puzzle의 몇번째 row, col에 위치해 있는지를 계산한다. 이후 puzzle의 (r,c)의 상하좌우에 0이 있는지, 즉 blank가 있는지를 검사하고 만약 있다면 numList에서 해당 index끼리 swap해준다. 이후 clearButtons를 호출해 버튼들을 지워주고 refreshButtons를 호출해 변경된 numList에 맞게 다시 puzzle 버튼들을 생성해준다. 이후 게임이 끝났는지를 검사해주는 checkFinish함수를 호출한다. checkFinish함수에서는 처음에 numList에 저장되있는 integer들이 1부터 순서대로 저장되어있는지 확인하고 만약 순서대로 저장되어 있다면 alertdialog를 통해 Complete라는 message를 뿌린다. 이후 alertdialog에 있는 ok 버튼을 클릭하면 mainActivity2를 종료시키고 timer service를 unbind 시켜준다.

**(2) 모든 버튼을 생성할 때 linear layout으로 구현하도록 한다.**

위에서 설명했듯 MainActivity와 MainActivity2 layout 모두 linear layout으로 구현했다. 이는 res / layout / 폴더의 XML 파일에서 확인 가능하다.

**(3) (추가구현) make button을 누르기 시작한 이후의 퍼즐 게임 플레이 시간을 나타내는 타이머를 Service로 구현한다.**

먼저 Manifest 파일에 다음과 같이 서비스를 추가했다.

<service

android:name="com.example.androidex.MyTimerService"

android:process=":remote"></service>

서비스 이름은 위와 같이 MyTimerService 이고 process 옵션을 remote로 줘서 remote service로 구현했다. 이 서비스를 통해 MainActivity2의 화면에서 행과 열을 입력받는 edit text 아래에 위치한 text view에 MM:SS 포맷으로 경과 시간을 띄운다.

먼저 프로젝트 패키지에 MyTimerService라는 java file를 생성한 후에 Service 객체를 상속받는다.

public class MyTimerService extends Service{

private static final String log = "log";

private boolean finish;

private int min = 0;

private int sec = 0;

log는 debug를 위한 string이고 finish는 게임이 종료되었는지를 확인하는 flag이다. min 과 sec은 각각 분과 초를 나타낸다. 먼저 서비스 객체가 생성되었을 때와 사라질때 호출되는 onCreate 함수와 onDestroy 함수를 다음과 같이 구현했다.

@Override

public void onCreate(){

super.onCreate();

//Using Thread

finish = false;

Thread timer = new Thread(new Timer());

timer.start();

}

@Override

public void onDestroy(){

super.onDestroy();

finish = true;

}

각각 service 객채의 onCreate함수와 onDestory 함수를 호출해주고 finish flag를 set해준다. 그리고 onCreate 함수에서 새로운 Timer라는 Thread를 생성해주는데 timer class 는 다음과 같다.

private class Timer implements Runnable{

@Override

public void run() {

// TODO Auto-generated method stub

//break if finish

while(!finish){

//sleep 1 sec

try{

Thread.sleep(1000);

}

catch(InterruptedException e){

e.printStackTrace();

}

sec++;

if(sec == 60){

min++;

sec = 0;

}

Log.v(log, "[SERVICE] min : " + Integer.toString(min) + " sec : " + Integer.toString(sec));

}

}

}

하는 일은 간단하게 finish가 아니면 loop를 돌면서 1초만큼 sleep된 후에 sec를 증가시키고 60초 후에는 min을 증가시켜 주는 것이다.

다음으로 sevice에서 진행되는 min과 sec를 MainActivity2로 보내주기 위해 필요한 AIDL 파일(일종의 interface파일이다.)이다. 파일의 내용은 다음과 같다.

package com.example.androidex;

interface IMyTimerService{

int getSec();

int getMin();

}

여기에 선언된 getSec과 getMin함수를 통해 sevice의 min과 sec를 가져오는 것이다.

이 함수들을 Sevice가 제공하는 binder 객체에 다음과 같이 bind해준다.

IMyTimerService.Stub binder = new IMyTimerService.Stub(){

@Override

public int getSec() throws RemoteException{

return sec;

}

public int getMin() throws RemoteException{

return min;

}

};

이후 onBind 메소드와 onUnbind메소드를 다음과 같이 구현한다.

