MICROOH 麦可网

Android-从程序员到架构师之路

出品人：Sundy
讲师：高焕堂（台湾）

http://www.microoh.com

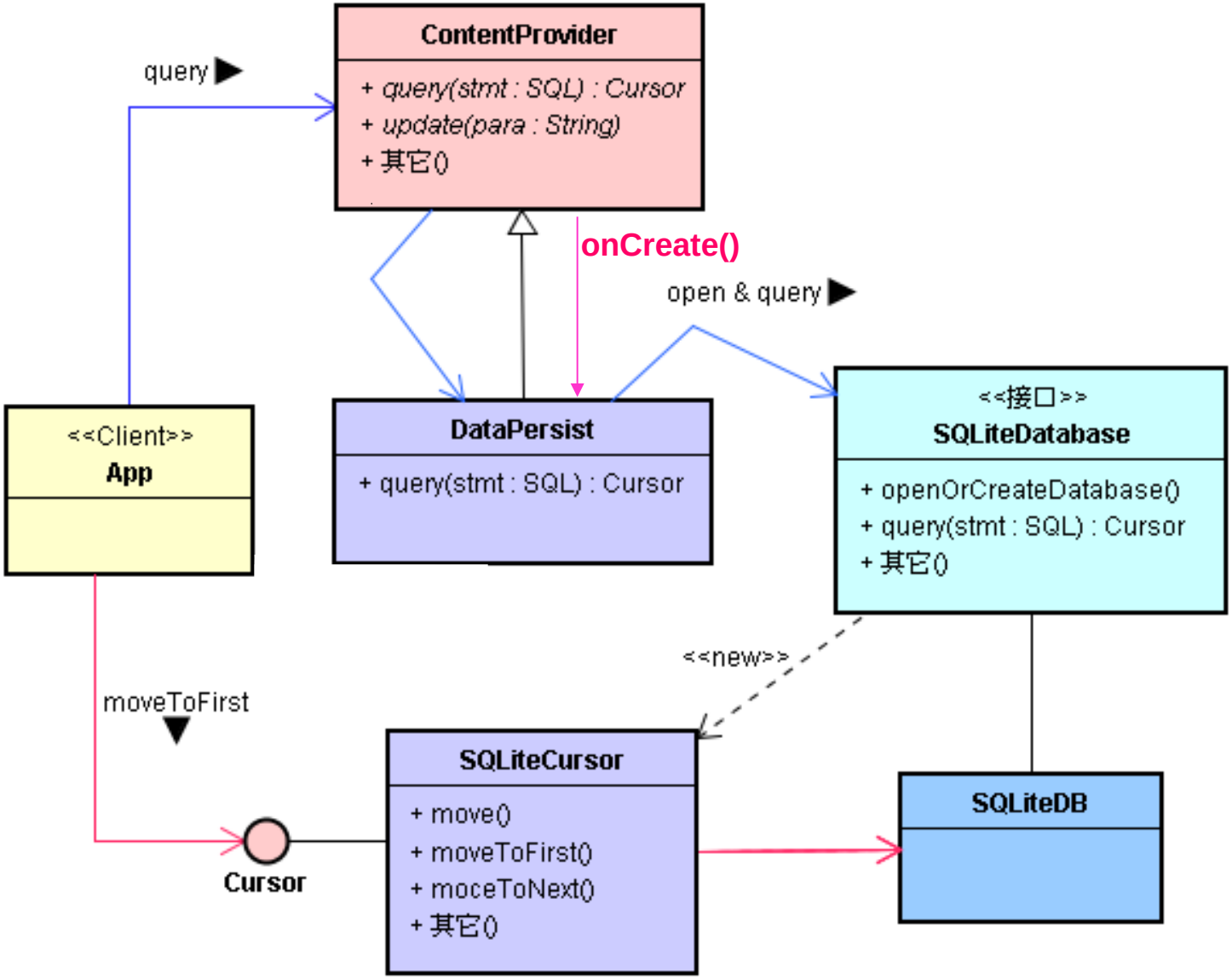
F06_d

观摩：ContentProvider 架构與DB引擎移植方法(d)

