JAVA ANDROID MOBILE PROGRAMMING: INTERMEDIATE

(INTEGRATIVE-GENERATIVE AI EDITION)

Author: Student Price Book Center

คำนำ

ในยุคที่สมาร์ตโฟนและอุปกรณ์พกพาได้กลายเป็นศูนย์กลางของการใช้ชีวิตประจำวัน การพัฒนาแอปพลิเคชันที่มีความสามารถเหนือกว่าระดับพื้นฐานถือเป็นสิ่งสำคัญสำหรับนักพัฒนา หนังสือ *Java Android Mobile Programming: Intermediate* เล่มนี้ถูกออกแบบมาเพื่อยกระดับความเข้าใจและทักษะของผู้อ่าน จากพื้นฐานสู่การพัฒนาแอป Android ที่ซับซ้อนขึ้น โดยใช้ภาษา **Java** ควบคู่กับเครื่องมือมาตรฐานใน Android Studio

เนื้อหาในเล่มนี้เริ่มต้นด้วย **บทที่ 6: Activity และ Intent** ซึ่งถือเป็นหัวใจสำคัญของการทำงานบน Android ผู้อ่านจะได้เรียนรู้วงจรชีวิตของ Activity อย่างละเอียด (Activity Lifecycle) การใช้ **Explicit และ Implicit Intent** เพื่อสื่อสารระหว่าง Activity และแอปอื่น รวมถึงวิธีส่งข้อมูลผ่าน putExtra, getIntent, Bundle และการใช้งาน **Serializable/Parcelable** เพื่อจัดการข้อมูลเชิงโครงสร้างได้อย่างมีประสิทธิภาพ

ต่อมาใน **บทที่ 7: Fragment และการจัดการหลายหน้าจอ** ผู้อ่านจะได้ทำความเข้าใจแนวคิดของ Fragment ซึ่งช่วยให้การพัฒนา UI ที่ซับซ้อนและรองรับหลายหน้าจอทำได้ง่ายขึ้น โดยมีการอธิบาย **Fragment Lifecycle**, การจัดการ **Fragment Transaction**, การสื่อสารระหว่าง Fragment และ Activity รวมถึงการสร้าง **Bottom Navigation, ViewPager และ TabLayout** เพื่อการใช้งานจริงในแอปที่มีหลายมุมมอง (multi-view applications)

บทที่ 8: RecyclerView และการแสดงข้อมูล มุ่งเน้นไปที่การแสดงผลข้อมูลที่มีจำนวนมากอย่างมีประสิทธิภาพ โดยนำเสนอการสร้าง **Custom Adapter**, การประยุกต์ใช้ **ViewHolder Pattern**, การจัดการ **Click Listener** รวมไปถึงการเพิ่ม **Divider และ ItemDecoration** เพื่อปรับปรุงการแสดงผลอย่างมืออาชีพ ทั้งนี้ยังมีตัวอย่างการประยุกต์ใช้งานจริงที่ผู้อ่านสามารถนำไปต่อยอดได้ทันที

จากนั้นใน **บทที่ 9: การเก็บข้อมูลในแอป** จะเจาะลึกเทคนิคการจัดเก็บข้อมูลภายในแอป Android ตั้งแต่การใช้ **SharedPreferences** เพื่อเก็บข้อมูลเบื้องต้น, การจัดการฐานข้อมูลด้วย **SQLite**, การทำงานร่วมกับ **ContentProvider** สำหรับการแชร์ข้อมูลระหว่างแอป และการใช้ **Room Persistence Library** ที่เป็นแนวทางสมัยใหม่ในการจัดการฐานข้อมูล พร้อมด้วยตัวอย่างโค้ดเชิงบูรณาการที่สามารถประยุกต์ใช้กับโครงการจริง

บทที่ 10: เชื่อมต่อ REST API นำผู้อ่านเข้าสู่โลกของการสื่อสารกับเซิร์ฟเวอร์ภายนอก ผ่านการทำงานของ **HTTP Request และ Response** พร้อมแนวทางจัดการงานเบื้องหลังอย่างถูกต้องเพื่อป้องกันปัญหาการบล็อก UI นอกจากนี้ยังมีการใช้งาน **Retrofit + OkHttp** ซึ่งเป็นไลบรารีมาตรฐานสำหรับการเชื่อมต่อ API บน Android และการแปลงข้อมูล JSON ให้เป็น Java Object ด้วย **Gson และ Moshi** เพื่อให้การเชื่อมต่อข้อมูลสมบูรณ์แบบ

สุดท้าย **บทที่ 11: การออกแบบ Material Design** จะช่วยให้นักพัฒนาสามารถสร้างประสบการณ์ผู้ใช้ที่สวยงามและเป็นมาตรฐาน โดยครอบคลุมการใช้ **Material Components** เช่น

CardView และ FloatingActionButton, การจัดการ **Toolbar และ AppBarLayout**, การสร้าง **Theme และ Style** ที่เหมาะสม รวมถึงการใช้ **Snackbar และ Toast** เพื่อสื่อสารกับผู้ใช้ได้อย่างมืออาชีพ พร้อมตัวอย่างเชิงบูรณาการที่สามารถปรับใช้ในแอปจริง

หนังสือเล่มนี้เหมาะสำหรับผู้ที่มีพื้นฐานการพัฒนา Android อยู่แล้ว และต้องการต่อยอดไปสู่ระดับกลาง (Intermediate) โดยมุ่งเน้นทั้งภาคทฤษฎีและภาคปฏิบัติ เพื่อให้ผู้อ่านเข้าใจเชิงลึก พร้อมลงมือสร้างแอปพลิเคชันที่สมบูรณ์และมีคุณภาพระดับมืออาชีพได้จริง

ด้วยรักและปรารถนาดี

ศูนย์หนังสือราคานักเรียน

สารบัญ

หน้า

บทที่ 6

Activity และ Intent

(Activity and Intent)

เนื้อหา

- Activity และ Intent
- Activity และ Intent (เชิงลึก)
- Activity Lifecycle (เชิงลึก)
- Explicit vs Implicit Intent สำหรับ Android + Java
- การส่งข้อมูลระหว่าง Activity บน Android + Java
- Bundle และ Serializable/Parcelable
- ตัวอย่างบูรณาการ

บทนี้จะพาผู้เรียนเข้าสู่การจัดการ **Activity และ Intent** ซึ่งเป็นหัวใจสำคัญของแอป Android โดยเริ่มจากการทำความเข้าใจ **Activity Lifecycle** ว่าการสร้าง, การแสดงผล, การหยุดชั่วคราว, และการทำลาย Activity เกิดขึ้นอย่างไร พร้อมอธิบาย callback methods สำคัญ เช่น onCreate(), onStart(), onResume(), onPause(), onStop(), และ onDestroy() เพื่อให้ผู้เรียนสามารถจัดการสถานะของ Activity อย่างเหมาะสม

ต่อมาเป็นเรื่องของ **Intent** ซึ่งเป็นกลไกหลักในการสื่อสารระหว่าง Activity และ Component ต่าง ๆ ของ Android แบ่งเป็น **Explicit Intent** สำหรับเรียก Activity หรือ Service ที่ระบุชื่อชัดเจน และ **Implicit Intent** สำหรับเรียก Component ที่ตอบสนองต่อ Action และ Data ที่กำหนด ทำให้สามารถเปิดแอปอื่นหรือแชร์ข้อมูลระหว่างแอปได้

