|  |
| --- |
| **학번 : 201911204, 이름 : 이주형** |
| **2023년 06주차 실습 체크 리스트** |
| |  |  | | --- | --- | | **실습 1** | **완성** | | **실습 2** | **완성** | |
| **실습 1 : [ implicit Intent와 Permission ]** |
| **activity\_main.xml** |
| <?xml version="1.0" encoding="utf-8"?> <androidx.constraintlayout.widget.ConstraintLayout 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:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".MainActivity">   <Button  android:id="@+id/btnCall"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="전화걸기"  app:layout\_constraintBottom\_toTopOf="@+id/btnMsg"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent" />   <Button  android:id="@+id/btnMsg"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="문자보내기"  app:layout\_constraintBottom\_toTopOf="@+id/btnWeb"  app:layout\_constraintStart\_toStartOf="@+id/btnCall"  app:layout\_constraintTop\_toBottomOf="@+id/btnCall" />   <Button  android:id="@+id/btnWeb"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="네이버"  app:layout\_constraintBottom\_toTopOf="@+id/btnMap"  app:layout\_constraintStart\_toStartOf="@+id/btnMap"  app:layout\_constraintTop\_toBottomOf="@+id/btnMsg" />   <Button  android:id="@+id/btnMap"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="맵"  app:layout\_constraintBottom\_toTopOf="@+id/btnCamera"  app:layout\_constraintStart\_toStartOf="@+id/btnCamera"  app:layout\_constraintTop\_toBottomOf="@+id/btnWeb" />   <Button  android:id="@+id/btnCamera"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="카메라"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/btnMap" /> </androidx.constraintlayout.widget.ConstraintLayout> |
| **AndroidManifest.xml** |
| <?xml version="1.0" encoding="utf-8"?> <manifest xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools">  <!--Call\_PHONE과 CAMERA 관련 permission 등록-->  <uses-permission android:name="android.permission.CALL\_PHONE"/>  <uses-permission android:name="android.permission.CAMERA"/>  <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.Lab01"  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** |
| package com.frost.lab01  import android.app.AlertDialog import android.content.Intent import android.content.pm.PackageManager import android.net.Uri import androidx.appcompat.app.AppCompatActivity import android.os.Bundle import android.provider.MediaStore import android.widget.Toast import androidx.activity.result.contract.ActivityResultContracts import androidx.core.app.ActivityCompat import com.frost.lab01.databinding.ActivityMainBinding  class MainActivity : AppCompatActivity() {  lateinit var binding: ActivityMainBinding  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  binding=ActivityMainBinding.inflate(*layoutInflater*)  setContentView(binding.*root*)  initLayout()  checkPermissions()  }  val permissions = *arrayOf*(android.Manifest.permission.*CALL\_PHONE*,  android.Manifest.permission.*CAMERA*)  //permissions 배열 설정   val multiplePermissionsLauncher =  registerForActivityResult(ActivityResultContracts.RequestMultiplePermissions())**{**map **->** //요청하고자 하는 복수의 권한 callback 함수를 매개변수로 등록 : ActivityResultContracts.RequestMultiplePermissions()  //multiple이라서 매개변수가 map 타입으로 data가 들어옴  val resultPermission = map.*all***{** entry **->** //여기서 매개변수는 Map의 Entry 정보 한 개  entry.value  //entry의 모든 value가 true이면 true를 반환  //하나라도 false이면 false를 반환  //즉, 들어온 권한들은 map entry 하나 하나씩인데 그 map entry의 value가 모두 true이어야 함.  **}** if(!resultPermission){  //하나라도 권한이 false이면 종료  //finish()    Toast.makeText(this, "모든 권한 승인됨",Toast.*LENGTH\_SHORT*).show()  }  **}** private fun initLayout() {  binding.*apply***{** btnCall.setOnClickListener **{** callAction()  **}** btnMsg.setOnClickListener **{** val msg = Uri.parse("sms:010-1234-1234")  val msgIntent = Intent(Intent.