By 高煥堂

5、展现DB引擎的变换自由度：以Linter引擎的移植为例

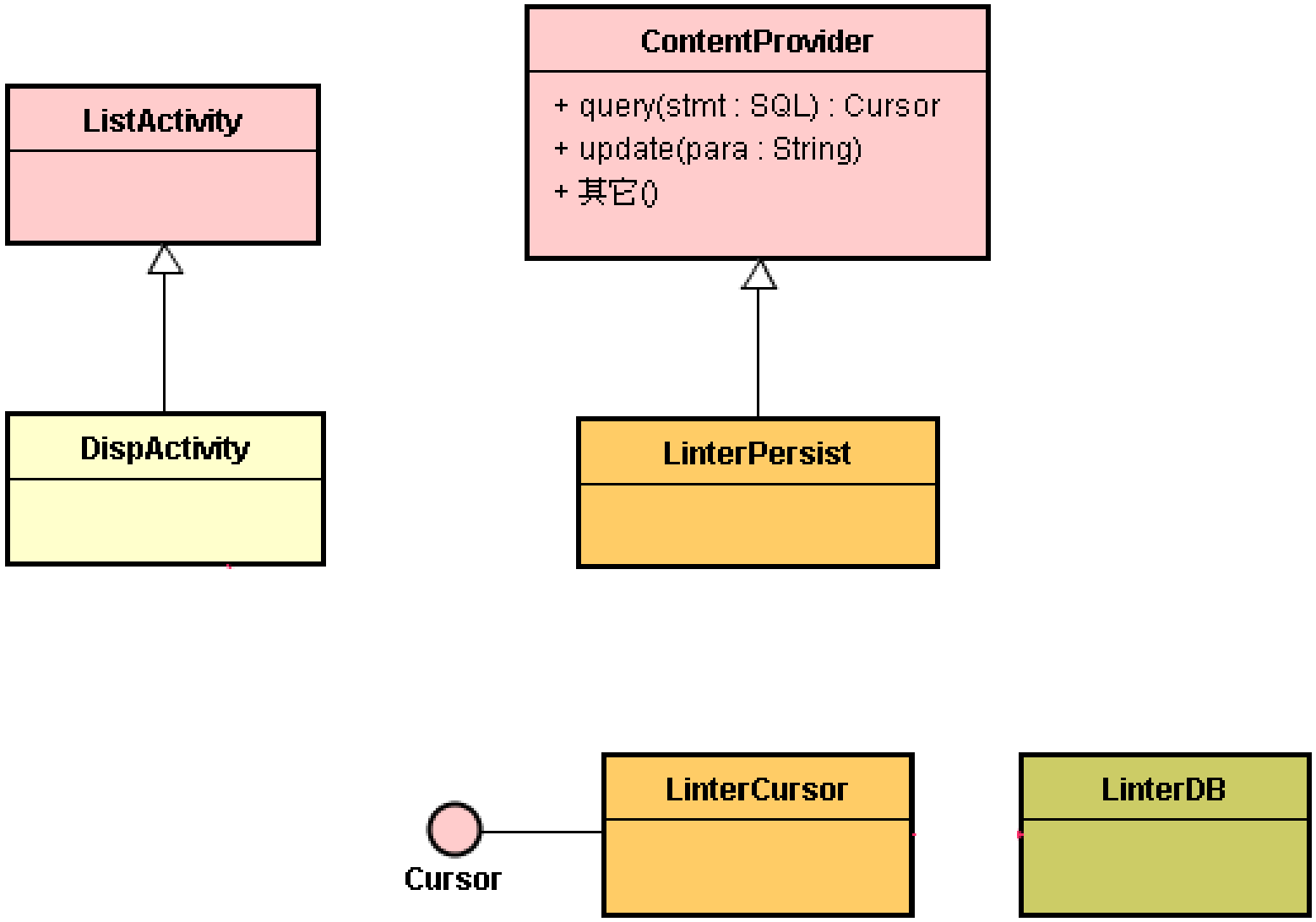
- 在上一个范例里，Client使用ContentProvider接口与SQLite DB引擎衔接。