บทนี้ยังเจาะลึกการ **ส่งข้อมูลระหว่าง Activity** โดยใช้ putExtra() และ getIntent() เพื่อส่งค่าข้อมูลพื้นฐาน เช่น String, int, boolean ระหว่าง Activity อย่างง่ายและตรงไปตรงมา นอกจากนี้ยังครอบคลุมการใช้ **Bundle** สำหรับจัดเก็บชุดข้อมูลหลายค่าและส่งผ่าน Activity

เพื่อรองรับข้อมูลที่ซับซ้อนกว่า ผู้เรียนจะได้เรียนรู้การใช้ **Serializable และ Parcelable** ซึ่งช่วยให้ส่ง Object ระหว่าง Activity ได้อย่างมีประสิทธิภาพ การเข้าใจความแตกต่างและข้อดีข้อเสียของแต่ละวิธีจะช่วยให้เลือกใช้ได้อย่างเหมาะสมกับประเภทข้อมูลและประสิทธิภาพของแอป

บทนี้ยังสอดแทรกตัวอย่างโค้ดและแนวทางปฏิบัติจริง เพื่อให้ผู้เรียนสามารถสร้าง Activity หลายหน้าจอ, ส่งข้อมูลระหว่าง Activity, และจัดการ Lifecycle ได้อย่างถูกต้องและปลอดภัย

นอกจากนี้ยังอธิบายแนวทาง **Debug และ Testing** Activity และ Intent เพื่อให้สามารถตรวจสอบการส่งข้อมูลและการทำงานของ Activity ได้อย่างถูกต้องบน Emulator และ Device จริง

บทนี้ถือเป็นรากฐานสำคัญที่ผู้เรียนต้องเข้าใจ ก่อนที่จะพัฒนาฟีเจอร์ที่ซับซ้อน เช่น การสื่อสารกับ Service, การใช้งาน Notification, และการจัดการ Multithreading ภายในแอป Android

Activity และ Intent

- Activity Lifecycle
- Explicit vs Implicit Intent
- การส่งข้อมูลระหว่าง Activity (putExtra, getIntent)
- การใช้ Bundle และ Serializable/Parcelable

รายละเอียดเชิงลึก **บทที่ 6: Activity และ Intent** สำหรับ Android + Java

---

บทที่ 6: Activity และ Intent

Android แอปจะประกอบด้วย **Activity** หลายหน้าจอ ซึ่งผู้ใช้สามารถโต้ตอบได้ การจัดการ **การเปลี่ยนหน้า (Navigation)** และ **การส่งข้อมูล** ระหว่าง Activity จึงเป็นเรื่องสำคัญ

---

1 Activity Lifecycle

Activity มี **Lifecycle** หลัก 7 ขั้นตอน:

Method	Description
onCreate()	สร้าง Activity และเรียกใช้งาน layout
onStart()	Activity กำลังจะมองเห็นหน้าจอ
onResume()	Activity พร้อมโต้ตอบผู้ใช้
onPause()	Activity กำลังถูกปิดชั่วคราว (เช่น เปิด Activity ใหม่)
onStop()	Activity ไม่ปรากฏบนหน้าจอ
onRestart()	Activity กลับมาเริ่มใหม่หลังจาก onStop()
onDestroy()	Activity ถูกทำลายจากระบบ

Tips:

- ใช้ onSaveInstanceState() + onRestoreInstanceState() เพื่อเก็บ/เรียกคืนสถานะ UI
- ใช้ Logcat เพื่อตรวจสอบ Lifecycle การเรียก Method

ตัวอย่าง Log Lifecycle

```
@Override
protected void onStart() {
```

```
    super.onStart();
    Log.d("Lifecycle", "onStart called");
}
```

---

2 Explicit vs Implicit Intent

Explicit Intent

- ใช้เปิด Activity ภายในแอป
- ระบุ **ชื่อ Activity ปลายทาง**

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
startActivity(intent);
```

Implicit Intent

- ใช้เรียกแอป/ฟีเจอร์ภายนอก เช่น โทรศัพท์, Browser, Email

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://www.google.com"));
startActivity(intent);
```

Tips:

- Explicit ⟶ ใช้ในแอปเดียว
- Implicit ⟶ ใช้เรียก App/Service ภายนอก

---

3 การส่งข้อมูลระหว่าง Activity

ใช้ putExtra / getIntent

```
// MainActivity.java
Intent intent = new Intent(this, SecondActivity.class);
intent.putExtra("USERNAME", "John");
startActivity(intent);

// SecondActivity.java
String name = getIntent().getStringExtra("USERNAME");
```

ชนิดข้อมูลที่รองรับ:

- String, int, boolean, float, double, long, Bundle, Serializable, Parcelable

---

4 การใช้ Bundle และ Serializable/Parcelable

Bundle

- เก็บหลายข้อมูลพร้อมกัน

```
Bundle bundle = new Bundle();
bundle.putString("NAME", "Alice");
bundle.putInt("AGE", 25);
intent.putExtras(bundle);
```

Serializable

- ใช้ส่ง Object แบบง่าย

```
public class User implements Serializable {
    String name;
    int age;
}

// ส่ง
intent.putExtra("USER", user);
```

Parcelable

- ประสิทธิภาพสูงกว่า Serializable

```
public class User implements Parcelable {
    String name;
    int age;

    protected User(Parcel in) {
        name = in.readString();
        age = in.readInt();
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(name);
        dest.writeInt(age);
    }
    ...
}
```

Tips:

- Serializable ⟶ ใช้ง่าย, ช้า
- Parcelable ⟶ ยุ่งยาก, เร็วกว่า ⟶ แนะนำสำหรับ Android

5 Best Practices

- เรียนรู้ Lifecycle ⟶ ป้องกัน Memory Leak
- ใช้ Explicit Intent สำหรับ Navigation ภายในแอป
- ใช้ Serializable/Parcelable สำหรับส่ง Object ขนาดใหญ่
- ใช้ Bundle เมื่อส่งหลายค่าพร้อมกัน

Activity และ Intent (เชิงลึก)

1 Activity และ Lifecycle

Activity คืออะไร:

- Activity คือ **หน้าจอ UI หนึ่ง** ในแอป Android
- แอปหนึ่งสามารถมีหลาย Activity
- Activity ทำงานบน **Main Thread (UI Thread)**

Lifecycle ของ Activity:

Activity มี **7 ขั้นตอนหลัก**:

Method	เมื่อถูกเรียก	ทำอะไร
onCreate()	Activity ถูกสร้าง	กำหนด layout, init UI, load data
onStart()	Activity กำลังปรากฏ	Activity จะมองเห็นบนหน้าจอ
onResume()	Activity พร้อมโต้ตอบ	UI Interaction, Sensor, Animation
onPause()	Activity ถูกซ้อน/ปิดชั่วคราว	Save temporary data, pause animations
onStop()	Activity ไม่มองเห็น	Release heavy resources, stop tasks
onRestart()	Activity กลับมา	เรียกก่อน onStart() หลังจาก onStop()
onDestroy()	Activity ถูกทำลาย	Release memory, cleanup resources

แนวคิดสำคัญ:

- **UI Thread** ⟶ การทำงานหนักต้องใช้ Background Thread (Thread, Coroutine, AsyncTask)
- **onSaveInstanceState() / onRestoreInstanceState()** ⟶ ใช้เก็บ/เรียกคืนสถานะ UI

ตัวอย่าง Logcat Debug Lifecycle

```
@Override
protected void onStart() {
```

```
    super.onStart();
    Log.d("ActivityLifecycle", "onStart called");
}
```

