startActivity(Intent)

literally, starting new activity

```
class FromActivity{
  fun onClickButton(){
    startActivity(Intent(this, ToActivity::class.java))
  }
}
```

pseudo code

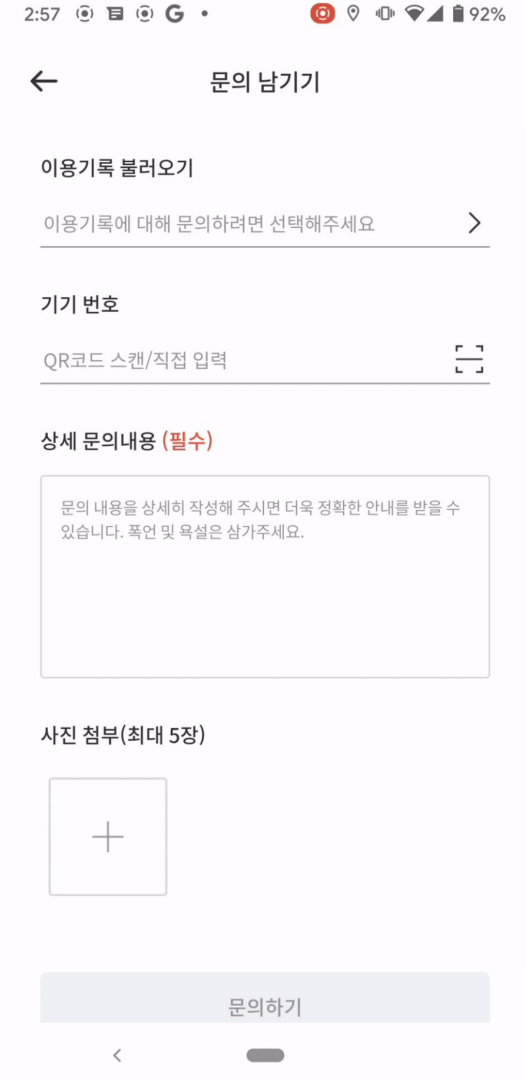
startActivity

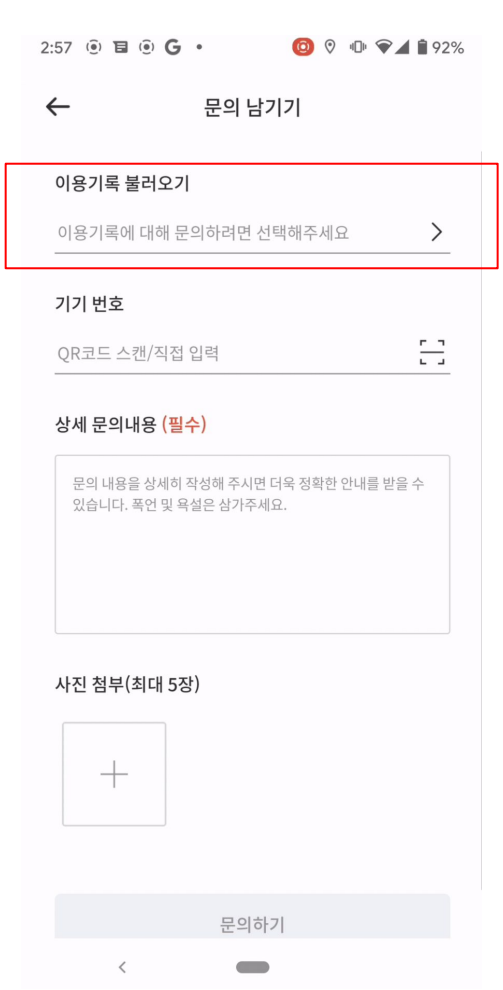
```
public void startActivity (Intent intent)
```