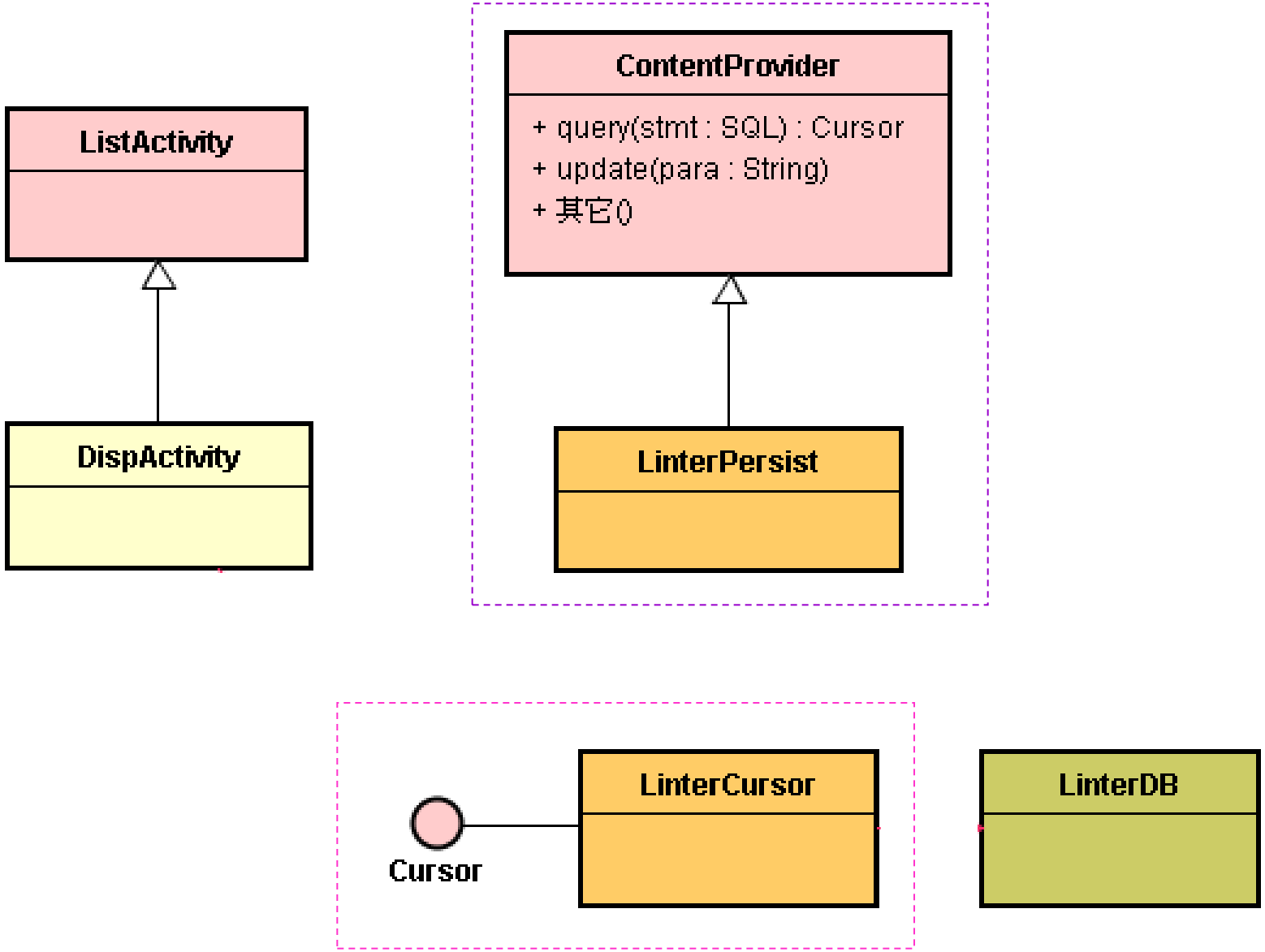

ContentProvider
+ query(stmt : SQL) : Cursor
+ update(para : String)
+ 其它()
query
onCreate()
open & query
<<Client>>
App
DataPersist
+ query(stmt : SQL) : Cursor
<<接口>>
SQLiteDatabase
+ openOrCreateDatabase()
+ query(stmt : SQL) : Cursor
+ 其它()
<<new>>
moveToFirst
SQLiteCursor
+ move()
+ moveToFirst()
+ moceToNext()
+ 其它()