*ACTION\_SENDTO*, msg)  //메세지 보낼 수 있는 앱 실행  msgIntent.putExtra("sms\_body", "빨리 다음꺼 하자....")  //해당 메세지 칸에 해당 내용 들어감  startActivity(msgIntent)  **}** btnWeb.setOnClickListener **{** val web = Uri.parse("http://www.naver.com")  val webIntent = Intent(Intent.*ACTION\_VIEW*, web)  //해당 프로토콜로 들어갈 수 있는 앱 실행  startActivity(webIntent)  **}** btnMap.setOnClickListener **{** val location = Uri.parse("geo:37.543684, 127.077130?z=16")  val mapIntent = Intent(Intent.*ACTION\_CALL*, location)  //지도를 보여줄 수 있는 앱 실행  startActivity(mapIntent)  **}** btnCamera.setOnClickListener **{** cameraAction()  **}  }** }  private fun checkPermissions(){  //모든 권한 허용되었는지 체크  when{  (ActivityCompat.checkSelfPermission(this, //CALL\_PHONE permission 허락된 경우  android.Manifest.permission.*CALL\_PHONE*)  == PackageManager.*PERMISSION\_GRANTED*)  &&  (ActivityCompat.checkSelfPermission(this, //CAMERA permission 허락된 경우  android.Manifest.permission.*CAMERA*)  //checkSelfPermission() 현재 권한 정보 허용되었는지 체크하는 함수  == PackageManager.*PERMISSION\_GRANTED*)-> {  Toast.makeText(this, "모든 권한 승인됨",Toast.*LENGTH\_SHORT*).show()  }   (ActivityCompat.shouldShowRequestPermissionRationale(this,  //명시적으로 CALL\_PHONE permission 거부된 경우  android.Manifest.permission.*CALL\_PHONE*))  ||  (ActivityCompat.shouldShowRequestPermissionRationale(this,  //명시적으로 CAMERA permission 거부된 경우  android.Manifest.permission.*CAMERA*))  ->{  //거부된 경우 내용 작성  permissionCheckAlertDlg()  }   else ->{//이 부분은 아직 권한 설정에 해당하는 상황이 허용이나 거부되지 않은 상황일 때 즉, 처음 실행시  //따라서 권한 허용을 요청해야 함  //permissionLauncher.launch(android.Manifest.permission.CALL\_PHONE)  //permissionLauncher 객체를 통해서 권한 승인 여부를 받음  //launch()로 해당 콜백함수 실행 매개변수로는 요청하는 permission 정보를 등록    multiplePermissionsLauncher.launch(permissions)  //모두 다 허용되는 경우를 요구  }  }  }   private fun permissionCheckAlertDlg() {  val builder = AlertDialog.Builder(this)  //AlertDialog의 Builder 객체 설정  builder//builder 객체 속성 설정  .setMessage("반드시 CALL\_PHONE 권한과 CAMERA 권한이 모두 허용되어야 합니다.")//다이얼로그 내용 설정  .setTitle("권한 체크")//다이얼로그 제목 설정  .setPositiveButton("OK")**{** //OK 버튼 누를 시  \_, \_ **->**//인자는 사용하지 않을거라서 \_로 처리  multiplePermissionsLauncher.launch(permissions)  //이번에는 multiplePermission을 구동하는 것이라 매개변수에 해당 권한의 배열 설정  //다시 요청하도록 수행  **}**.setNegativeButton("Cancel")**{** //Cancel 버튼 누를 시  dlg, \_ **->**//dlg 인자는 dialog 관련 정보, 뒤는 사용하지 않을 인자  dlg.dismiss()  //finish()는 앱이 종료가 되는 것이고  //dialog는 dismiss로 종료  **}** //chain으로 정보 설정 가능  val dlg = builder.create()  //builder 객체 자체는 정보만 있는 것이고 create해야 생성됨  dlg.show()  //show()를 통해서 화면에 보여짐  }  fun allPermissionGranted()=permissions.*all***{** //it은 permissions 배열의 원소 각각 하나이고  //all을 통해서 모든 배열의 원소 체크  ActivityCompat.checkSelfPermission(this, **it**)==PackageManager.*PERMISSION\_GRANTED* **}** fun callphonePermissionGranted()=ActivityCompat.checkSelfPermission(this, //CALL\_PHONE permission 허락된 경우  android.Manifest.permission.*CALL\_PHONE*) == PackageManager.*PERMISSION\_GRANTED* fun cameraPermissionGranted()=ActivityCompat.checkSelfPermission(this, //CALL\_PHONE permission 허락된 경우  android.Manifest.permission.*CAMERA*) == PackageManager.*PERMISSION\_GRANTED* //camera와 callphone 각각의 권한이 체크가 되었는지 확인하는 함수 생성   val permissionLauncher = registerForActivityResult(ActivityResultContracts.RequestPermission())**{** //요청하고자 하는 단일 권한 callback함수를 매개변수로 등록 : ActivityResultContracts.RequestPermission()  //이후 해당 body 부분에 수행할 내용 작성  if(**it**){//it로 권한 승인 여부를 boolean 타입으로 받음  callAction()  }else{  Toast.makeText(this, "권한 승인이 거부되었습니다.",Toast.*LENGTH\_SHORT*).show()  }  **}** private fun callAlertDlg()  {  val builder = AlertDialog.Builder(this)  //AlertDialog의 Builder 객체 설정  builder//builder 객체 속성 설정  .setMessage("반드시 CALL\_PHONE 권한이 허용되어야 합니다.")