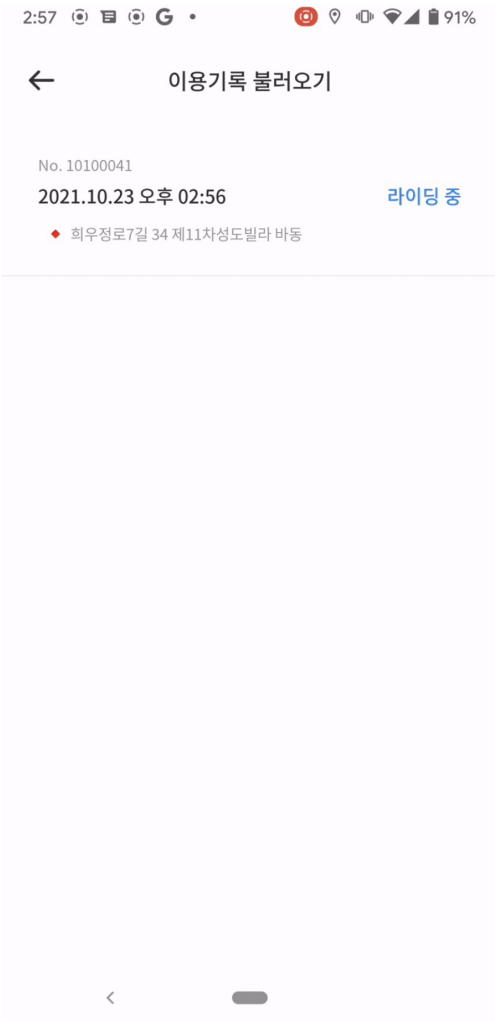
Same as `startActivity(android.content.Intent, android.os.Bundle)` with no options specified.

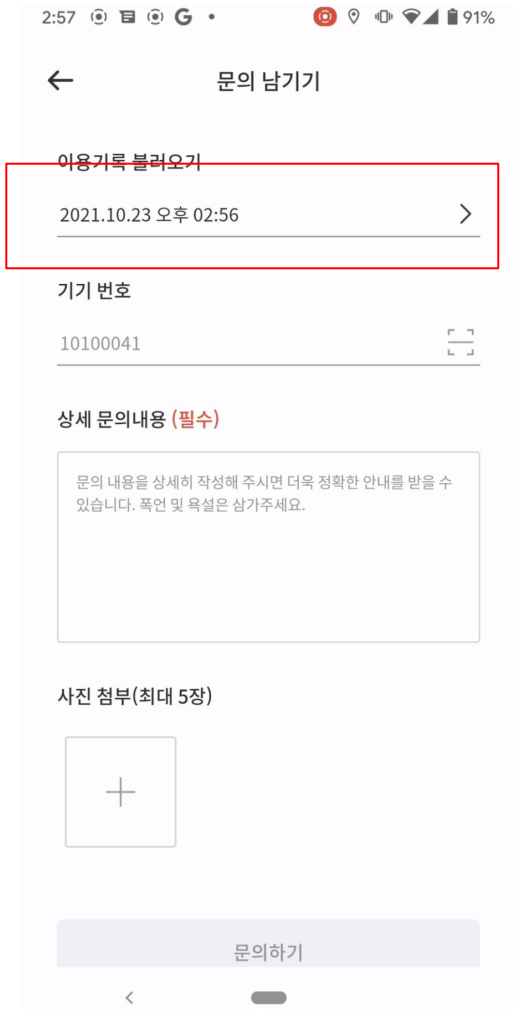
Parameters	
`intent`	`Intent`: The intent to start.