SQLiteDB
Cursor

- DataPersist对象是由Android框架所创建的。
- 之后，Client就调用getContentResolver()函数来要求Android框架去进行配对和绑定此DataPersist对象。然后间接绑定了LinterDB引擎。

- 此ContentProvider接口与特定DB引擎是无关的，可以让Client与DB引擎互为独立。非常有助于双方的独立成长，或各自的版本更新，甚至新DB引擎的移植。

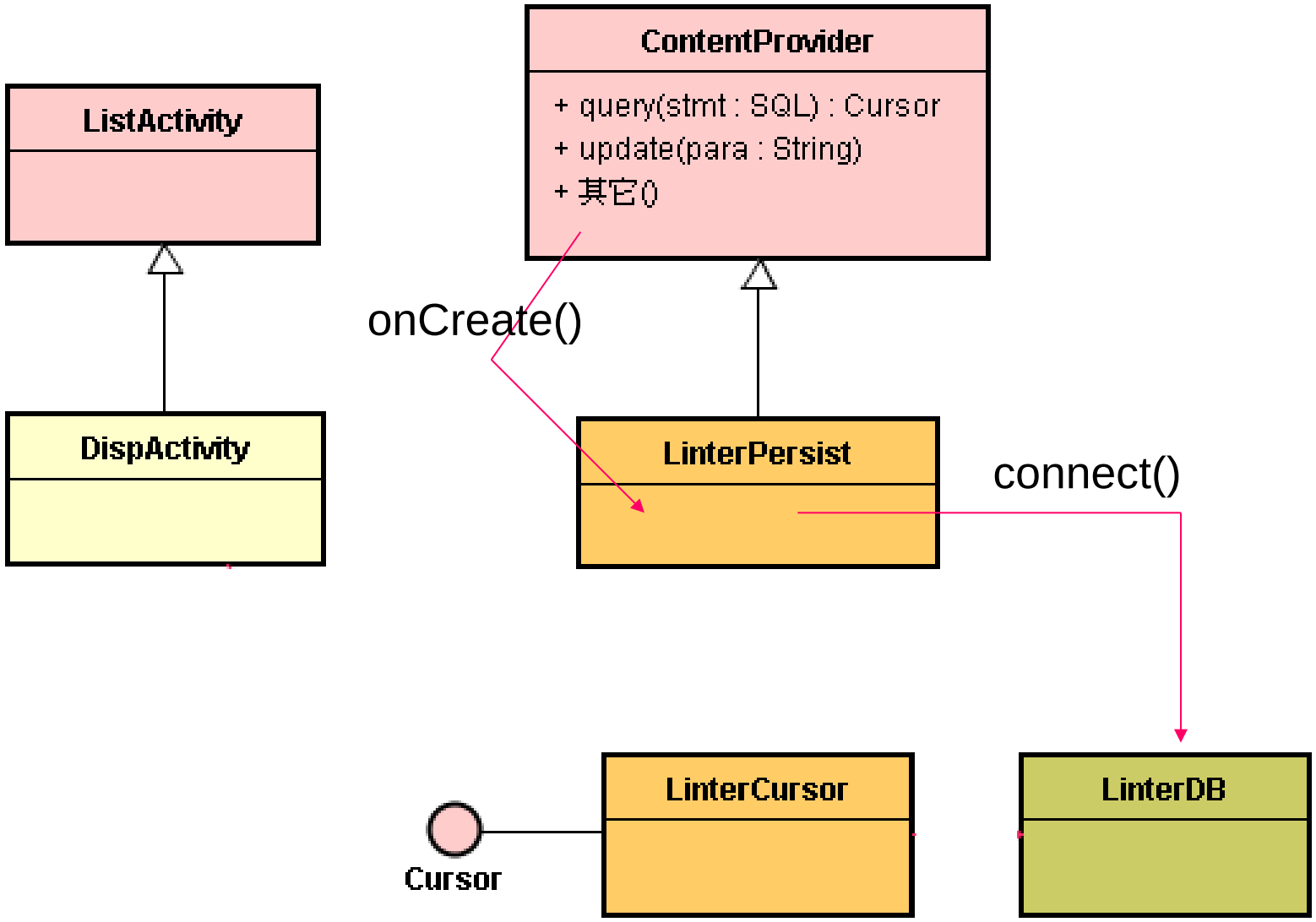
- 例如，我们先将Linter DB引擎安装到Android的Linux环境里，并建立JDBC存取通道；接着就我们能轻易地将Linter引擎整合到ContentProvider框架里。