---

2 Intent: การสื่อสารระหว่าง Activity / App

Intent คืออะไร:

- เป็น **ข้อความกลาง (Message Object)** ใช้สื่อสารระหว่าง Component ของ Android เช่น Activity, Service, Broadcast Receiver

ประเภท Intent

1. **Explicit Intent**
    - ใช้เปิด Activity ภายในแอป
    - ระบุ Class ของ Activity ปลายทาง
2. Intent intent = new Intent(MainActivity.this, SecondActivity.class);
3. startActivity(intent);
4. **Implicit Intent**
    - ใช้เรียก App / Service ภายนอก เช่น Browser, Call, Email
5. Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://www.google.com"));
6. startActivity(intent);

แนวทางปฏิบัติ:

- Explicit ⟶ Navigation ภายในแอป
- Implicit ⟶ Integration กับระบบ Android หรือ App อื่น

---

3 การส่งข้อมูลระหว่าง Activity

3.1 putExtra / getIntent

- ส่งค่าแบบง่าย (String, int, boolean, float)

```
// ส่งข้อมูล
Intent intent = new Intent(this, SecondActivity.class);
intent.putExtra("USERNAME", "John");
startActivity(intent);

// รับข้อมูล
String name = getIntent().getStringExtra("USERNAME");
```

3.2 Bundle

- ส่งข้อมูลหลายตัวพร้อมกัน

```
Bundle bundle = new Bundle();
bundle.putString("NAME", "Alice");
bundle.putInt("AGE", 25);
intent.putExtras(bundle);

// รับข้อมูล
Bundle data = getIntent().getExtras();
String name = data.getString("NAME");
int age = data.getInt("AGE");
```

3.3 Serializable / Parcelable

- ส่ง Object ขนาดใหญ่
- Serializable ⟶ ง่ายแต่ช้า
- Parcelable ⟶ ยุ่งยากแต่เร็ว

Serializable Example

```
public class User implements Serializable {
    String name;
    int age;
}

// ส่ง
intent.putExtra("USER", user);

// รับ
User user = (User) getIntent().getSerializableExtra("USER");
```

Parcelable Example

```
public class User implements Parcelable {
    String name;
    int age;

    protected User(Parcel in) {
        name = in.readString();
        age = in.readInt();
    }
```

```
    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(name);
        dest.writeInt(age);
    }

    public static final Creator<User> CREATOR = new Creator<User>() {
        @Override
        public User createFromParcel(Parcel in) {
            return new User(in);
        }
        @Override
        public User[] newArray(int size) {
            return new User[size];
        }
    };
}
```

---

4 Activity Lifecycle + Intent การบูรณาการ

หลักการสำคัญ:

1. ตรวจสอบ Lifecycle ก่อนส่ง/รับข้อมูล
2. ใช้ Bundle / Parcelable สำหรับข้อมูลขนาดใหญ่
3. ใช้ Explicit Intent สำหรับ Activity ภายใน, Implicit Intent สำหรับ App อื่น
4. เก็บค่าใน onSaveInstanceState() เพื่อตอบสนอง Rotation / Configuration Change

ตัวอย่างเชิงแนวคิด:

- MainActivity ⟶ ส่งชื่อผู้ใช้ไป SecondActivity
- SecondActivity ⟶ แสดงข้อความ “Hello, [Name]!”
- ใช้ Parcelable หากส่ง Object ขนาดใหญ่

---

5 Best Practices

- **Memory Management:**
    - อย่าเก็บ Context ใน Object นานเกินไป ⟶ Memory Leak
- **Bundle / Serializable / Parcelable:**

- Serializable ⟶ สะดวก, Parcelables ⟶ เร็ว, แนะนำสำหรับ Android

- **Lifecycle Awareness:**
  - ใช้ ViewModel + LiveData / StateFlow ร่วมกับ Activity ⟶ ป้องกันข้อมูลหายเมื่อลอง Rotate
- **Testing:**
  - Emulator ⟶ Test หน้าจอ, Rotation, API Level
  - Device จริง ⟶ Test Performance, Touch, Gesture, Sensor

Activity Lifecycle (เชิงลึก)

Activity Lifecycle คืออะไร:

- Activity คือหน้าจอของแอป Android
- Lifecycle คือ **ชุดขั้นตอน (Callback Methods)** ที่ Android เรียกเมื่อ Activity ถูกสร้าง, แสดง, หยุด หรือทำลาย
- การเข้าใจ Lifecycle สำคัญมากสำหรับ **การจัดการ UI, Resource, Background Task และการป้องกัน Memory Leak**

---

1 Callback Methods หลัก

Method	เรียกเมื่อ	สิ่งที่ควรทำ
onCreate(Bundle savedInstanceState)	Activity ถูกสร้างครั้งแรก	กำหนด layout, initialize UI, bind data
onStart()	Activity กำลังจะมองเห็น	เริ่ม tasks ที่เกี่ยวกับ UI
onResume()	Activity พร้อมโต้ตอบ	เริ่ม Animation, Sensor, Video, Audio
onPause()	Activity กำลังซ้อน / กำลังหยุด	Save temporary data, pause Animation/Video
onStop()	Activity ไม่ปรากฏบนหน้าจอ	Release memory-heavy resource, stop background tasks
onRestart()	Activity กลับมาแสดงหลัง onStop()	เรียก onStart() ต่อ
onDestroy()	Activity ถูกทำลาย	Cleanup resources, finalize

---

2 Lifecycle Flow Diagram

onCreate() ⟶ onStart() ⟶ onResume()

↘ Pause ⟶ onPause() ⟶ onStop() ⟶ onRestart() ⟶ onStart() ⟶ onResume()

↘ Finish ⟶ onPause() ⟶ onStop() ⟶ onDestroy()

Tips:

- onPause() ⟶ สถานะกึ่งหยุด, Activity ยังอยู่ใน memory
- onStop() ⟶ Activity ซ่อนจากผู้ใช้
- onDestroy() ⟶ Activity ถูกทำลาย, Resource ต้อง cleanup

---

3 ตัวอย่างโค้ด Log Lifecycle

MainActivity.java

```
package com.example.lifecycleexample;

import android.os.Bundle;
import android.util.Log;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "ActivityLifecycle";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(TAG, "onCreate called");
    }

    @Override
    protected void onStart() {
        super.onStart();
        Log.d(TAG, "onStart called");
    }

    @Override
```

```
    protected void onResume() {
        super.onResume();
        Log.d(TAG, "onResume called");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.d(TAG, "onPause called");
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.d(TAG, "onStop called");
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        Log.d(TAG, "onRestart called");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d(TAG, "onDestroy called");
    }
}
```

4 การสังเกตผลลัพธ์ (Logcat)

Action	Lifecycle Callback ถูกเรียก
เปิดแอป	onCreate ⟶ onStart ⟶ onResume

Action **Lifecycle Callback ถูกเรียก**

กด Home onPause ⟶ onStop

กลับ App onRestart ⟶ onStart ⟶ onResume

กด Back onPause ⟶ onStop ⟶ onDestroy

---

5 การใช้งานจริง

1. **onCreate()** ⟶ initialize UI, binding, data fetch
2. **onStart() / onResume()** ⟶ start animations, sensor, video
3. **onPause() / onStop()** ⟶ pause animations, save state, release resource
4. **onDestroy()** ⟶ cleanup final memory-heavy objects

ตัวอย่าง:

- Save EditText text ใน onPause()
- Restore text ใน onResume()
- Release MediaPlayer ใน onStop()

