|  |
| --- |
| **학번 : 201911204, 이름 : 이주형** |
| **2023년 05주차 실습 체크 리스트** |
| |  |  | | --- | --- | | **실습 1** | **완성** | | **실습 2** | **완성** | |
| **실습 1 : [ RecyclerView & dataAdapter 연동 & Click이벤트처리 ]** |
| **activity\_main.xml** |
| activity\_main.xml  <?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".MainActivity">   <TextView  android:layout\_width="match\_parent"  android:layout\_height="90dp"  android:text="영어 단어장"  android:textSize="30dp"  android:textStyle="bold"  android:gravity="center"  />   <androidx.recyclerview.widget.RecyclerView  android:id="@+id/recyclerView"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" />  </LinearLayout> |
| row.xml  <?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content">   <TextView  android:id="@+id/textView"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:background="@drawable/textview\_style"  android:padding="15dp"  android:layout\_margin="15dp"  android:text="TextView" />   <TextView  android:id="@+id/textMeaning"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_margin="15dp"  android:background="@drawable/textview\_style"  android:padding="15dp"  android:text="TextView" /> </LinearLayout> |
| textview\_style.xml  <?xml version="1.0" encoding="utf-8"?> <shape xmlns:android="http://schemas.android.com/apk/res/android">  <!--shape 객체 생성 -->   <!--stroke는 선 -->  <stroke  android:width="5dp"  android:color="#ff0"/>   <!--solid는 내부 면을 채우는 정보 -->  <solid  android:color="#0ff"/>   <!--corners는 사각형 끝을 둥글게 정보 -->  <corners  android:radius="1dp"  android:topRightRadius="100dp"  /> </shape> |
| **MainActivity.kt** |
| MainActivity.kt  package com.frost.lab01  import androidx.appcompat.app.AppCompatActivity import android.os.Bundle import android.util.Log import android.view.View import androidx.recyclerview.widget.LinearLayoutManager import com.frost.lab01.databinding.ActivityMainBinding import java.util.\* import kotlin.collections.ArrayList  class MainActivity : AppCompatActivity() {  lateinit var binding: ActivityMainBinding  val data:ArrayList<MyData> = ArrayList()  lateinit var adapter:MyDataAdapter  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  binding = ActivityMainBinding.inflate(*layoutInflater*)  setContentView(binding.*root*)  initData()  initRecyclerView()  }   private fun initData() {  val scan = Scanner(*resources*.openRawResource(R.raw.*words*))  //resources의 rawResource를 open해서 words.txt 진입  while(scan.hasNextLine()){  val word = scan.nextLine()  val meaning = scan.nextLine()  data.add(MyData(word, meaning, false))  }  scan.close()  }   private fun initRecyclerView() {  binding.recyclerView.*layoutManager* = LinearLayoutManager(this,  LinearLayoutManager.*VERTICAL*, false)  adapter = MyDataAdapter(data)//명시적으로 adapter 객체 생성 후 사용  adapter.itemClickListener = object :MyDataAdapter.OnItemClickListener{  //adapter 객체의 item 클릭시 리스너 초기화... 익명 객체로(object)  override fun OnItemClick(data: MyData, position: Int) {  //mainActivity에서 data를 받음  /\*  Toast.makeText(this@MainActivity, data.meaning, Toast.LENGTH\_SHORT).show()  //그냥 this는 안돼 그냥 this는 mainActivity가 아닌 adapter의 this를 의미  \*/  if(data.flag){  data.flag=false  }else{  data.flag=true  }  adapter.notifyItemChanged(position)  }  }  binding.recyclerView.*adapter* = adapter  //어댑터 연결  } } |