ListActivity
ContentProvider
+ query(stmt : SQL) : Cursor
+ update(para : String)
+ 其它()
DispActivity
LinterPersist
LinterCursor
LinterDB
Cursor

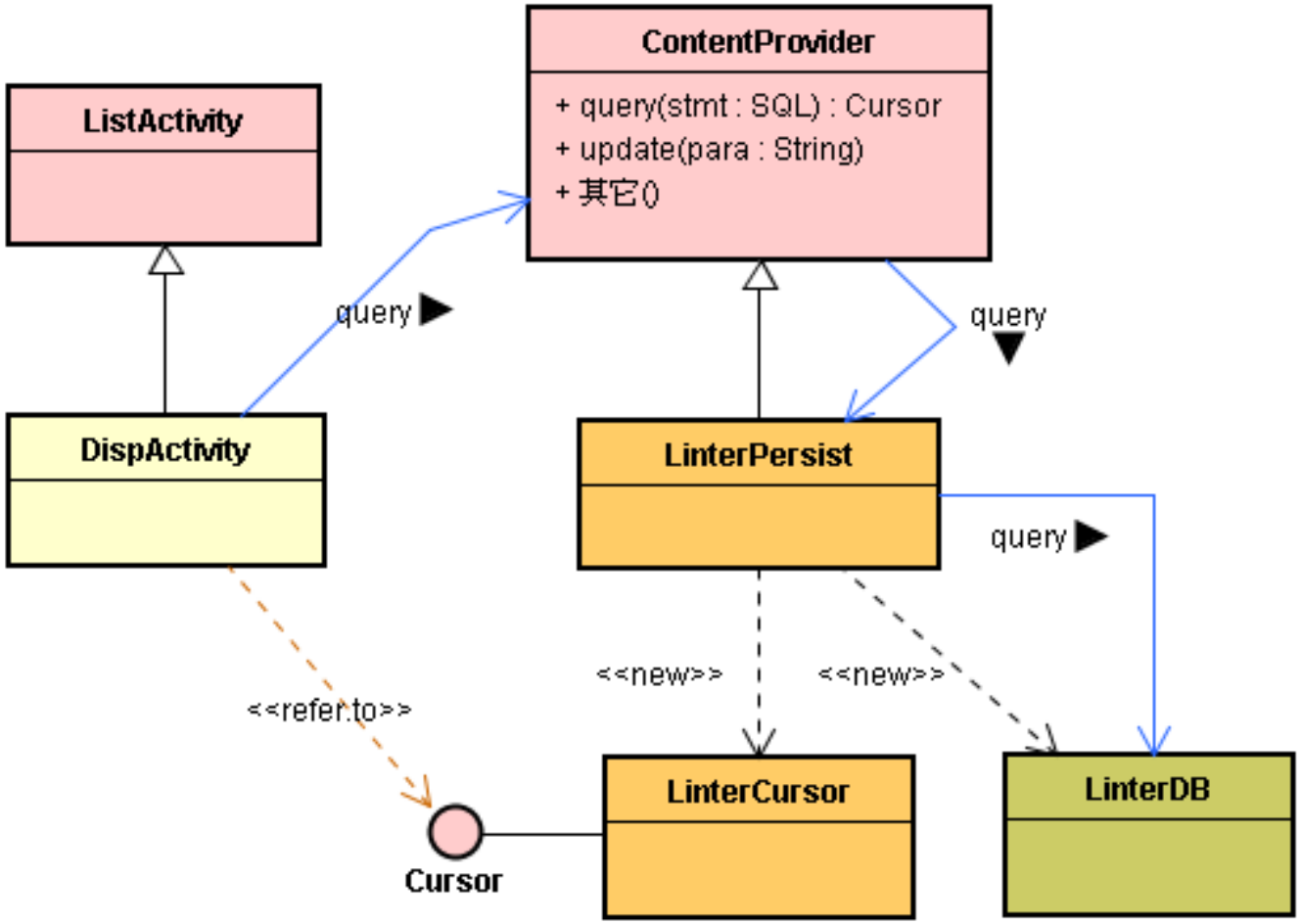

ListActivity
DispActivity
ContentProvider
+ query(stmt : SQL) : Cursor
+ update(para : String)
+ 其它()
LinterPersist
Cursor
LinterCursor
LinterDB

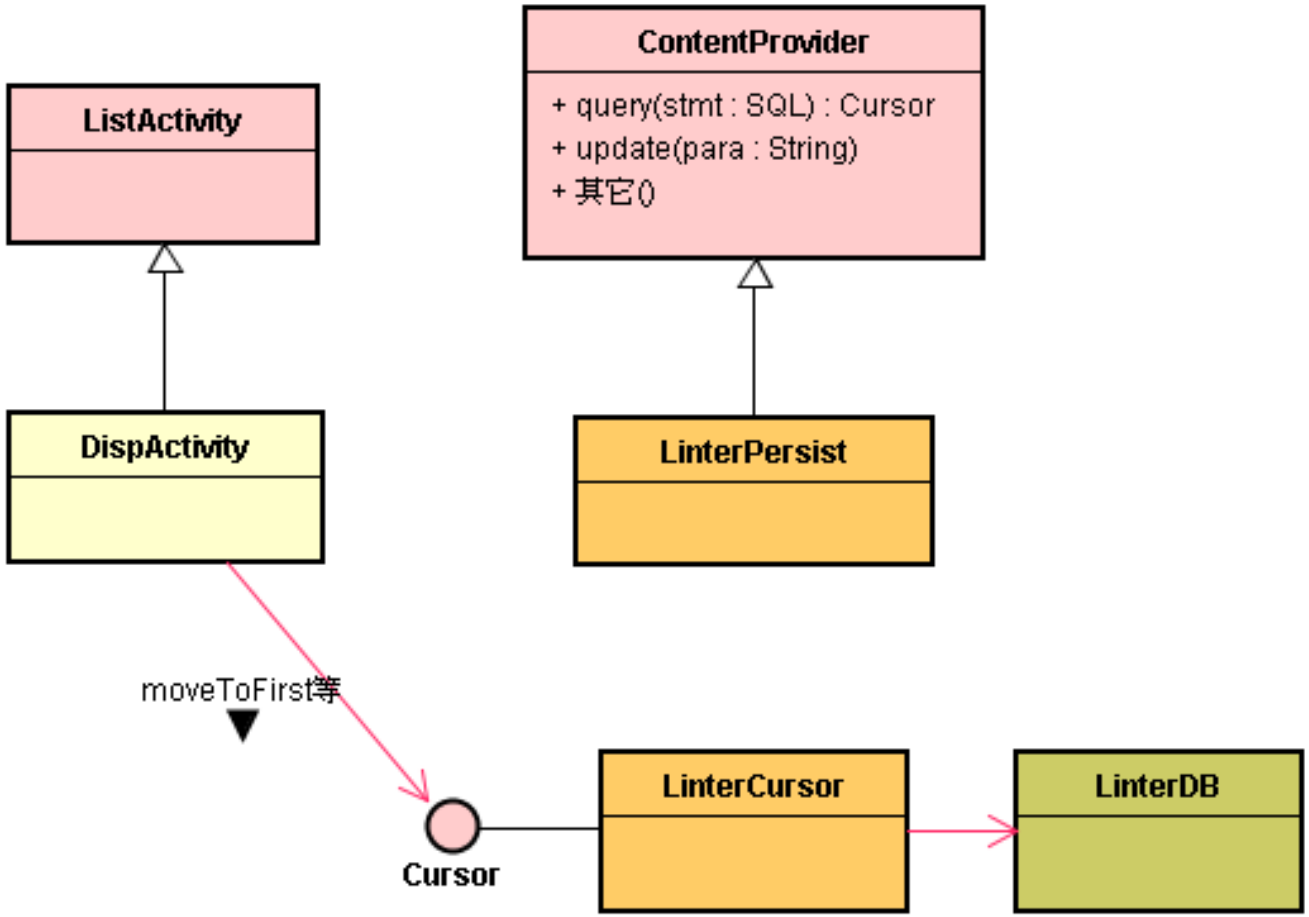
- 这包含了3个类：
  1. LinterPersist类
  2. LinterCursor类
  3. DispActivity类