---

6 Best Practices

- ใช้ **ViewModel / LiveData** ⟶ รักษาข้อมูลข้าม rotation โดยไม่พึ่ง Lifecycle callbacks
- **Background Task** ⟶ ปิดใน onPause() หรือ onStop()
- **Memory Leak Prevention** ⟶ ปล่อย resource เช่น Sensor, Camera, MediaPlayer ใน onStop() หรือ onDestroy()

ตัวอย่างโปรแกรม Activity Lifecycle แบบเต็มไฟล์ พร้อม **โครงสร้าง, คำอธิบายโค้ด และผลการรัน** จำนวน 3 โปรแกรมพื้นฐาน และ 3 โปรแกรมแนวประยุกต์

ตัวอย่างพื้นฐาน 3 โปรแกรม – Activity Lifecycle

ตัวอย่าง 1: Lifecycle Log Basic

โครงสร้างโปรเจกต์

```
LifecycleBasic/
├── app/src/main/java/com/example/lifecycle/MainActivity.java
├── app/src/main/res/layout/activity_main.xml
└── AndroidManifest.xml
```

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:gravity="center"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/tvStatus"
        android:text="Lifecycle Logger"
        android:textSize="24sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

MainActivity.java

```
package com.example.lifecycle;

import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    TextView tvStatus;
    private static final String TAG = "ActivityLifecycle";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tvStatus = findViewById(R.id.tvStatus);
        Log.d(TAG, "onCreate called");
        tvStatus.setText("onCreate called");
```

```
}

@Override
protected void onStart() {
    super.onStart();
    Log.d(TAG, "onStart called");
    tvStatus.setText("onStart called");
}

@Override
protected void onResume() {
    super.onResume();
    Log.d(TAG, "onResume called");
    tvStatus.setText("onResume called");
}

@Override
protected void onPause() {
    super.onPause();
    Log.d(TAG, "onPause called");
    tvStatus.setText("onPause called");
}

@Override
protected void onStop() {
    super.onStop();
    Log.d(TAG, "onStop called");
    tvStatus.setText("onStop called");
}

@Override
protected void onRestart() {
    super.onRestart();
    Log.d(TAG, "onRestart called");
```

```
        tvStatus.setText("onRestart called");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d(TAG, "onDestroy called");
        tvStatus.setText("onDestroy called");
    }
}
```

ผลการรัน:

- เปิดแอป ⟶ TextView แสดง “onCreate called”
- กด Home ⟶ “onPause called” ⟶ “onStop called”
- กลับ App ⟶ “onRestart called” ⟶ “onStart called” ⟶ “onResume called”
- กด Back ⟶ “onPause called” ⟶ “onStop called” ⟶ “onDestroy called”

ตัวอย่าง 2: Save/Restore State

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="16dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/etInput"
        android:hint="Type something"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/tvDisplay"
        android:text="Your text appears here"
```

```
        android:textSize="18sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"/>
</LinearLayout>
```

MainActivity.java

```
package com.example.lifecycle;

import android.os.Bundle;
import android.widget.EditText;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    EditText etInput;
    TextView tvDisplay;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        etInput = findViewById(R.id.etInput);
        tvDisplay = findViewById(R.id.tvDisplay);

        if(savedInstanceState != null){
            tvDisplay.setText(savedInstanceState.getString("TEXT_KEY"));
        }
    }

    @Override
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
```

```
        outState.putString("TEXT_KEY", etInput.getText().toString());
    }
}
```

ผลการรัน:

- พิมพ์ข้อความ ⟶ หมุนจอ ⟶ ข้อความยังคงอยู่
- แสดงการใช้งาน onSaveInstanceState() และ onRestoreInstanceState()

ตัวอย่าง 3: Two Activities Lifecycle

โครงสร้างโปรเจกต์

```
LifecycleTwoActivities/
├── app/src/main/java/com/example/lifecycle/MainActivity.java
├── app/src/main/java/com/example/lifecycle/SecondActivity.java
├── app/src/main/res/layout/activity_main.xml
├── app/src/main/res/layout/activity_second.xml
└── AndroidManifest.xml
```

MainActivity.java

```
package com.example.lifecycle;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button btnNext = findViewById(R.id.btnNext);
        btnNext.setOnClickListener(v -> {
            Intent intent = new Intent(this, SecondActivity.class);
            startActivity(intent);
        });
```

```
    }
}
```

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:gravity="center"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/btnNext"
        android:text="Open Second Activity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

SecondActivity.java

```
package com.example.lifecycle;

import android.os.Bundle;
import android.util.Log;
import androidx.appcompat.app.AppCompatActivity;

public class SecondActivity extends AppCompatActivity {

    private static final String TAG = "SecondActivityLifecycle";

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        Log.d(TAG, "onCreate called");
    }

    @Override
```

```
    protected void onStart(){
        super.onStart();
        Log.d(TAG, "onStart called");
    }

    @Override
    protected void onResume(){
        super.onResume();
        Log.d(TAG, "onResume called");
    }

    @Override
    protected void onPause(){
        super.onPause();
        Log.d(TAG, "onPause called");
    }

    @Override
    protected void onStop(){
        super.onStop();
        Log.d(TAG, "onStop called");
    }

    @Override
    protected void onDestroy(){
        super.onDestroy();
        Log.d(TAG, "onDestroy called");
    }
}
```

activity_second.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:gravity="center"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent">

    <TextView
        android:text="Second Activity"
        android:textSize="24sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

ผลการรัน:

- MainActivity ⟶ กดปุ่ม ⟶ SecondActivity เปิด ⟶ Logcat แสดง Lifecycle ของ SecondActivity
- กด Back ⟶ SecondActivity ถูกทำลาย ⟶ MainActivity onResume ถูกเรียก

Explicit vs Implicit Intent สำหรับ Android + Java

Explicit vs Implicit Intent

ใน Android **Intent** คือวัตถุ (Object) ที่ใช้สำหรับ **สื่อสารระหว่าง Component** เช่น Activity, Service หรือ Broadcast Receiver

---

1 Explicit Intent

Explicit Intent ใช้เมื่อเราต้องการเปิด **Activity ภายในแอปของเราเอง**

- ระบุ **ชื่อ Activity ปลายทาง** โดยตรง
- ใช้สำหรับ Navigation ภายในแอป

ตัวอย่างโค้ด

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
startActivity(intent);
```

ขั้นตอน

1. สร้าง Intent ระบุ **Context** และ **Class ของ Activity ปลายทาง**
2. เรียก startActivity(intent)
3. Activity ปลายทางจะเริ่มทำงานตาม Lifecycle

ตัวอย่างการส่งข้อมูลพร้อม Explicit Intent

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
intent.putExtra("USERNAME", "John");
```

```
startActivity(intent);
```

ใน SecondActivity

```
String name = getIntent().getStringExtra("USERNAME");
```

---

2 Implicit Intent

Implicit Intent ใช้เมื่อเราต้องการให้ **ระบบ Android เลือก App/Component ที่สามารถตอบสนองได้**

- เช่น โทรศัพท์, Browser, Email, Camera
- ไม่ต้องระบุ Class ปลายทาง

ตัวอย่างโค้ด

```
// เปิด Browser
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://www.google.com"));
startActivity(intent);

// โทรศัพท์
Intent callIntent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:0812345678"));
startActivity(callIntent);
```

หลักการทำงาน

1. Android จะหา Component ที่ **รองรับ Action + Data**
2. ถ้ามีหลาย App ⟶ ระบบให้เลือก App (Chooser)
3. ถ้าไม่มี ⟶ App จะ crash ⟶ ต้องใช้ Intent.resolveActivity() เพื่อตรวจสอบ

ตัวอย่างตรวจสอบก่อนเรียก

