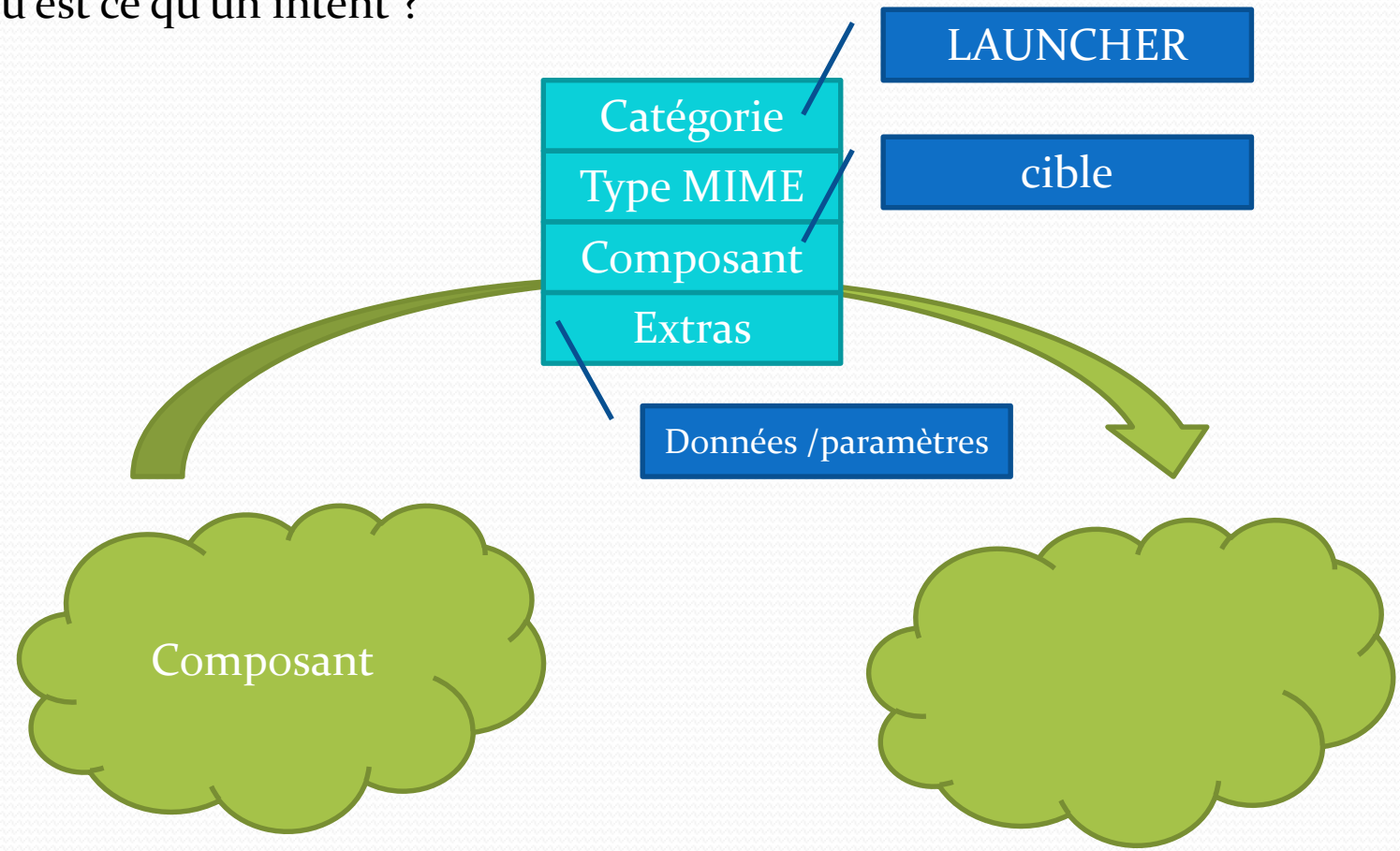
PROGRAMMATION AVEC ANDROID

Intent / Coopération entre activités

Communication entre composants

Qu'est ce qu'un intent ?