2:57
92%
고객센터
자주 묻는 질문
서비스 이용
자전거 이용방법
이용 중 문제 발생
결제
계정
보험/사고 접수
기타
아직 해결되지 않으셨나요?
전화상담시간
평일(월-금) 09:00 - 18:00
(점심시간 12:00 - 13:00), 주말 및 공휴일 휴무
고객센터
문의 남기기


2:57
92%
문의 남기기
이용기록 불러오기
이용기록에 대해 문의하려면 선택해주세요
기기 번호
QR코드 스캔/직접 입력
상세 문의내용 (필수)
문의 내용을 상세히 작성해 주시면 더욱 정확한 안내를 받을 수 있습니다. 폭언 및 욕설은 삼가주세요.
사진 첨부(최대 5장)
문의하기

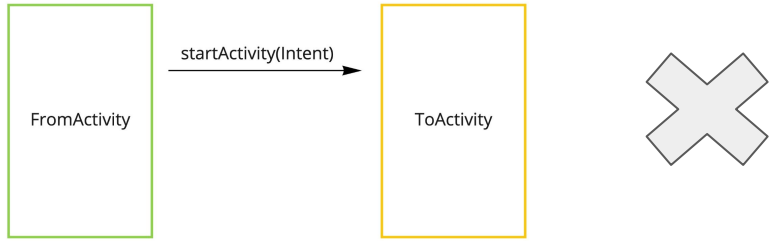

2:57
92%
문의 남기기
이용기록 불러오기
이용기록에 대해 문의하려면 선택해주세요
기기 번호
QR코드 스캔/직접 입력
상세 문의내용 (필수)
문의 내용을 상세히 작성해 주시면 더욱 정확한 안내를 받을 수 있습니다. 폭언 및 욕설은 삼가주세요.
사진 첨부(최대 5장)
문의하기

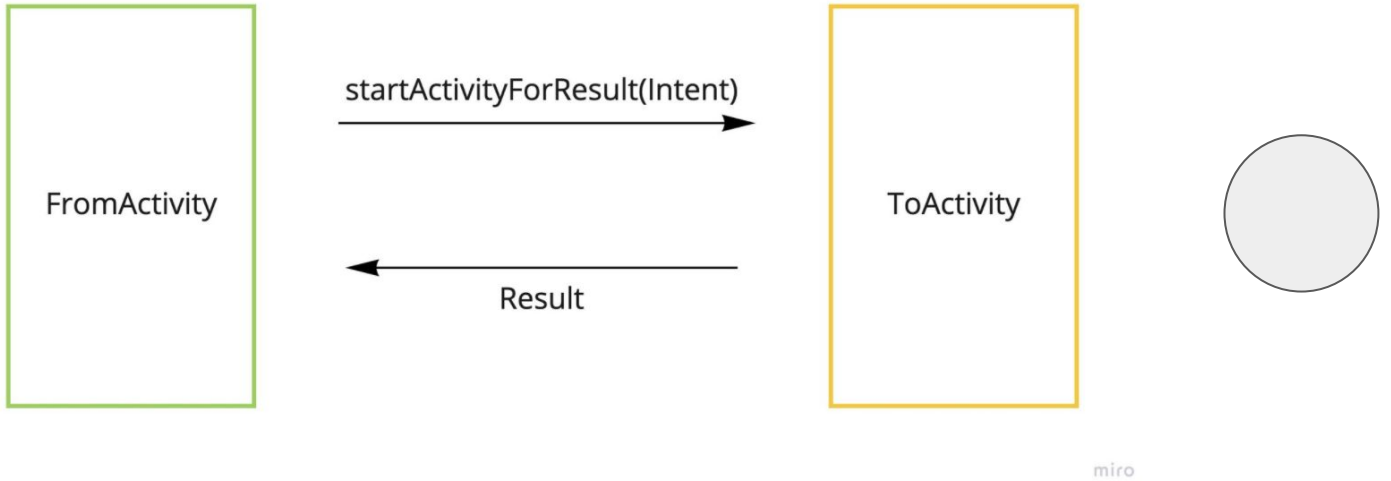

2:57
91%
이용기록 불러오기
No. 10100041
2021.10.23 오후 02:56
라이딩 중
희우정로7길 34 제11차성도빌라 바동


2:57
91%
문의 남기기
이용기록 불러오기
2021.10.23 오후 02:56
기기 번호
10100041
상세 문의내용 (필수)
문의 내용을 상세히 작성해 주시면 더욱 정확한 안내를 받을 수 있습니다. 폭언 및 욕설은 삼가주세요.
사진 첨부(최대 5장)
문의하기