| MyDataAdapter.kt  package com.frost.lab01  import android.util.Log import android.view.LayoutInflater import android.view.View import android.view.ViewGroup import androidx.recyclerview.widget.RecyclerView import com.frost.lab01.databinding.RowBinding  class MyDataAdapter (val items:ArrayList<MyData>):RecyclerView.Adapter<MyDataAdapter.ViewHolder>() {  //어댑터 클래스 생성시 아이템을 받아야함  //어댑터 클래스는 RecyclerView의 어댑터를 상속받아서 만들어야 함  // 그 때 상속할 어댑터에는 ViewHolder클래스가 들어가야함   interface OnItemClickListener{  //클릭 리스너 인터페이스 선언 : 완전히 독립적인 클래스로 구현하기 위함  //메인에서 이용할 일종의 틀을 설계  fun OnItemClick(data:MyData, position: Int)  //클릭 리스너 사용시 수행할 이벤트  //매개변수에는 이 어댑터에서 연동 가능한 것을 넣기  }  var itemClickListener:OnItemClickListener?=null  //누군가가 이것을 구현할테니 일단 객체 선언  inner class ViewHolder(val binding: RowBinding):RecyclerView.ViewHolder(binding.root){  //viewHolder 클래스는 inner 클래스로 만들었고 들어오는 데이터는 binding객체가 들어옴  //해당 ViewHolder 클래스도 RecyclerView의 ViewHolder를 상속받아 생성   //위치는 view가 만들어졌을 때가 좋으니 ViewHolder 클래스 내부에 수행  //view의 아이템 클릭시 itemClickListener 이벤트 처리하기 위해서 여기서 호출 시점 결정  init {  binding.textView.setOnClickListener**{** itemClickListener?.OnItemClick(items[adapterPosition], adapterPosition)  //포지션은 자동으로 가져옴  //이 adatperView를 부착하고 있는 누군가가(MainActivity) 구현할 것  **}** }  }  override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {  //RowBinding 객체 만들어서 ViewHolder 생성  val view = RowBinding.inflate(LayoutInflater.from(parent.context),  parent, false)  return ViewHolder(view)  }   override fun getItemCount(): Int {  return items.size  }   override fun onBindViewHolder(holder: ViewHolder, position: Int) {  //items의 데이터 연결  holder.binding.textView.text = items[position].word  holder.binding.textMeaning.text = items[position].meaning  if(items[position].flag){  holder.binding.textMeaning.visibility= View.VISIBLE  }else{  holder.binding.textMeaning.visibility= View.GONE  }  } } |
| MyData.kt  package com.frost.lab01  data class MyData(var word:String, var meaning:String, var flag:Boolean) {} |
| **실행결과** |
|  |

|  |
| --- |
| **실습2 : [ RecyclerView & dataAdapter & swipe & DragAndDrop 구현 ]** |
| **activity\_main.xml** |
| activity\_main.xml  <?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".MainActivity">   <TextView  android:layout\_width="match\_parent"  android:layout\_height="90dp"  android:text="영어 단어장"  android:textSize="30dp"  android:textStyle="bold"  android:gravity="center"  />   <androidx.recyclerview.widget.RecyclerView  android:id="@+id/recyclerView"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" /> </LinearLayout> |
| row.xml  <?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content">   <TextView  android:id="@+id/textView"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:background="@drawable/textview\_style"  android:padding="15dp"  android:layout\_margin="15dp"  android:text="TextView" />  </LinearLayout> |