```
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(intent);
} else {
    Toast.makeText(this, "No App can handle this action", Toast.LENGTH_SHORT).show();
}
```

---

3 ตัวอย่าง Action/Category ของ Implicit Intent

Action	Usage
ACTION_VIEW	เปิด Browser, Gallery
ACTION_DIAL	เปิด Dialer
ACTION_SEND	Share Text/Image

Action	Usage
ACTION_PICK	เลือก Item จาก Contacts / Media
ACTION_EDIT	เปิด Editor ของ File

Category:

- CATEGORY_DEFAULT ⟶ เริ่ม Activity ปกติ
- CATEGORY_BROWSABLE ⟶ เปิดจาก Browser / Web link

---

4 เปรียบเทียบ Explicit vs Implicit

Feature	Explicit Intent	Implicit Intent
ระบุ Activity	ใช่	ไม่จำเป็น
ใช้ใน App	ภายใน	ภายนอก/ภายใน
การส่งข้อมูล	ผ่าน putExtra	ผ่าน putExtra / Uri
การเลือก App	ไม่มี	ระบบอาจให้เลือก App
ตัวอย่าง	new Intent(this, SecondActivity.class)	new Intent(Intent.ACTION_VIEW, Uri.parse(url))

---

5 Best Practices

1. ใช้ Explicit Intent สำหรับ **Navigation ภายในแอป**
2. ใช้ Implicit Intent สำหรับ **เรียก App อื่น / ฟีเจอร์ระบบ**
3. ตรวจสอบด้วย resolveActivity() ก่อนเรียก Implicit Intent ⟶ ป้องกัน crash
4. ส่งข้อมูลผ่าน putExtra หรือ Uri
5. ใช้ Chooser เพื่อให้ผู้ใช้เลือก App ⟶ เพิ่ม UX

ตัวอย่างโปรแกรม Explicit vs Implicit Intent แบบเต็มไฟล์ พร้อม **โครงสร้างโปรเจกต์, คำอธิบายโค้ด และผลการรัน**

ตัวอย่างพื้นฐาน 3 โปรแกรม – Explicit vs Implicit Intent

ตัวอย่าง 1: Explicit Intent (Activity ภายใน)

โครงสร้างโปรเจกต์

ExplicitIntentBasic/

```
├── app/src/main/java/com/example/explicitintent/MainActivity.java
├── app/src/main/java/com/example/explicitintent/SecondActivity.java
├── app/src/main/res/layout/activity_main.xml
├── app/src/main/res/layout/activity_second.xml
└── AndroidManifest.xml
```

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="16dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/btnOpenSecond"
        android:text="Open Second Activity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

MainActivity.java

```
package com.example.explicitintent;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        Button btn = findViewById(R.id.btnOpenSecond);
        btn.setOnClickListener(v -> {
            Intent intent = new Intent(MainActivity.this, SecondActivity.class);
            intent.putExtra("MESSAGE", "Hello from MainActivity!");
            startActivity(intent);
        });
    }
}
```

SecondActivity.java

```
package com.example.explicitintent;

import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class SecondActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);

        TextView tv = findViewById(R.id.tvMessage);
        String message = getIntent().getStringExtra("MESSAGE");
        tv.setText(message);
    }
}
```

activity_second.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:gravity="center"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
```

```
        android:id="@+id/tvMessage"
        android:textSize="20sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

ผลการรัน:

- กดปุ่มใน MainActivity ⟶ เปิด SecondActivity ⟶ แสดงข้อความ “Hello from MainActivity!”

ตัวอย่าง 2: Implicit Intent – เปิด Browser

activity_main.xml (ใช้ LinearLayout + Button)

```
<Button
    android:id="@+id/btnOpenBrowser"
    android:text="Open Google"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

MainActivity.java

```
package com.example.implicitintent;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.widget.Button;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button btn = findViewById(R.id.btnOpenBrowser);
        btn.setOnClickListener(v -> {
```

```
            Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://www.google.com"));
            if (intent.resolveActivity(getPackageManager()) != null) {
                startActivity(intent);
            } else {
                Toast.makeText(this, "No browser app found", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

ผลการรัน:

- กดปุ่ม ⟶ ระบบเปิด Browser ⟶ เข้า Google
- ถ้าไม่มี Browser ⟶ แสดง Toast “No browser app found”

ตัวอย่าง 3: Implicit Intent – โทรศัพท์

activity_main.xml

```
<Button
    android:id="@+id/btnCall"
    android:text="Call Number"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

MainActivity.java

```
package com.example.implicitintentcall;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.widget.Button;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);

        Button btn = findViewById(R.id.btnCall);
        btn.setOnClickListener(v -> {
            Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:0812345678"));
            if (intent.resolveActivity(getPackageManager()) != null) {
                startActivity(intent);
            } else {
                Toast.makeText(this, "No dialer app found", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

ผลการรัน:

- กดปุ่ม ⟶ เปิด Dialer ⟶ หมายเลข 0812345678
- ถ้าไม่มี Dialer ⟶ แสดง Toast

ตัวอย่างแนวประยุกต์ 3 โปรแกรม – Explicit & Implicit Intent

ตัวอย่าง 4: ส่ง Object ด้วย Explicit Intent

- สร้าง Class User implements Serializable
- MainActivity ⟶ กดปุ่มส่ง User ⟶ SecondActivity ⟶ แสดงชื่อและอายุ

```
// User.java
public class User implements Serializable {
    public String name;
    public int age;
    public User(String name, int age){
        this.name = name; this.age = age;
    }
}
```

MainActivity.java

```
User user = new User("Alice", 25);
Intent intent = new Intent(this, SecondActivity.class);
intent.putExtra("USER_OBJ", user);
```

```
startActivity(intent);
```

SecondActivity.java

```
User user = (User) getIntent().getSerializableExtra("USER_OBJ");
tvMessage.setText(user.name + ", " + user.age);
```

ตัวอย่าง 5: Share Text ด้วย Implicit Intent

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_TEXT, "Hello, share this text!");
startActivity(Intent.createChooser(intent, "Share via"));
```

- ระบบจะเปิด Chooser ⟶ ผู้ใช้เลือก App เพื่อแชร์ข้อความ

ตัวอย่าง 6: เปิด Map ด้วย Implicit Intent + Safety Check

```
Uri location = Uri.parse("geo:13.7563,100.5018?q=Bangkok");
Intent intent = new Intent(Intent.ACTION_VIEW, location);
intent.setPackage("com.google.android.apps.maps");

if(intent.resolveActivity(getPackageManager()) != null){
    startActivity(intent);
}else{
    Toast.makeText(this,"Google Maps not installed",Toast.LENGTH_SHORT).show();
}
```

- ถ้า Google Maps ติดตั้ง ⟶ เปิด Map
- ถ้าไม่ติดตั้ง ⟶ แสดง Toast

การส่งข้อมูลระหว่าง Activity บน Android + Java

การส่งข้อมูลระหว่าง Activity

ใน Android เราสามารถส่งข้อมูลจาก Activity หนึ่งไปยังอีก Activity หนึ่งได้โดยใช้ **Intent**

- ใช้ **putExtra()** เพื่อส่งข้อมูล
- ใช้ **getIntent()** + **getXXXExtra()** เพื่อดึงข้อมูลใน Activity ปลายทาง