@Override

public IBinder onBind(Intent intent) {

// TODO Auto-generated method stub

return binder;

}

onBind 메소드에선 binder를 return해주고 이 binder를 통해 MainActivity2가 getSec과 getMin함수를 사용하여 sevice의 min과 sec 값을 가져오는 것이다.

@Override

public boolean onUnbind(Intent intent){

finish = true;

return super.onUnbind(intent);

}

MyTimerService의 구현은 여기까지이고 이제 MainActivity2에서 service를 어떤식으로 사용하는지 설명하겠다. 우선 binder와 intent를 다음과 같이 선언하고 timer가 돌고있는지를 확인하는 flag도 다음과 같이 선언해준다.

TextView tvTimer;

private IMyTimerService binder;

private Intent intent;

private boolean timerRunning;

이후 ServiceConnection 객채를 다음과 같이 생성해준다.

private ServiceConnection connection = new ServiceConnection(){

@Override

public void onServiceConnected(ComponentName name, IBinder service) {

// TODO Auto-generated method stub

//get binder from service

binder = IMyTimerService.Stub.asInterface(service);

}

@Override

public void onServiceDisconnected(ComponentName name) {

// TODO Auto-generated method stub

}

};

이후 mainActivity2에서 make button이 눌렸을 때 호출되는 listener에서

intent = new Intent(mainActivity, MyTimerService.class);

bindService(intent, connection, BIND\_AUTO\_CREATE);

timerRunning = true;

new Thread(new GetTimerThread()).start();

다음과 같이 intent를 사용하여 service를 bind해주고 새로운 thread를 생성해 bind된 service를 사용하는 GetTimerThread를 실행시켜준다. bind 된 service는 puzzle이 끝나거나 사용자가 make button을 다시 눌렀을 때 unbind 된다. thread로 실행되는 GetTimerThread는 다음과 같다.

private class GetTimerThread implements Runnable{

private Handler handler = new Handler();

private int min = 0 , sec = 0;

@Override

public void run() {

// TODO Auto-generated method stub

while(timerRunning){

if(binder == null){

continue;

}

handler.post(new Runnable(){

@Override

public void run() {

try{

min = binder.getMin();

sec = binder.getSec();

tvTimer.setText(min / 10 + "" + min % 10 + ":" + sec / 10 + "" + sec % 10);

}

catch(RemoteException e){

e.printStackTrace();

}

}

});

try{

Thread.sleep(500);

}catch(InterruptedException e){

e.printStackTrace();

}

}

}

}

먼저 GetTimerThread가 실행되면 timerRunning가 true이면 loop를 돌면서 handler를 사용하여 (text view인 tvTimer를 갱신시켜주기 위해 handler를 사용했다.) handler안에서 binder를 이용해 service interface의 getMin과 getSec를 통해 min과 sec을 가져와서 tvTimer를 set해준다. 이후 0.5초동안 sleep을 해주는데 만약 sleep해 주지 않으면 생성한 Thread가 계속해서 돌면서 app이 정상적으로 작동하지 않았기 때문이다.

**3. 기능**

(1) App실행 시 초기화면

-> 퍼즐 게임 버튼 : 퍼즐게임 페이지로 넘어간다.

-> 현재 화면에서 TextView를 통해 자신의 학번을 확인할 수 있게 한다.

(2) 퍼즐 게임

-> 퍼즐 개수는 EditText를 통해 “row col” 형식으로 입력 받아, java file에서 dynamically 버튼을 만들어준다. 이 때 버튼의 숫자는 random하게 할당하고 버튼 중 한 개의 버튼은 빈 공간으로 다른 색의 버튼을 사용한다. 버튼의 개수는 최대 5×5까지로 한다. 빈 공간의 위, 아래, 왼쪽, 오른쪽에 있는 퍼즐 중 하나를 눌렀을 경우, 해당 퍼즐과 빈 공간의 위치가 바뀌게 된다. 이 과정을 반복하여 모든 숫자가 정렬되고 빈 공간이 우 하단에 위치하면 퍼즐 게임이 끝나고 초기 화면으로 돌아가게 된다.

**4. 기타**