//다이얼로그 내용 설정  .setTitle("권한 체크")//다이얼로그 제목 설정  .setPositiveButton("OK")**{** //OK 버튼 누를 시  \_, \_ **->**//인자는 사용하지 않을거라서 \_로 처리  permissionLauncher.launch(android.Manifest.permission.*CALL\_PHONE*)  //다시 요청하도록 수행  **}**.setNegativeButton("Cancel")**{** //Cancel 버튼 누를 시  dlg, \_ **->**//dlg 인자는 dialog 관련 정보, 뒤는 사용하지 않을 인자  dlg.dismiss()  //finish()는 앱이 종료가 되는 것이고  //dialog는 dismiss로 종료  **}** //chain으로 정보 설정 가능  val dlg = builder.create()  //builder 객체 자체는 정보만 있는 것이고 create해야 생성됨  dlg.show()  //show()를 통해서 화면에 보여짐  }   private fun cameraAction() {  val intent = Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*)  //카메라 사용 가능한 앱 실행  //if(cameraPermissionGranted()){//해당 기능 권한 설정 허락 여부 체크  if(allPermissionGranted()){//모든 권한 수행 허락시  startActivity(intent)  }else{  checkPermissions()  }  }   private fun callAction() {  //해당 앱은 권한이 다 승인되어야 수행되기 때문에 승인이 안되면 실행이 안되기에 별도의 작업 필요 없음  val number = Uri.parse("tel:010-1234-1234")  val callIntent = Intent(Intent.*ACTION\_CALL*, number)  //전화를 걸 수 있는 앱 실행  //val callIntent = Intent(Intent.ACTION\_DIAL, number)  //전화 걸기 전에 다이얼 앱 실행  //묵시적인 Intent는 무슨 앱인지는 모르지만 해당 action은 설정을 하고  //실행 가능한 앱을 알아서 찾아줄 것  //하지만 전화 거는 것은 돈이 들어가기에 권한 설정을 필수적으로 해야함  if(callphonePermissionGranted()){//권한 설정 허락 여부 체크  startActivity(callIntent)  }else{  checkPermissions()  }  }  /\*  private fun callAction() {  val number = Uri.parse("tel:010-1234-1234")  val callIntent = Intent(Intent.ACTION\_CALL, number)  //전화를 걸 수 있는 앱 실행  //val callIntent = Intent(Intent.ACTION\_DIAL, number)  //전화 걸기 전에 다이얼 앱 실행  //묵시적인 Intent는 무슨 앱인지는 모르지만 해당 action은 설정을 하고  //실행 가능한 앱을 알아서 찾아줄 것  startActivity(callIntent)  //하지만 전화 거는 것은 돈이 들어가기에 권한 설정을 필수적으로 해야함  when{  ActivityCompat.checkSelfPermission(this, //permission 허락된 경우  android.Manifest.permission.CALL\_PHONE)  //checkSelfPermission() 현재 권한 정보 허용되었는지 체크하는 함수  == PackageManager.PERMISSION\_GRANTED -> {  startActivity(callIntent)  }  ActivityCompat.shouldShowRequestPermissionRationale(this,  //명시적으로 permission 거부된 경우  //참고로 두 번이나 명시적으로 거부한 경우 더이상 창이 뜨지 않고 앱 속성을 통해서 권한 설정해야함  android.Manifest.permission.CALL\_PHONE)  ->{  //거부된 경우 내용 작성  callAlertDlg()  }  else ->{//이 부분은 아직 권한 설정에 해당하는 상황이 허용이나 거부되지 않은 상황일 때 즉, 처음 실행시  //따라서 권한 허용을 요청해야 함  permissionLauncher.launch(android.Manifest.permission.CALL\_PHONE)  //permissionLauncher 객체를 통해서 권한 승인 여부를 받음  //launch()로 해당 콜백함수 실행 매개변수로는 요청하는 permission 정보를 등록  }  }  }  \*/ } |
| **실행결과** |
|  |

|  |
| --- |
| **실습2 : [ explicit Intent ]** |
| **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> |
| **activity\_intro.xml** |
| <?xml version="1.0" encoding="utf-8"?> <androidx.constraintlayout.widget.ConstraintLayout 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:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".IntroActivity">   <Button  android:id="@+id/voclistbtn"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginTop="179dp"  android:text="영어단어장"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent" />   <Button  android:id="@+id/addvocbtn"  android:layout\_width="0dp"  android:layout\_height="wrap\_content"  android:layout\_marginTop="30dp"  android:text="단어추가"  app:layout\_constraintEnd\_toEndOf="@+id/voclistbtn"  app:layout\_constraintStart\_toStartOf="@+id/voclistbtn"  app:layout\_constraintTop\_toBottomOf="@+id/voclistbtn" /> </androidx.constraintlayout.widget.ConstraintLayout> |
| **activity\_add\_voc.xml** |