Getting a result from an activity

startActivityForResult [Deprecated😕]





pseudocode

```
class FromActivity{
  companion object{
    const val REQUEST_CODE = 123
    const val RESULT_CODE = 456
  }

  fun startToActivity(){
    // [0] "From" activity에서 "To" activity로 전달해줄 데이터 지정
    val intent = Intent(this, ToActivity::class.java).putExtra("KEY", "VALUE")
    // [1] "From" activity에서 "To" activity 실행
    startActivityForResult(intent, REQUEST_CODE)
  }

  // [4] "FromActivity"의 onActivityResult 실행
  override fun onActivityResult(requestCode: Int, re
t?){
    super.onActivityResult(requestCode, resultCode,
    if(requestCode == REQUEST_CODE){
      if(resultCode == RESULT_CODE){
        // [5] REQUEST_CODE, RESULT_CODE를 통해
        // "From" -> "To" -> "From"로 되돌아온 케이스임을
        textView.text = data.getStringExtra("RESULT_
      }
    }
  }
}
```

"From" activity

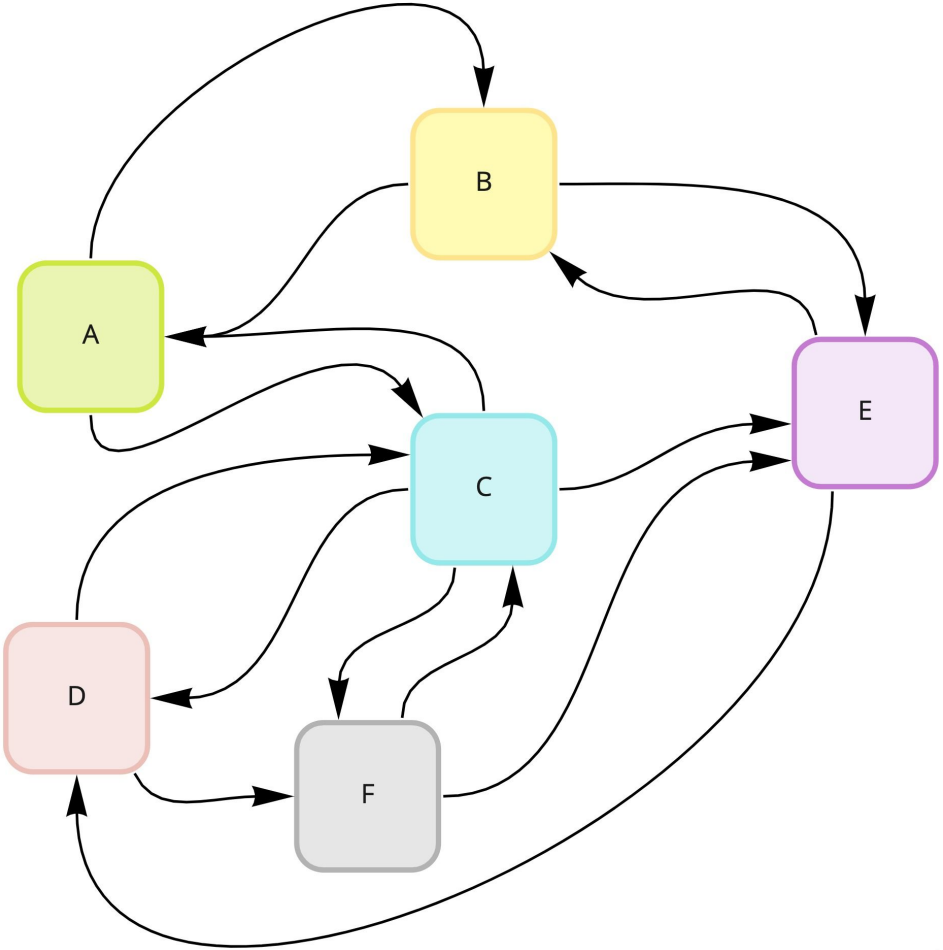
```
class ToActivity{
  fun finishToActivity(){
    // [2] "To" activity에서 "From" activity로 전달해줄 데이터 지정
    val resultIntent = Intent().putExtra("RESULT_KEY", "RESULT_VALUE")
    setResult(RESULT_CODE, resultIntent)
    // [3] "To" activity 종료
    finish()
  }
}
```