| textview\_style.xml  <?xml version="1.0" encoding="utf-8"?> <shape xmlns:android="http://schemas.android.com/apk/res/android">  <!--shape 객체 생성 -->   <!--stroke는 선 -->  <stroke  android:width="5dp"  android:color="#ff0"/>   <!--solid는 내부 면을 채우는 정보 -->  <solid  android:color="#0ff"/>   <!--corners는 사각형 끝을 둥글게 정보 -->  <corners  android:radius="1dp"  android:topRightRadius="100dp"  /> </shape> |
| AndroidManifest,xml -> TTS를 위한 <queries>설정  <?xml version="1.0" encoding="utf-8"?> <manifest xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools">  <!--TTS\_SERVICE 사용 설정-->  <queries>  <intent>  <action android:name="android.intent.action.TTS\_SERVICE"/>  </intent>  </queries>  <application  android:allowBackup="true"  android:dataExtractionRules="@xml/data\_extraction\_rules"  android:fullBackupContent="@xml/backup\_rules"  android:icon="@mipmap/ic\_launcher"  android:label="@string/app\_name"  android:supportsRtl="true"  android:theme="@style/Theme.Lab02"  tools:targetApi="31">  <activity  android:name=".MainActivity"  android:exported="true">  <intent-filter>  <action android:name="android.intent.action.MAIN" />  <category android:name="android.intent.category.LAUNCHER" />  </intent-filter>  </activity>  </application> </manifest> |
| **MainActivity.kt** |
| MainActivity.kt  package com.frost.lab02  import androidx.appcompat.app.AppCompatActivity import android.os.Bundle import android.speech.tts.TextToSpeech import android.widget.Toast import androidx.recyclerview.widget.ItemTouchHelper import androidx.recyclerview.widget.LinearLayoutManager import androidx.recyclerview.widget.RecyclerView import com.frost.lab02.databinding.ActivityMainBinding import java.util.\* import kotlin.collections.ArrayList  class MainActivity : AppCompatActivity() {  lateinit var binding: ActivityMainBinding  val data:ArrayList<MyData> = ArrayList()  lateinit var adapter: MyDataAdapter  lateinit var tts: TextToSpeech  var isTTSReady = false   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  binding = ActivityMainBinding.inflate(*layoutInflater*)  setContentView(binding.*root*)   initData()  initRecyclerView()  initTTS()  }   private fun initTTS() {  tts = TextToSpeech(this)**{** //함수가 두번째 인자면 밖으로 따로 빼서 사용 가능  isTTSReady = true //TTS 준비 되었다는 true  tts.*language* = Locale.*US*//영어로 언어 설정  //아이템 클릭시 콜백함수로서 읽어줄 것!  **}** }   private fun initData() {  val scan = Scanner(*resources*.openRawResource(R.raw.*words*))  //resources의 rawResource를 open해서 words.txt 진입  while(scan.hasNextLine()){  val word = scan.nextLine()  val meaning = scan.nextLine()  data.add(MyData(word, meaning))  }  scan.close()  }   private fun initRecyclerView() {  binding.recyclerView.*layoutManager* = LinearLayoutManager(this,  LinearLayoutManager.*VERTICAL*, false)  adapter = MyDataAdapter(data)//명시적으로 adapter 객체 생성 후 사용  adapter.itemClickListener = object : MyDataAdapter.OnItemClickListener{  //adapter 객체의 item 클릭시 리스너 초기화... 익명 객체로(object)  override fun OnItemClick(data: MyData, position: Int) {  //mainActivity에서 data를 받음  if(isTTSReady){  tts.speak(data.word, TextToSpeech.*QUEUE\_ADD*, null, null)  //클릭 할 때마다 word를 Queue에 추가  //클릭할 때 TextToSpeech 콜백함수로 활용  }  Toast.makeText(this@MainActivity, data.meaning, Toast.*LENGTH\_SHORT*).show()  //그냥 this는 안돼 그냥 this는 mainActivity가 아닌 adapter의 this를 의미  }  }  binding.recyclerView.