---

1 การส่งข้อมูลแบบพื้นฐาน (Primitive Types)

MainActivity.java

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
intent.putExtra("USERNAME", "John");
intent.putExtra("AGE", 25);
startActivity(intent);
```

SecondActivity.java

```
Intent intent = getIntent();
String username = intent.getStringExtra("USERNAME");
int age = intent.getIntExtra("AGE", 0); // default 0
```

---

2 การส่ง Object (Serializable / Parcelable)

Serializable

```
// User.java
public class User implements Serializable {
    public String name;
    public int age;
    public User(String name, int age) {
        this.name = name;
        this.age = age;
    }
}

// MainActivity.java
User user = new User("Alice", 30);
Intent intent = new Intent(this, SecondActivity.class);
intent.putExtra("USER_OBJ", user);
startActivity(intent);

// SecondActivity.java
User user = (User) getIntent().getSerializableExtra("USER_OBJ");
```

Parcelable (เร็วกว่า Serializable)

```
// User.java
public class User implements Parcelable {
    public String name;
```

```
    public int age;

    protected User(Parcel in) {
        name = in.readString();
        age = in.readInt();
    }

    public static final Creator<User> CREATOR = new Creator<User>() {
        @Override
        public User createFromParcel(Parcel in) {
            return new User(in);
        }
        @Override
        public User[] newArray(int size) {
            return new User[size];
        }
    };

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(name);
        dest.writeInt(age);
    }

    @Override
    public int describeContents() { return 0; }

    public User(String name, int age){
        this.name = name; this.age = age;
    }
}

// MainActivity.java
intent.putExtra("USER_PARCEL", user);
```

```
// SecondActivity.java
User user = getIntent().getParcelableExtra("USER_PARCEL");
```

---

3 การรับผลลัพธ์จาก Activity (startActivityForResult → ActivityResult API)

ใหม่ (API ≥ 29) ใช้ **ActivityResultLauncher** แทน startActivityForResult

MainActivity.java

```
ActivityResultLauncher<Intent> launcher = registerForActivityResult(
    new ActivityResultContracts.StartActivityForResult(),
    result -> {
        if(result.getResultCode() == Activity.RESULT_OK){
            String reply = result.getData().getStringExtra("REPLY");
        }
    }
);

Button btn = findViewById(R.id.btnOpen);
btn.setOnClickListener(v -> {
    Intent intent = new Intent(this, SecondActivity.class);
    launcher.launch(intent);
});
```

SecondActivity.java

```
Button btnReply = findViewById(R.id.btnReply);
btnReply.setOnClickListener(v -> {
    Intent resultIntent = new Intent();
    resultIntent.putExtra("REPLY", "Hello MainActivity!");
    setResult(RESULT_OK, resultIntent);
    finish();
});
```

---

4 ตัวอย่างประเภทข้อมูลที่ส่งได้ด้วย putExtra()

Type	Method
int	putExtra(String, int) / getIntExtra()

Type	Method
float	putExtra(String, float) / getFloatExtra()
boolean	putExtra(String, boolean) / getBooleanExtra()
String	putExtra(String, String) / getStringExtra()
Serializable Object	putExtra(String, Serializable) / getSerializableExtra()
Parcelable Object	putExtra(String, Parcelable) / getParcelableExtra()

---

5 Best Practices

1. **ใช้ Keys เป็น Constant** ⟶ ป้องกัน typo

```
public static final String EXTRA_USERNAME = "USERNAME";
```

2. **เลือก Serializable หรือ Parcelable** ⟶ Parcelable เร็วกว่า Serializable
3. **ตรวจสอบ null ก่อนใช้ getIntent()**
4. **สำหรับ ActivityResult** ⟶ ใช้ ActivityResultLauncher แทน startActivityForResult

---

ตัวอย่างโปรแกรมส่งข้อมูลระหว่าง Activity (putExtra / getIntent) แบบเต็มไฟล์ พร้อม **โครงสร้าง, คำอธิบายโค้ด และผลการรัน** จำนวน 3 โปรแกรมพื้นฐาน และ 3 โปรแกรมแนวประยุกต์

ตัวอย่างพื้นฐาน 3 โปรแกรม – putExtra/getIntent

ตัวอย่าง 1: ส่ง String ระหว่าง Activity

โครงสร้างโปรเจกต์

```
SendString/
├── app/src/main/java/com/example/sendstring/MainActivity.java
├── app/src/main/java/com/example/sendstring/SecondActivity.java
├── app/src/main/res/layout/activity_main.xml
├── app/src/main/res/layout/activity_second.xml
└── AndroidManifest.xml
```

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="16dp"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/etInput"
        android:hint="Enter your name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <Button
        android:id="@+id/btnSend"
        android:text="Send to Second Activity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"/>
</LinearLayout>
```

MainActivity.java

```
package com.example.sendstring;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        EditText etInput = findViewById(R.id.etInput);
        Button btnSend = findViewById(R.id.btnSend);
```

```
        btnSend.setOnClickListener(v -> {
            String name = etInput.getText().toString();
            Intent intent = new Intent(MainActivity.this, SecondActivity.class);
            intent.putExtra("USERNAME", name);
            startActivity(intent);
        });
    }
}
```

SecondActivity.java

```
package com.example.sendstring;

import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class SecondActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);

        TextView tv = findViewById(R.id.tvDisplay);
        String username = getIntent().getStringExtra("USERNAME");
        tv.setText("Hello, " + username + "!");
    }
}
```

activity_second.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:gravity="center"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
```

```
        android:id="@+id/tvDisplay"
        android:textSize="24sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

ผลการรัน:

- พิมพ์ชื่อ ⟶ กดปุ่ม ⟶ เปิด SecondActivity ⟶ แสดง “Hello, [ชื่อ]!”

ตัวอย่าง 2: ส่ง Integer และ Boolean

MainActivity.java

```
int age = 30;
boolean isMember = true;

Intent intent = new Intent(this, SecondActivity.class);
intent.putExtra("AGE", age);
intent.putExtra("MEMBER", isMember);
startActivity(intent);
```

SecondActivity.java

```
int age = getIntent().getIntExtra("AGE", 0);
boolean isMember = getIntent().getBooleanExtra("MEMBER", false);
tvDisplay.setText("Age: " + age + ", Member: " + isMember);
```

ผลการรัน:

- แสดง Age และ Member status ตามข้อมูลที่ส่ง

ตัวอย่าง 3: ส่ง Object แบบ Serializable

User.java

```
package com.example.sendobject;

import java.io.Serializable;

public class User implements Serializable {
    public String name;
    public int age;
```

```
    public User(String name, int age){
        this.name = name; this.age = age;
    }
}
```

MainActivity.java

```
User user = new User("Alice", 28);
Intent intent = new Intent(this, SecondActivity.class);
intent.putExtra("USER_OBJ", user);
startActivity(intent);
```

SecondActivity.java

```
User user = (User) getIntent().getSerializableExtra("USER_OBJ");
tvDisplay.setText("Name: " + user.name + ", Age: " + user.age);
```

ผลการรัน:

- แสดงข้อมูล Object “Name: Alice, Age: 28”

ตัวอย่างแนวประยุกต์ 3 โปรแกรม

ตัวอย่าง 4: Serializable Object + EditText + Button

- MainActivity: กรอกชื่อและอายุ ⟶ ส่ง Object User ⟶ SecondActivity แสดงข้อมูล

activity_main.xml

- EditText สำหรับ Name + Age
- Button ส่งข้อมูล

SecondActivity.java

- แสดง User.name + User.age ใน TextView

ผลการรัน:

- ผู้ใช้กรอกข้อมูล ⟶ กดส่ง ⟶ SecondActivity แสดงข้อมูลครบ

ตัวอย่าง 5: Parcelable Object + ImageView

- MainActivity: สร้าง Object UserParcelable (name, age, imageResId) ⟶ ส่งไป SecondActivity
- SecondActivity: แสดงข้อมูลและรูปภาพ

ผลการรัน:

- SecondActivity แสดงชื่อ, อายุ และรูปภาพตาม Object ที่ส่ง

ตัวอย่าง 6: ActivityResultLauncher (รับผลลัพธ์กลับ)

MainActivity.java

```
ActivityResultLauncher<Intent> launcher = registerForActivityResult(
    new ActivityResultContracts.StartActivityForResult(),
    result -> {
        if(result.getResultCode() == RESULT_OK){
            String reply = result.getData().getStringExtra("REPLY");
            tvReply.setText("Reply: " + reply);
        }
    }
);

btnOpen.setOnClickListener(v -> {
    Intent intent = new Intent(this, SecondActivity.class);
    launcher.launch(intent);
});
```

SecondActivity.java

```
btnReply.setOnClickListener(v -> {
    Intent resultIntent = new Intent();
    resultIntent.putExtra("REPLY", "Hello MainActivity!");
    setResult(RESULT_OK, resultIntent);
    finish();
});
```

ผลการรัน:

- MainActivity ⟶ กดปุ่ม ⟶ เปิด SecondActivity ⟶ กดปุ่มส่งกลับ ⟶ MainActivity แสดงข้อความ Reply

---

Bundle และ Serializable/Parcelable

---

1 การใช้ Bundle

Bundle เป็น Container สำหรับเก็บคู่ Key-Value เพื่อส่งข้อมูลระหว่าง Activity หรือ Fragment

- สามารถเก็บ Primitive Types, String, Parcelable, Serializable, Array, List ได้

ตัวอย่างการใช้ Bundle กับ Intent

MainActivity.java

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
Bundle bundle = new Bundle();
bundle.putString("USERNAME", "John");
bundle.putInt("AGE", 25);
intent.putExtras(bundle); // แนบ Bundle ไปกับ Intent
startActivity(intent);
```

SecondActivity.java

```
Bundle bundle = getIntent().getExtras();
if(bundle != null){
    String username = bundle.getString("USERNAME");
    int age = bundle.getInt("AGE");
    tvDisplay.setText("Name: " + username + ", Age: " + age);
}
```

จุดเด่นของ Bundle

- เหมาะสำหรับส่ง **หลายค่าในครั้งเดียว**
- สามารถแนบกับ **Intent, Fragment arguments, Service**

---

2 การใช้ Serializable

Serializable เป็น interface สำหรับทำให้ Object สามารถ **แปลงเป็น byte stream** ⟶ ส่งผ่าน Intent/Bundle

User.java

```
import java.io.Serializable;


public class User implements Serializable {
    public String name;
    public int age;
    public User(String name, int age){
        this.name = name;
        this.age = age;
    }
}
```

ส่ง Object ผ่าน Bundle

```
User user = new User("Alice", 30);
```

```
Bundle bundle = new Bundle();
bundle.putSerializable("USER_OBJ", user);
Intent intent = new Intent(this, SecondActivity.class);
intent.putExtras(bundle);
startActivity(intent);
```

รับ Object ใน SecondActivity

```
Bundle bundle = getIntent().getExtras();
if(bundle != null){
    User user = (User) bundle.getSerializable("USER_OBJ");
    tvDisplay.setText("Name: " + user.name + ", Age: " + user.age);
}
```

ข้อดี: ใช้ง่าย ไม่ต้องเขียนโค้ดมาก

ข้อเสีย: ช้ากว่า Parcelable ⟶ ใช้กับ Object ขนาดใหญ่ไม่เหมาะ

---

3 การใช้ Parcelable

Parcelable เป็น interface ของ Android สำหรับส่ง Object ระหว่าง Component

- เร็วกว่า Serializable เพราะ Android ใช้ native code ในการ parcel/unparcel

UserParcelable.java

```
import android.os.Parcel;
import android.os.Parcelable;

public class UserParcelable implements Parcelable {
    public String name;
    public int age;

    public UserParcelable(String name, int age){
        this.name = name;
        this.age = age;
    }

    protected UserParcelable(Parcel in){
        name = in.readString();
        age = in.readInt();
    }
```

```
    public static final Creator<UserParcelable> CREATOR = new Creator<UserParcelable>() {
        @Override
        public UserParcelable createFromParcel(Parcel in) {
            return new UserParcelable(in);
        }
        @Override
        public UserParcelable[] newArray(int size) {
            return new UserParcelable[size];
        }
    };

    @Override
    public int describeContents() { return 0; }

    @Override
    public void writeToParcel(Parcel dest, int flags){
        dest.writeString(name);
        dest.writeInt(age);
    }
}
```

ส่ง Object ผ่าน Bundle

```
UserParcelable user = new UserParcelable("Bob", 35);
Bundle bundle = new Bundle();
bundle.putParcelable("USER_PARCEL", user);
Intent intent = new Intent(this, SecondActivity.class);
intent.putExtras(bundle);
startActivity(intent);
```

รับ Object ใน SecondActivity

```
Bundle bundle = getIntent().getExtras();
if(bundle != null){
    UserParcelable user = bundle.getParcelable("USER_PARCEL");
    tvDisplay.setText("Name: " + user.name + ", Age: " + user.age);
}
```

ข้อดี:

- เร็วและเหมาะกับ Android
- รองรับ Parcelable Array, List, Bundle nested

ข้อเสีย:

- ต้องเขียนโค้ดเยอะกว่า Serializable

---

4 สรุปการเลือกใช้

วิธี	ใช้เมื่อ	ข้อดี	ข้อเสีย
Bundle + Primitive/String	ส่งหลายค่าเล็ก ๆ	ง่าย, อ่านง่าย	ใช้กับ Object ขนาดใหญ่ไม่ได้
Serializable	ส่ง Object ขนาดเล็ก	เขียนง่าย	ช้า, ใช้กับ Object ขนาดใหญ่ไม่เหมาะ
Parcelable	ส่ง Object ขนาดใหญ่, performance	เร็ว, Android optimized	ต้องเขียนโค้ดมาก

ตัวอย่างโปรแกรมส่งข้อมูลระหว่าง Activity ด้วย Bundle + Serializable/Parcelable แบบเต็มไฟล์ พร้อม **โครงสร้างโปรเจกต์, คำอธิบายโค้ด และผลการรัน** จำนวน 3 โปรแกรมพื้นฐาน และ 3 โปรแกรมแนวประยุกต์

ตัวอย่างพื้นฐาน 3 โปรแกรม – Bundle + Serializable/Parcelable

ตัวอย่าง 1: ส่งข้อมูล String + int ด้วย Bundle

โครงสร้างโปรเจกต์

```
BundleBasic/
├── app/src/main/java/com/example/bundlebasic/MainActivity.java
├── app/src/main/java/com/example/bundlebasic/SecondActivity.java
├── app/src/main/res/layout/activity_main.xml
├── app/src/main/res/layout/activity_second.xml
└── AndroidManifest.xml
```

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:gravity="center"
```

```
    android:padding="16dp" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/btnSendBundle"
        android:text="Send Bundle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

MainActivity.java