- LinterPersist类实现query()接口，实际呼叫Linter数据库的功能。
- 也就是说，DispActivity透过ContentProvider接口呼叫到LinterPersist的query()函数。
- 此时创建一个LinterCursor对象，将其Cursor接口回传给DispActivity 。


ContentProvider
+ query(stmt : SQL) : Cursor
+ update(para : String)
+ 其它()
ListActivity
DispActivity
onCreate()
LinterPersist
connect()
LinterCursor
Cursor
LinterDB

- 此LinterPersist对象是由Android框架所创建的。
- 之后，Client就调用getContentResolver()函数来要求Android框架去进行配对和绑定此LinterPersist对象。然后间接绑定了Linter DB引擎。


ContentProvider
+ query(stmt : SQL) : Cursor
+ update(para : String)
+ 其它()
ListActivity
DispActivity
LinterPersist
LinterCursor
LinterDB
Cursor
query
query
query
<<new>>
<<new>>
<<refer to>>


ContentProvider
+ query(stmt : SQL) : Cursor
+ update(para : String)
+ 其它()
ListActivity
DispActivity
LinterPersist
moveToFirst等
Cursor
LinterCursor
LinterDB

Linter + Android 范例代码

- 于此，我们需要撰写3个类：

1. LinterPersist类
2. LinterCursor类
3. DispActivity类

撰写LinterPersist类代码

```
/* ----- LinterPersist.java 程序代码 ------*/
// ……...
public class LinterPersist extends ContentProvider {
        private static final String LINTER_TABLE_NAME
                        = "Student123";
        private Connection con;
```

```
@Override public boolean onCreate() {
        try {
      Class.forName("com.relx.jdbc.LinterDriver").newInstance();
      con = DriverManager.getConnection(
              "jdbc:Linter:linapid:localhost:1070:local", "SYSTEM",
                  "MANAGER");
      } catch(Exception e) { Log.e("Conn failed", e.toString());}
            return false;  }
      try{ Statement stmt = con.createStatement();
        stmt.executeUpdate("drop table " +
                                              LINTER_TABLE_NAME);
      } catch (Exception e)
             { Log.e("drop table failed", e.toString()); }
```

```
try { Statement stmt = con.createStatement();
            stmt.executeUpdate( "create table " +
                    LINTER_TABLE_NAME + " (stud_no char(10),
                                                stud_name char(20));");
       PreparedStatement prepstmt1 = con.prepareStatement(
            "insert into " + LINTER_TABLE_NAME + " values (?,?);");
      prepstmt1.setString(1, "linter_5");  prepstmt1.setString(2, "Lisa");
      prepstmt1.executeUpdate();
      prepstmt1.setString(1, "linter_8");  prepstmt1.setString(2, "Kitty");
      prepstmt1.executeUpdate( );
    } catch (Exception e){ Log.e("create/insert table failed",
                                e.toString());
      return false;  }
    return true;
  }
```

```
@Override public Cursor query(Uri uri, String[]
            projection, String selection,
            String[] selectionArgs, String sortOrder) {
    ResultSet rs = null;
    try { Statement stmt = con.createStatement();
            rs = stmt.executeQuery("select * from " +
                        LINTER_TABLE_NAME);
     } catch (Exception e) { e.printStackTrace();
            return null; }
    Cursor c = new LinterCursor(rs, con);
    return c;
  }
 @Override  public String getType(Uri uri)  {
    return null;
  }
```

```
@Override public Uri insert(Uri uri, ContentValues initialValues) {
        String field_1 = initialValues.get("stud_no").toString();
        String field_2 = initialValues.get("stud_name").toString();
        try{ PreparedStatement prepstmt1 = con.prepareStatement(
                  "insert into " + LINTER_TABLE_NAME + " values (?,?);");
            prepstmt1.setString(1, field_1);
            prepstmt1.setString(2, field_2);
            prepstmt1.executeUpdate();
        } catch (Exception e) { Log.e("ERROR", e.toString()); }
        return uri;
 }
@Override public int delete(Uri uri, String where, String[]
          whereArgs)  { return 0; }
@Override public int update(Uri uri, ContentValues values, String
          where,String[] whereArgs) {  return 0;  }
```