"To" activity

pseudocode

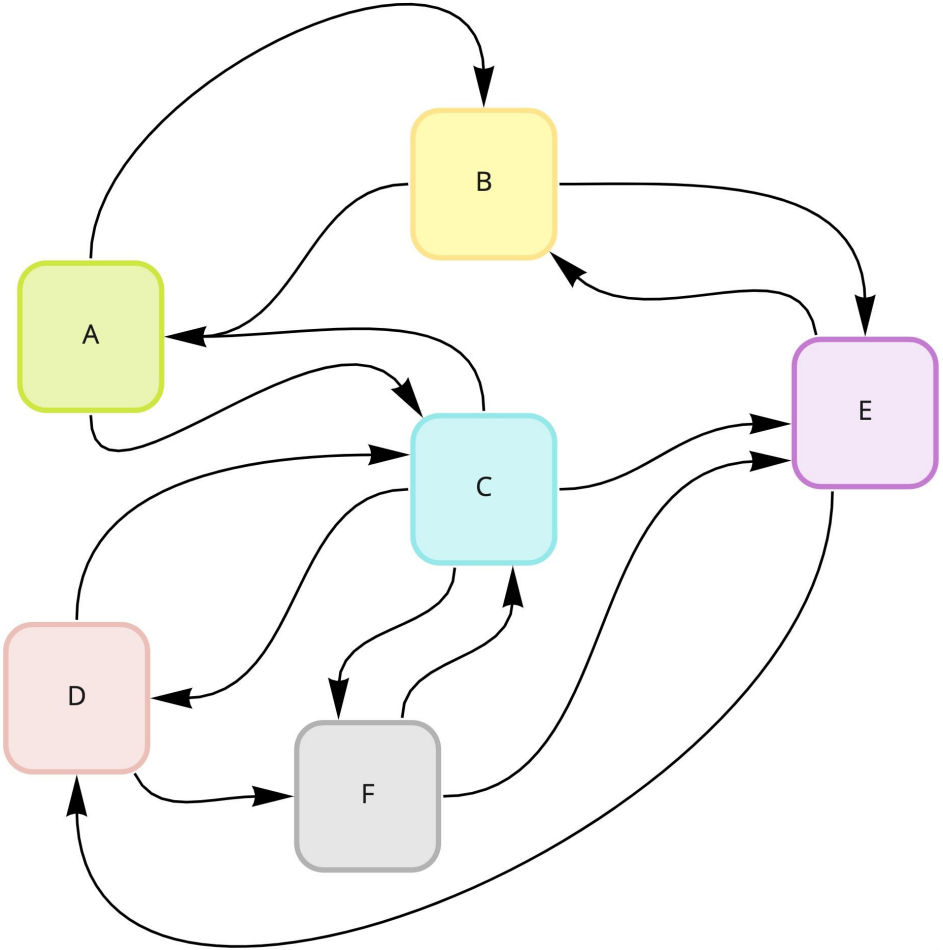
```
class FromActivity{
  companion object{
    const val REQUEST_CODE = 123
    const val RESULT_CODE = 456
  }

  fun startToActivity(){
    // [0] "From" activity에서 "To" activity로 전달해줄 데이터 지정
    val intent = Intent(this, ToActivity::class.java).putExtra("KEY", "VALUE")
    // [1] "From" activity에서 "To" activity 실행
    startActivityForResult(intent, REQUEST_CODE)
  }

  // [4] "FromActivity"의 onActivityResult 실행
  override fun onActivityResult(requestCode: Int, resultCode: Int, data: Inten
t?){
    super.onActivityResult(requestCode, resultCode, data)
    if(requestCode == REQUEST_CODE){
      if(resultCode == RESULT_CODE){
        // [5] REQUEST_CODE, RESULT_CODE를 통해
        // "From" -> "To" -> "From"로 되돌아온 케이스임을 판단하여 분기 처리
        textView.text = data.getStringExtra("RESULT_KEY")
      }
    }
  }
}
```