*adapter* = adapter  //어댑터 연결   //1. 콜백 익명 객체 생성  val simpleCallback = object:ItemTouchHelper.SimpleCallback(  ItemTouchHelper.*UP* or ItemTouchHelper.*DOWN*, ItemTouchHelper.*RIGHT* ){  override fun onMove(//Drag&Drop 시 호출됨  recyclerView: RecyclerView,//기능 recyclerView 설정  viewHolder: RecyclerView.ViewHolder,//Drag 할 viewHolder  target: RecyclerView.ViewHolder//Drag 될 viewHolder  ): Boolean {  adapter.moveItem(viewHolder.*adapterPosition*, target.*adapterPosition*)  return true  }   override fun onSwiped(viewHolder: RecyclerView.ViewHolder, direction: Int) {  //Swipe 시 호출됨  //recyclerView의 data는 Adapter 클래스에서 관리 중이라   // Adapter 클래스에서 삭제해주면서 layout 갱신  adapter.removeItem(viewHolder.*adapterPosition*)  //해당 위치의 data 삭제되었음을 알리면 어댑터에서 알아서 삭제하고 갱신해줌  }  }  val itemTouchHelper = ItemTouchHelper(simpleCallback)  //해당 기능을 갖는 아이템 터치 helper 객체 생성  itemTouchHelper.attachToRecyclerView(binding.recyclerView)  //recyclerView에 attach하기  }   override fun onStop() {  //앱이 멈춰있는 상태일 때 TTS 서비스도 멈추기  super.onStop()  tts.stop()  }   override fun onDestroy() {  //완전히 앱이 종료되었을 때 TTS 서비스도 shutdown해주기 for 안전한 사용  super.onDestroy()  tts.shutdown()  } } |
| MyDataAdapter.kt  package com.frost.lab02  import android.view.LayoutInflater import android.view.ViewGroup import androidx.recyclerview.widget.RecyclerView import com.frost.lab02.databinding.RowBinding  class MyDataAdapter (val items:ArrayList<MyData>):RecyclerView.Adapter<MyDataAdapter.ViewHolder>() {  //어댑터 클래스 생성시 아이템을 받아야함  //어댑터 클래스는 RecyclerView의 어댑터를 상속받아서 만들어야 함  // 그 때 상속할 어댑터에는 ViewHolder클래스가 들어가야함   interface OnItemClickListener{  //클릭 리스너 인터페이스 선언 : 완전히 독립적인 클래스로 구현하기 위함  //메인에서 이용할 일종의 틀을 설계  fun OnItemClick(data:MyData, position: Int)  //클릭 리스너 사용시 수행할 이벤트  //매개변수에는 이 어댑터에서 연동 가능한 것을 넣기  }  var itemClickListener:OnItemClickListener?=null  //누군가가 이것을 구현할테니 일단 객체 선언  inner class ViewHolder(val binding: RowBinding):RecyclerView.ViewHolder(binding.root){  //viewHolder 클래스는 inner 클래스로 만들었고 들어오는 데이터는 binding객체가 들어옴  //해당 ViewHolder 클래스도 RecyclerView의 ViewHolder를 상속받아 생성   //위치는 view가 만들어졌을 때가 좋으니 ViewHolder 클래스 내부에 수행  //view의 아이템 클릭시 itemClickListener 이벤트 처리하기 위해서 여기서 호출 시점 결정  init {  binding.textView.setOnClickListener**{** itemClickListener?.OnItemClick(items[adapterPosition], adapterPosition)  //포지션은 자동으로 가져옴  //이 adatperView를 부착하고 있는 누군가가(MainActivity) 구현할 것  **}** }  }  fun removeItem(pos:Int){  items.removeAt(pos)  //items(ArrayList)의 해당 pos의 데이터 삭제  notifyItemRemoved(pos)  //지우고 다시 그리라는 함수  }  fun moveItem(oldPos:Int, newPos:Int){  //이동되었음을 알리라는 함수  val temp = items[oldPos]  items.removeAt(oldPos)  items.add(newPos, temp)  notifyItemMoved(oldPos, newPos)  }  override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {  //RowBinding 객체 만들어서 ViewHolder 생성  val view = RowBinding.inflate(LayoutInflater.from(parent.context),  parent, false)  return ViewHolder(view)  }   override fun getItemCount(): Int {  return items.size  }   override fun onBindViewHolder(holder: ViewHolder, position: Int) {  //items의 데이터 연결  holder.binding.textView.text = items[position].word  } } |
| MyData.kt  package com.frost.lab02  data class MyData(var word:String, var meaning:String) {} |
| **실행결과** |
|  |