撰写LinterCursor类代码

```
//----------------  定义LinterCursor 类 -------------------------------
class LinterCursor implements Cursor{
    ResultSet res;  Connection conn;
    LinterCursor( ResultSet rs, Connection con)
            { res = rs;  conn = con;  }
    @Override  public void close() {
            try {  res.close();  conn.close();
            } catch (java.sql.SQLException e) { e.printStackTrace();  }
     }
```

```
@Override
 public void copyStringToBuffer(int columnIndex,
         CharArrayBuffer buffer) {}
 @Override public void deactivate() {}
 @Override public byte[] getBlob(int columnIndex) {  return null;  }
 @Override public int getColumnCount() {  return 0;  }
 @Override public int getColumnIndex(String columnName)
          {  return 0;  }
 @Override
 public int getColumnIndexOrThrow(String columnName)
                  throws IllegalArgumentException {  return 0;       }
 @Override public String getColumnName(int columnIndex)
          {  return null;    }
 @Override public String[] getColumnNames() { return null;  }
 @Override public int getCount() { return 0;       }
 @Override public double getDouble(int columnIndex)
           { return 0;  }
```

```
@Override public Bundle getExtras() {  return null;  }
@Override public float getFloat(int columnIndex) {       return 0;  }
@Override public int getInt(int columnIndex) {  return 0;  }
@Override public long getLong(int columnIndex) {       return 0; }
@Override public int getPosition() { return 0;    }
@Override public short getShort(int columnIndex) { return 0;  }
@Override public String getString(int columnIndex) {
        try { return res.getString(columnIndex + 1);
        } catch (java.sql.SQLException e) { e.printStackTrace(); }
        return null;   }
 @Override public boolean getWantsAllOnMoveCalls()
        { return false;  }
 @Override public boolean isAfterLast() {
        try { return res.isAfterLast();
        } catch (java.sql.SQLException e)
                        { e.printStackTrace(); return false; }}
```

```
@Override public boolean isBeforeFirst() {
            try {   return res.isBeforeFirst();
            } catch (java.sql.SQLException e)
                        {  e.printStackTrace(); return false; }}
@Override public boolean isClosed() { return false;  }
@Override public boolean isFirst() { return false;          }
@Override public boolean isLast() {return false;            }
@Override public boolean isNull(int columnIndex) {  return false; }
@Override public boolean move(int offset) {  return false;          }
@Override public boolean moveToFirst() {
        try { return res.first();
        } catch (java.sql.SQLException e)
                    { e.printStackTrace(); return false; }}
@Override public boolean moveToLast() {  return false;  }
@Override public boolean moveToNext() {
        try { return res.next();
        } catch (java.sql.SQLException e)
                    {e.printStackTrace(); return false; }}
```

```
  @Override public boolean moveToPosition(int position)
          {  return false;  }
  @Override public boolean moveToPrevious() {  return false;  }
  @Override public void registerContentObserver(
                          ContentObserver observer) {}
  @Override public void registerDataSetObserver(
                          DataSetObserver observer) {}
  @Override public boolean requery() {  return false;  }
  @Override public Bundle respond(Bundle extras) {  return null; }
  @Override public void setNotificationUri(
                          ContentResolver cr, Uri uri) {}
  @Override public void unregisterContentObserver(
                          ContentObserver observer) {}
  @Override public void unregisterDataSetObserver(
                          DataSetObserver observer) {}
}}
```

- LinterPersist类里的指令：

```
@Override public Cursor query(Uri uri, String[] projection,
        String selection,  String[] selectionArgs, String sortOrder)
  {
     ResultSet rs = null;
     try { Statement stmt = con.createStatement();
           rs = stmt.executeQuery("select * from " +
                                          LINTER_TABLE_NAME);
     } catch (Exception e)
          { e.printStackTrace();  return null; }
    Cursor c = new LinterCursor(rs, con);
    return c;
  }
```

- 此函数呼叫了Linter的executeQuery()函数，要求Linter进行数据库的查询任务。Linter回传一个ResultSet对象，让应用程序可浏览所查询到的各笔数据。在这query()函数里，就诞生一个LinterCursor对象，让它内含该ResultSet对象，然后将此LinterCursor对象回传给DispActivity应用类。

- 于是，顺利地将Linter数据库配上ContentProvider接口，飞上枝头变凤凰，成为Android嫡系成员。
- 终于完成我们的目：让远从万里之外的Linter舶来组件，顺利融入(移植到)Android之中，成为其嫡系成员之一

撰写DispActivity类代码

```
/* ----- DispActivity.java  -----*/
// .............
public class DispActivity extends ListActivity {
    public static final String AUTHORITY
        = "com.misoo.provider.rx09-04";
    public static final Uri CONTENT_URI
        = Uri.parse("content://" + AUTHORITY +
                    /"/Stud202");
    private static final String[] PROJECTION
        =  new String[] { "stud_no", "stud_name"  };
```

```
@Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Intent intent = getIntent();
        if (intent.getData() == null) {  intent.setData(CONTENT_URI);  }
        Cursor cursor = getContentResolver().query(
                getIntent().getData(),PROJECTION, null, null, null);
        if(cursor == null)     return;
        ArrayList<Map<String, Object>> coll
                = new ArrayList<Map<String, Object>>();
        Map<String, Object> item;
        cursor.moveToFirst();
```

```
            while (!cursor.isAfterLast()) {
                item = new HashMap<String, Object>();
                item.put("c1", cursor.getString(0) + ",  " +
                                        cursor.getString(1));
                coll.add(item);
                cursor.moveToNext();
        }
        this.setListAdapter(new SimpleAdapter(this, coll,
            android.R.layout.simple_list_item_1, new String[] { "c1" },
            new int[] {android.R.id.text1}));
          }
   @Override
   protected void onListItemClick(ListView l, View v, int position,
                        long id)
          { finish(); }
}
```

- 指令：

```
Cursor cursor = getContentResolver().query(
    getIntent().getData(),PROJECTION, null, null, null);
```