| <?xml version="1.0" encoding="utf-8"?> <androidx.constraintlayout.widget.ConstraintLayout 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:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".AddVocActivity">  <TextView  android:id="@+id/textView2"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginStart="4dp"  android:layout\_marginEnd="19dp"  android:text="단어"  app:layout\_constraintBaseline\_toBaselineOf="@+id/word"  app:layout\_constraintEnd\_toStartOf="@+id/word"  app:layout\_constraintHorizontal\_chainStyle="packed"  app:layout\_constraintStart\_toStartOf="parent" />   <TextView  android:id="@+id/textView3"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginBottom="10dp"  android:text="뜻"  app:layout\_constraintBottom\_toBottomOf="@+id/meaning"  app:layout\_constraintStart\_toStartOf="@+id/textView2" />   <EditText  android:id="@+id/word"  android:layout\_width="wrap\_content"  android:layout\_height="50dp"  android:layout\_marginTop="102dp"  android:ems="10"  android:inputType="textPersonName"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toEndOf="@+id/textView2"  app:layout\_constraintTop\_toTopOf="parent" />   <EditText  android:id="@+id/meaning"  android:layout\_width="wrap\_content"  android:layout\_height="50dp"  android:layout\_marginTop="16dp"  android:ems="10"  android:inputType="textPersonName"  app:layout\_constraintStart\_toStartOf="@+id/word"  app:layout\_constraintTop\_toBottomOf="@+id/word" />   <Button  android:id="@+id/addbtn"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginStart="104dp"  android:text="추가"  app:layout\_constraintBaseline\_toBaselineOf="@+id/cancelbtn"  app:layout\_constraintStart\_toStartOf="parent" />   <Button  android:id="@+id/cancelbtn"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginTop="58dp"  android:layout\_marginEnd="26dp"  android:text="취소"  app:layout\_constraintEnd\_toEndOf="@+id/meaning"  app:layout\_constraintTop\_toBottomOf="@+id/meaning" /> </androidx.constraintlayout.widget.ConstraintLayout> |
| **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> |
| **AndroidManifest.xml** |
| <?xml version="1.0" encoding="utf-8"?> <manifest xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools">  <!--Activity정보 수정시 확인해야함-->  <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.Lab02explicitIntent"  tools:targetApi="31">  <activity  android:name=".AddVocActivity"  android:exported="false" />  <activity  android:name=".IntroActivity"  android:exported="true">  <intent-filter>  <action android:name="android.intent.action.MAIN" />   <category android:name="android.intent.category.LAUNCHER" />  </intent-filter>  </activity>  <activity  android:name=".MainActivity"  android:exported="true">  </activity>  </application> </manifest> |
| **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="#000"/>  <!--corners는 사각형 끝을 둥글게 정보 -->  <corners  android:radius="100dp"  /> </shape> |
| **MainActivity.kt** |
| package com.frost.lab02explicitintent  import androidx.appcompat.app.AppCompatActivity import android.os.Bundle import android.widget.Toast import androidx.recyclerview.widget.LinearLayoutManager import com.frost.lab02explicitintent.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() {  try {  //내가 단어를 추가해 파일에서 읽어오기  val scan2 = Scanner(openFileInput("voc.txt"))  readScanFile(scan2)  scan2.close()  }catch(ex:Exception){  }  val scan = Scanner(*resources*.openRawResource(R.raw.*words*))  //resources의 rawResource를 open해서 words.txt 진입  readScanFile(scan)  scan.close()   }   private fun readScanFile(scan: Scanner) {  //scan로 객체로 파일을 읽어서 data에 추가하는 메서드  while(scan.hasNextLine()){  val word = scan.nextLine()  val meaning = scan.nextLine()  data.add(MyData(word, meaning))  }  }   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) {  //mainActivity에서 data를 받음  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  } } |
| **IntroActivity.kt** |