"From" activity



complex situations?
ex. multiple “from” activity
ex. multiple “to” activity

>> cubersome



type safety
null safety
..

risky

registerForActivityResult [New way🙂]

```
class FromActivity{
  companion object{
    const val REQUEST_CODE = 123
    const val RESULT_CODE = 456
  }
  private val toActivityLauncher
  = registerForActivityResult(ActivityResultContracts.StartActivityForResult()){
  result->
    if(requestCode == REQUEST_CODE){
      if(resultCode == RESULT_CODE){
        textView.text = data.getStringExtra("RESULT_KEY")
      }
    }
  }

  fun startToActivity(){
    val intent = Intent(this, ToActivity::class.java)
    intent.putExtra("KEY", "VALUE")
    toActivityLauncher.launch(intent)
  }
```

registerForActivityResult를 활용한 가장 심플한 예시

ActivityResultContract 활용하기 [Even better😊]

ActivityResultContract

Kotlin | Java

```
abstract class ActivityResultContract<I : Any?, O : Any?>
```

Known direct subclasses

ActivityResultContracts.CaptureVideo, ActivityResultContracts.CreateDocument, ActivityResultContracts.GetContent, ActivityResultContracts.GetMultipleContents, ActivityResultContracts.OpenDocumentTree, ActivityResultContracts.OpenDocument, ActivityResultContracts.OpenMultipleDocuments, ActivityResultContracts.PickContact, ActivityResultContracts.RequestMultiplePermissions, ActivityResultContracts.RequestPermission, ActivityResultContracts.StartActivityForResult, ActivityResultContracts.StartIntentSenderForResult, ActivityResultContracts.TakePicturePreview, ActivityResultContracts.TakePicture, ActivityResultContracts.TakeVideo, WatchFaceEditorContract

pseudocode

```
class ToContract: ActivityResultContract<String?, String?>() {

    override fun createIntent(context: Context, input: Any?): Intent {
        return Intent(context, ToActivity::class.java)
    }

    override fun parseResult(resultCode: Int, intent: Intent?): String? {
        return when (resultCode) {
            RESULT_CODE -> {
                intent?.getStringExtra("date")
            }
            else -> null
        }
    }
}
```

```
class FromActivity{
  private val toActivityLauncher = registerForActivityResult(ToContract()){
    it?.let{
      // 결과값 처리
    }
  }

  fun startToActivity(){
    val intent = Intent(this, ToActivity::class.java)
    toActivityLauncher.launch(intent)
  }
```

Before & After 개선 사항 정리

기존 방식의 불편함을 되새기면서 새로운 방식의 편리함에 좀 더 취해 보자.

`before` **복잡한 분기. 중복된 코드. 가독성 👎🏻**

→ onActivityResult에 길고 복잡한 if-else 블록들이 불가피하게 존재했다. 그리고 각 분기를 구분하기 위해 각종 RESULT_CODE, REQUST_CODE들도 필요했고, 겹치지 않게 하기 위한 관리도 필요했다.

`after` ActivityResultContract의 구현체에서 분기를 상당수 미리 처리할 수 있다. 최후 결과값에 대한 처리 로직만 ActivityResultCallback에서 작성하면 된다.

`before` **type safety 및 null safety 침해**

→ startActivityForResult 및 setResult의 인풋은 Intent일 뿐. 해당 Intent를 통해 전달되는 데이터들 자체의 type safety 및 null safety가 보장되지 않았다.

`after` createIntent, parseResult에서 명시적으로 type safety 및 null safety를 정의해줌으로써 안전성이 보장된다.