- 透过getContentResolver()而要求Android寻找适当的ContentProvider实作类(如本范例的LinterPersist)，并呼叫其query()函数，以进行数据查询的任务。

- 其查询出多笔的数据，查询之后，它会传回数据指针值(Record pointer)给LinterCursor，并将LinterCursor的Cursor接口回传给DispActivity程序。

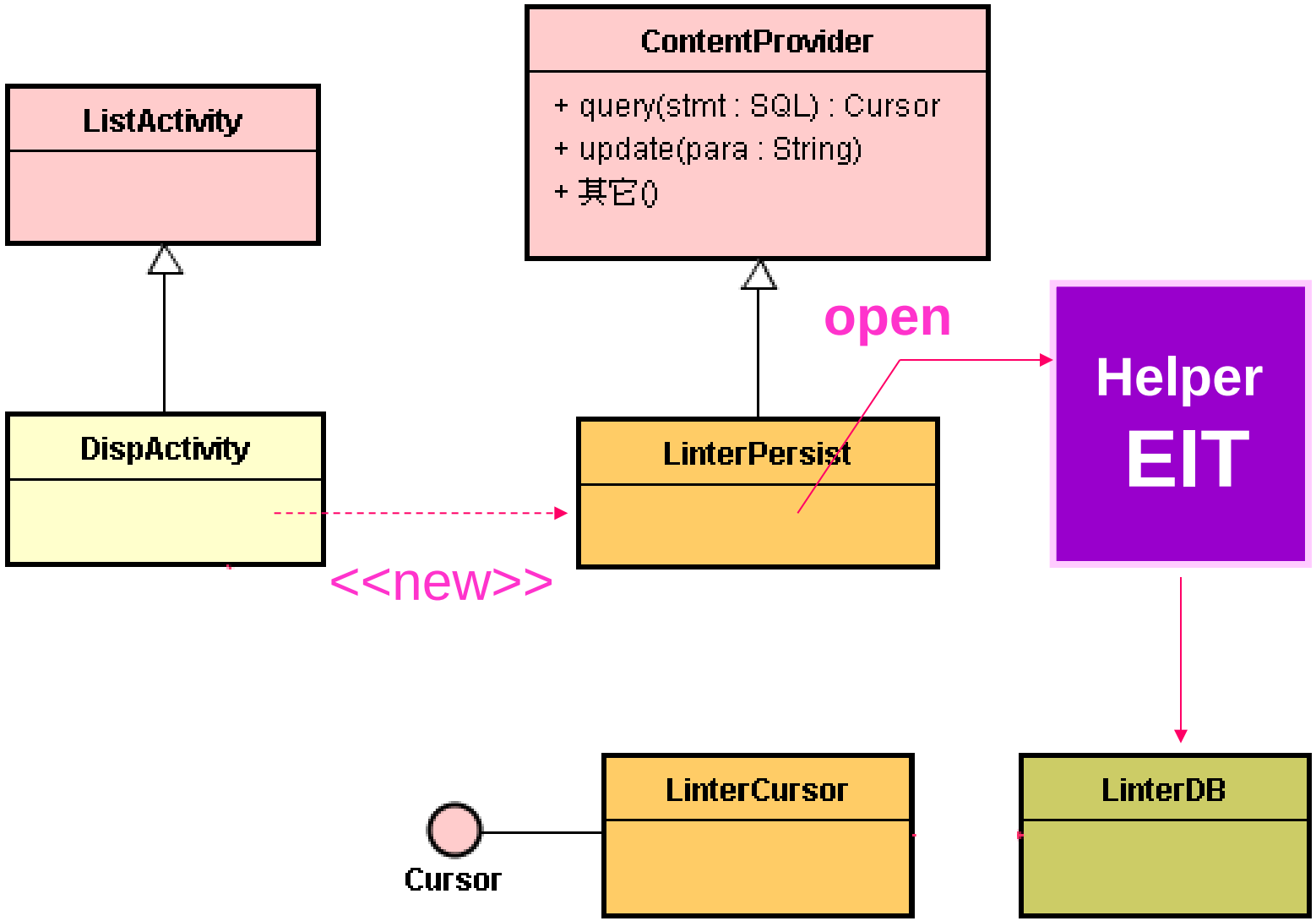
- 由于在LinterPersist类里，我们定义了一个LinterCursor类，所以这指令所传回来的cursor是参考(Reference)到LinterCursor对象。
- 我们把原来的 Cursor实作类抽换掉了，换为新的LinterCursor类，这些应用类(如DispActivity等)则丝毫不受影响，表现出高度的抽换性。

- 接下来，DispActivity只要调用Cursor接口的函数，就能浏览Linter DB里的内容了。
- 例如指令：

```
cursor.moveToFirst();
```

- 就将DB的数据指针值(Record pointer)移到最前头。

也能提供Linter DB
特殊性的Helper EIT


ContentProvider
+ query(stmt : SQL) : Cursor
+ update(para : String)
+ 其它()
ListActivity
DispActivity
LinterPersist
open
Helper
EIT
<<new>>
LinterCursor
Cursor
LinterDB

Thanks...

高煥堂