Communication entre composants

Une de vos activité appelle une autre de vos activités

Une de vos activité appelle une fonctionnalité d'une autre application

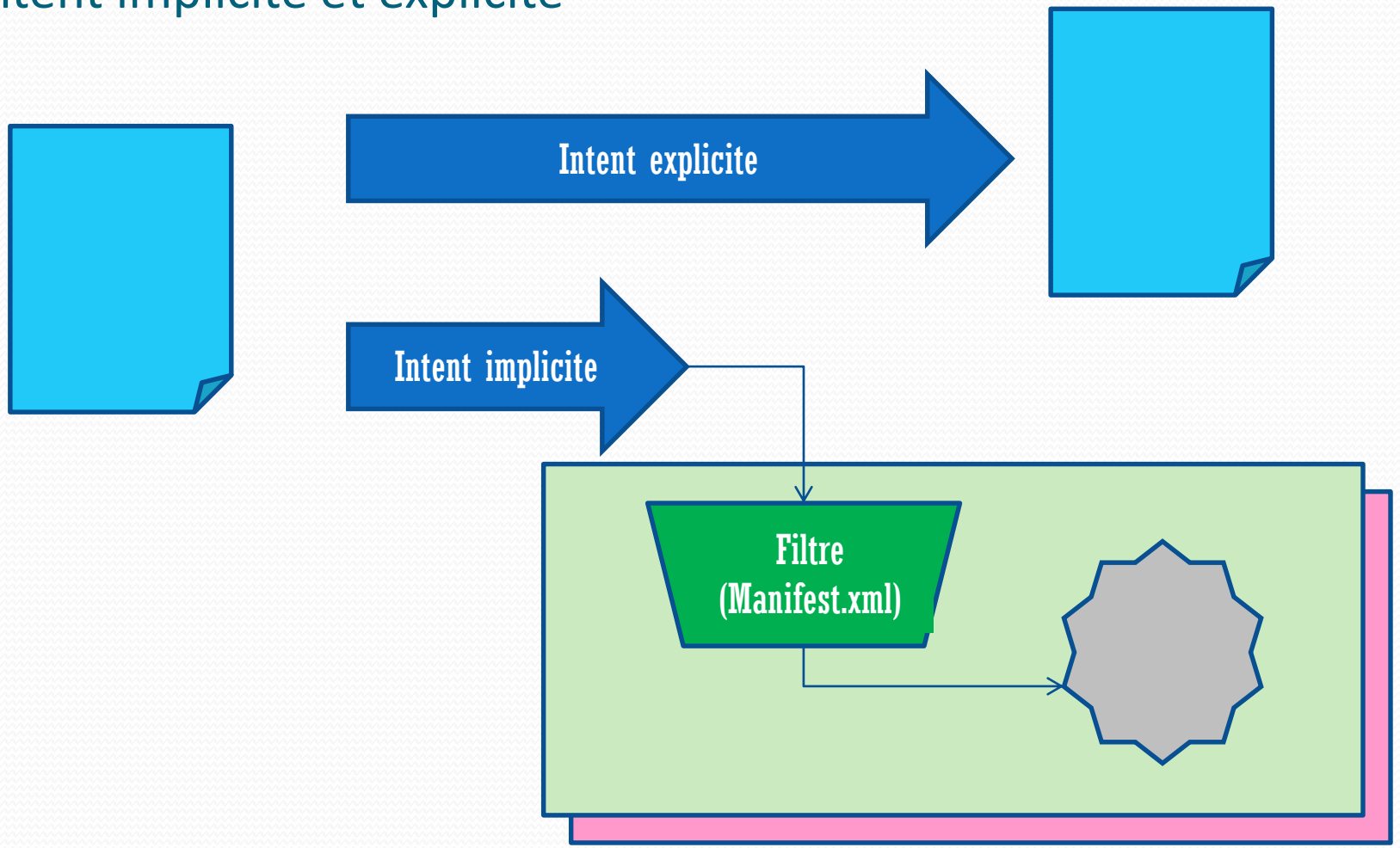
Une de vos activité appelle une fonctionnalité système