| package com.frost.lab02explicitintent  import android.app.Activity import android.app.Instrumentation.ActivityResult import android.content.Intent import androidx.appcompat.app.AppCompatActivity import android.os.Bundle import android.widget.Toast import androidx.activity.result.contract.ActivityResultContracts import com.frost.lab02explicitintent.databinding.ActivityIntroBinding import com.frost.lab02explicitintent.databinding.ActivityMainBinding  class IntroActivity : AppCompatActivity() {  lateinit var binding: ActivityIntroBinding  val launcher = //인텐트로 상호작용할 launcher 객체 생성  registerForActivityResult(ActivityResultContracts.StartActivityForResult()){  //콜백 함수 등록  if(it.resultCode == Activity.RESULT\_OK){  val voc = it.data?.getSerializableExtra("voc") as MyData  //인텐트 정보를 반환  Toast.makeText(this, voc.word + "추가됨", Toast.LENGTH\_SHORT).show()  }  }  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  binding = ActivityIntroBinding.inflate(layoutInflater)  setContentView(binding.root)   initLayout()  }   private fun initLayout() {  binding.voclistbtn.setOnClickListener {  val i = Intent(this, MainActivity::class.java)  startActivity(i)  }  binding.addvocbtn.setOnClickListener {  val i = Intent(this, AddVocActivity::class.java)  launcher.launch(i)  //인텐트를 담아서 만든 런처 객체 launch 하기  //그러면 콜백함수로 startActivityForResult 함수가 호출  //이렇게 전용함수로 만든 것 실행  }  } } |
| **AddVocActivity.kt** |
| package com.frost.lab02explicitintent  import android.app.Activity import android.content.Context import android.content.Intent import androidx.appcompat.app.AppCompatActivity import android.os.Bundle import com.frost.lab02explicitintent.databinding.ActivityAddVocBinding import java.io.PrintStream  class AddVocActivity : AppCompatActivity() {  lateinit var binding:ActivityAddVocBinding  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  binding = ActivityAddVocBinding.inflate(layoutInflater)  setContentView(binding.root)   initLayout()  }   private fun initLayout() {  binding.addbtn.setOnClickListener {  val word = binding.word.text.toString()  val meaning = binding.meaning.text.toString()   val output = PrintStream(openFileOutput("voc.txt", Context.MODE\_APPEND))  //파일 출력을 위한 output 객체 open해서 append모드로 추가  output.println(word)  output.println(meaning)  //해당 파일에 출력하기  output.close()  //파일 닫기   val i = Intent()  i.putExtra("voc", MyData(word, meaning))  //인텐트 생성해서 데이터 담기  setResult(Activity.RESULT\_OK, i)  //OK 라는 result값 넘겨서 해당 콜백함수 실행하고 데이터 담은 인텐트 정보 전달  finish()  }  binding.cancelbtn.setOnClickListener {  setResult(Activity.RESULT\_CANCELED)  //캔슬되었다는 result값 넘기고  finish()  //종료  }  } } |
| **MyDataAdapter.kt** |
| package com.frost.lab02explicitintent  import android.app.Activity import android.content.Context import android.content.Intent import androidx.appcompat.app.AppCompatActivity import android.os.Bundle import com.frost.lab02explicitintent.databinding.ActivityAddVocBinding import java.io.PrintStream  class AddVocActivity : AppCompatActivity() {  lateinit var binding:ActivityAddVocBinding  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  binding = ActivityAddVocBinding.inflate(*layoutInflater*)  setContentView(binding.*root*)   initLayout()  }   private fun initLayout() {  binding.addbtn.setOnClickListener **{** val word = binding.word.*text*.toString()  val meaning = binding.meaning.*text*.toString()   val output = PrintStream(openFileOutput("voc.txt", Context.*MODE\_APPEND*))  //파일 출력을 위한 output 객체 open해서 append모드로 추가  output.println(word)  output.println(meaning)  //해당 파일에 출력하기  output.close()  //파일 닫기   val i = Intent()  i.putExtra("voc", MyData(word, meaning))  //인텐트 생성해서 데이터 담기  setResult(Activity.*RESULT\_OK*, i)  //OK 라는 result값 넘겨서 해당 콜백함수 실행하고 데이터 담은 인텐트 정보 전달  finish()  **}** binding.cancelbtn.setOnClickListener **{** setResult(Activity.*RESULT\_CANCELED*)  //캔슬되었다는 result값 넘기고  finish()  //종료  **}** } } |
| **MyData.kt** |
| package com.frost.lab02explicitintent  data class MyData(var word:String, var meaning:String):java.io.Serializable {} |
| **실행결과** |
|  |