```
package com.example.bundlebasic;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button btnSend = findViewById(R.id.btnSendBundle);
        btnSend.setOnClickListener(v -> {
            Bundle bundle = new Bundle();
            bundle.putString("USERNAME", "John");
            bundle.putInt("AGE", 25);

            Intent intent = new Intent(MainActivity.this, SecondActivity.class);
            intent.putExtras(bundle);
            startActivity(intent);
        });
    }
```

```
}
```

SecondActivity.java

```
package com.example.bundlebasic;

import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class SecondActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);

        TextView tv = findViewById(R.id.tvDisplay);
        Bundle bundle = getIntent().getExtras();
        if(bundle != null){
            String username = bundle.getString("USERNAME");
            int age = bundle.getInt("AGE");
            tv.setText("Name: " + username + ", Age: " + age);
        }
    }
}
```

activity_second.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:gravity="center"
    android:layout_width="match_parent" android:layout_height="match_parent">

    <TextView
        android:id="@+id/tvDisplay"
        android:textSize="24sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

ผลการรัน:

- กดปุ่ม ⟶ SecondActivity แสดง “Name: John, Age: 25”

ตัวอย่าง 2: ส่ง Object แบบ Serializable

User.java

```
package com.example.bundlebasic;

import java.io.Serializable;

public class User implements Serializable {
    public String name;
    public int age;

    public User(String name, int age){
        this.name = name;
        this.age = age;
    }
}
```

MainActivity.java

```
User user = new User("Alice", 30);
Bundle bundle = new Bundle();
bundle.putSerializable("USER_OBJ", user);

Intent intent = new Intent(this, SecondActivity.class);
intent.putExtras(bundle);
startActivity(intent);
```

SecondActivity.java

```
Bundle bundle = getIntent().getExtras();
if(bundle != null){
    User user = (User) bundle.getSerializable("USER_OBJ");
    tvDisplay.setText("Name: " + user.name + ", Age: " + user.age);
}
```

ผลการรัน:

- SecondActivity แสดง “Name: Alice, Age: 30”

ตัวอย่าง 3: ส่ง Object แบบ Parcelable

UserParcelable.java

```
package com.example.bundlebasic;

import android.os.Parcel;
import android.os.Parcelable;

public class UserParcelable implements Parcelable {
    public String name;
    public int age;

    public UserParcelable(String name, int age){
        this.name = name;
        this.age = age;
    }

    protected UserParcelable(Parcel in){
        name = in.readString();
        age = in.readInt();
    }

    public static final Creator<UserParcelable> CREATOR = new Creator<UserParcelable>() {
        @Override
        public UserParcelable createFromParcel(Parcel in) {
            return new UserParcelable(in);
        }
        @Override
        public UserParcelable[] newArray(int size) {
            return new UserParcelable[size];
        }
    };

    @Override
```

```
    public int describeContents() { return 0; }

    @Override
    public void writeToParcel(Parcel dest, int flags){
        dest.writeString(name);
        dest.writeInt(age);
    }
}
```

MainActivity.java

```
UserParcelable user = new UserParcelable("Bob", 35);
Bundle bundle = new Bundle();
bundle.putParcelable("USER_PARCEL", user);

Intent intent = new Intent(this, SecondActivity.class);
intent.putExtras(bundle);
startActivity(intent);
```

SecondActivity.java

```
Bundle bundle = getIntent().getExtras();
if(bundle != null){
    UserParcelable user = bundle.getParcelable("USER_PARCEL");
    tvDisplay.setText("Name: " + user.name + ", Age: " + user.age);
}
```

ผลการรัน:

- SecondActivity แสดง “Name: Bob, Age: 35”

ตัวอย่างแนวประยุกต์ 3 โปรแกรม

ตัวอย่าง 4: Serializable Object + EditText + Button

- MainActivity: กรอกชื่อและอายุ ⟶ สร้าง User Serializable ⟶ ส่งผ่าน Bundle ⟶ SecondActivity แสดงข้อมูล
- **ผลการรัน:** กรอกชื่อ/อายุ ⟶ กดปุ่ม ⟶ SecondActivity แสดงข้อมูลครบ

ตัวอย่าง 5: Parcelable Object + ImageView

- MainActivity: สร้าง UserParcelable (name, age, imageResId) ⟶ ส่ง Bundle ⟶ SecondActivity แสดงชื่อ, อายุ, และรูปภาพ
- **ผลการรัน:** SecondActivity แสดงชื่อ, อายุ, รูปภาพตาม Object ที่ส่ง

ตัวอย่าง 6: Bundle Nested + List of Parcelable

- MainActivity: สร้าง ArrayList ⟶ ใส่ใน Bundle ⟶ ส่งไป SecondActivity
- SecondActivity: รับ List ⟶ Loop แสดงแต่ละ User ใน TextView
- **ผลการรัน:** แสดงรายการ User หลายคนบนหน้าจอเดียว

ตัวอย่าง 4: Serializable Object + EditText + Button

โครงสร้างโปรเจกต์

SerializableApp/

```
├── app/src/main/java/com/example/serializableapp/MainActivity.java
├── app/src/main/java/com/example/serializableapp/SecondActivity.java
├── app/src/main/java/com/example/serializableapp/User.java
├── app/src/main/res/layout/activity_main.xml
├── app/src/main/res/layout/activity_second.xml
└── AndroidManifest.xml
```

User.java

```
package com.example.serializableapp;

import java.io.Serializable;

public class User implements Serializable {
    public String name;
    public int age;

    public User(String name, int age){
        this.name = name;
        this.age = age;
    }
}
```

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:gravity="center"
    android:padding="16dp" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/etName"
        android:hint="Enter Name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <EditText
        android:id="@+id/etAge"
        android:hint="Enter Age"
        android:inputType="number"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <Button
        android:id="@+id/btnSend"
        android:text="Send User"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"/>
</LinearLayout>
```

MainActivity.java

```
package com.example.serializableapp;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;
```

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        EditText etName = findViewById(R.id.etName);
        EditText etAge = findViewById(R.id.etAge);
        Button btnSend = findViewById(R.id.btnSend);

        btnSend.setOnClickListener(v -> {
            String name = etName.getText().toString();
            int age = Integer.parseInt(etAge.getText().toString());
            User user = new User(name, age);

            Intent intent = new Intent(MainActivity.this, SecondActivity.class);
            Bundle bundle = new Bundle();
            bundle.putSerializable("USER_OBJ", user);
            intent.putExtras(bundle);
            startActivity(intent);
        });
    }
}
```

SecondActivity.java

```
package com.example.serializableapp;

import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class SecondActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_second);

        TextView tv = findViewById(R.id.tvDisplay);
        Bundle bundle = getIntent().getExtras();
        if(bundle != null){
            User user = (User) bundle.getSerializable("USER_OBJ");
            tv.setText("Name: " + user.name + "\nAge: " + user.age);
        }
    }
}
```

activity_second.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:gravity="center"
    android:layout_width="match_parent" android:layout_height="match_parent">

    <TextView
        android:id="@+id/tvDisplay"
        android:textSize="24sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

ผลการรัน:

- กรอกชื่อและอายุ ⟶ กดปุ่ม ⟶ SecondActivity แสดงข้อมูลครบ

ตัวอย่าง 5: Parcelable Object + ImageView

UserParcelable.java

```
package com.example.parcelableapp;

import android.os.Parcel;
import android.os.Parcelable;

public class UserParcelable implements Parcelable {
    public String name;
    public int age;
```