Le système appelle un composant de votre activité

Votre application offre une fonctionnalité à une application tierce

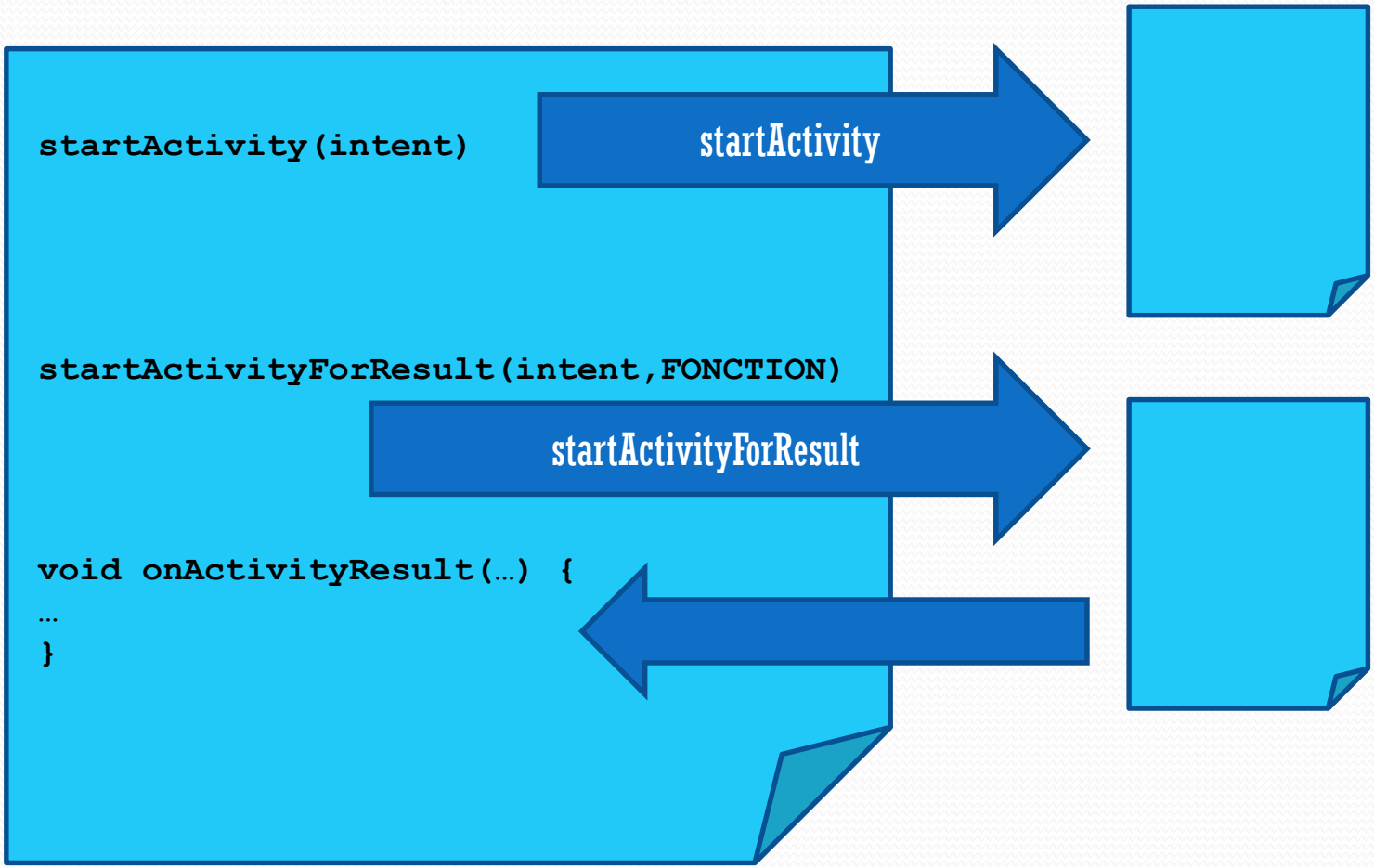
Communication entre composants

Intent implicite et explicite



Communication entre composants

lancement avec ou sans resultat



Exemple

```
public void changeDate(View view) {
    Intent intent = new Intent(this, ChangeDateActivity.class);
    startActivityForResult(intent, SHOW_CHANGEDATE) ;
    }

protected void onActivityResult(int requestCode, int resultCode,
   Intent data) {
        if (requestCode == SHOW_CHANGEDATE) {
            if (resultCode == RESULT_OK) {
            String date =
                  data.getStringExtra(ChangeDateActivity.DATE_KEY) ;
            tvDate.setText(date) ;
            }
        }
    }
```

```
public class ChangeDateActivity extends Activity {
    protected DatePicker dpSelecteurDate ;
    public final static String DATE_KEY = "fr.univlille1.jmp.selectedDate" ;
    public void onCreate(Bundle  savedInstanceState) {
            super.onCreate(savedInstanceState) ;
            setContentView(R.layout.changedate) ;
    }
    public void onOk(View view) {
            java.text.DateFormat df ;
            df = DateFormat.getDateFormat(this) ;
            dpSelecteurDate = (DatePicker) findViewById(R.id.idSelectDate) ;
            Intent reponse = new Intent() ;
            GregorianCalendar d = new GregorianCalendar(dpSelecteurDate.getYear(),
                        dpSelecteurDate.getMonth(), dpSelecteurDate.getDayOfMonth()) ;
            reponse.putExtra(DATE_KEY, df.format(d.getTime())) ;
            dpSelecteurDate.getYear() ;
            setResult(RESULT_OK, reponse) ;
            finish() ;
    }
    …
```

```
...
    public void onAnnumer(View view) {
            setResult(RESULT_CANCELED, null) ;
            finish() ;
    }
}
```

PROGRAMMATION AVEC ANDROID

Données persistantes

Données persistantes

3 modalités

- Préférences (Données simples)
  - PreferenceActivity
- Fichiers simple
  - Espace local à l'application
  - Espace externe
  - Attention: Droits
  - Attention: Réactivité
- Bases de données

PROGRAMMATION AVEC ANDROID

Bas de données embarquée.

Accès aux bases de données

Première problématique:

La première exécution d'une application doit avoir un comportement différent des autres exécutions (création des tables, initialisation des données).

De même, si pour une nouvelle version de l'application il y a peut être des modifications (de table, de colonne) à opérer.

Comment faire ?

Accès aux bases de données

SQLiteOpenHelper fournit le support qui offre une solution simple par:

- La création d'un classe dérivée de SQLiteOpenHelpr.
- Le super constructeur permet de spécifier la base et le no de version de la base.
- Si il s'agit d'une nouvelle base la méthode `onCreate(SQLiteDatabase)` (à surcharger) est appelée.
- Si il s'agit d'une mise à jour, la méthode `onUpgrade(SQLiteDatabase, int oldVersion, int newVersion)` est appelée.

Accès aux bases de données

La classe offre finalement deux méthode permettant d'obtenir un objet SQLiteDatabase accessible en lecture (`getReadableDatabase()`) ou en écriture (`getWritableDatabase()`).

Application (réalisation).

Création de la classe Helper:

```
public class SqlConnect extends SQLiteOpenHelper {
static protected final String SQL_NAME = "maBase.db" ;
static protected final int SQL_VERSION = 1 ;
static protected final String
   SQL_CREATE = "CREATE TABLE val (nom TEXT PRIMARY KEY, valeur INTEGER) ;" ;
static protected final String
   SQL_INSERT_A = "INSERT INTO val VALUES ('A', 10) ; " ;
static protected final String
   SQL_INSERT_B = "INSERT INTO val VALUES ('B', 40) ; " ;
protected SQLiteDatabase db = null ;
public SqlConnect(Context context) {
   super(context, SQL_NAME, null, SQL_VERSION) ;
   db = getWritableDatabase() ;
}
public void onCreate(SQLiteDatabase db) {
   db.execSQL(SQL_CREATE) ;
   db.execSQL(SQL_INSERT_A) ;
   db.execSQL(SQL_INSERT_B) ;
}
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
}
…
```

Application (réalisation).

Création de la classe Helper:

```
public class SqlConnect extends SQLiteOpenHelper {
static protected final String SQL_NAME = "maBase.db" ;
static protected final int SQL_VERSION = 1 ;
static protected final String
   SQL_CREATE = "CREATE TABLE val (nom TEXT PRIMARY KEY, valeur INTEGER) ;" ;
static protected final String
   SQL_INSERT_A = "INSERT INTO val VALUES ('A', 10) ; " ;
static protected final String
   SQL_INSERT_B = "INSERT INTO val VALUES ('B', 40) ; " ;
…
public SqlConnect(Context context) {
   super(context, SQL_NAME, null, SQL_VERSION) ;
…
}
public void onCreate(SQLiteDatabase db) {
   db.execSQL(SQL_CREATE) ;
   db.execSQL(SQL_INSERT_A) ;
   db.execSQL(SQL_INSERT_B) ;
}
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
}
…
```

Application (réalisation).

Il est possible d'intégrer à la classe Helper tous les accès:

```
public class SqlConnect extends SQLiteOpenHelper {
    ...
    protected SQLiteDatabase db = null ;
    public SqlConnect(Context context) {
                        super(context, SQL_NAME, null, SQL_VERSION) ;
                        db = getWritableDatabase() ;
    }
    public int readA() {
    ...
    }
    public int readB() {
    ...
    }
    public void writeA(int a) {
    ...
    }
    public void writeB(int b) {
    ...
    }
}
```

Application (lecture d'une valeur).

Il est possible d'intégrer à la classe Helper tous les accès:

```
public int readA() {
   String[] results = { "valeur"} ;
   String where = "nom = 'A'" ;
   Cursor    a = db.query("val", new String[] {"valeur"},
                        where, null, null, "", "", "") ;
   a.moveToFirst() ;
   return a.getInt(0) ;
}
```

Les paramètres de db.query:

- Nom de la table.
- Liste des colonnes (String[])
- Clause where
- Les arguments éventuels de la clause where
- Clause group by
- Clause order by

Application (écriture d'une valeur).

Il est possible d'intégrer à la classe Helper tous les accès:

```
public void writeA(int a) {
   ContentValues updated = new ContentValues() ;
   updated.put("valeur", a) ;
   db.update("val", updated, "nom = 'A'", null) ;
}
```

Les paramètres de db.update:

Nom de la table.

Liste des colonnes/valeurs (ContentValues)

Clause where

Les arguments éventuels de la clause where