Lifecycle本身作为抽象类，提供给外部使用，在实现类Activity中通过调用getLifecycle()返回LifecycleRegistry对象，该对象在ComponentActivity中作为成员变量实现并初始化，具体实现代码如下图所示：

private final LifecycleRegistry mLifecycleRegistry = new LifecycleRegistry(this);

@NonNull

@Override

public Lifecycle getLifecycle() {

return mLifecycleRegistry;

}

Lifecycle提供给外部使用的方式大致有如下几种：

①以自定义View为例,首先实现接口LifecycleEventObserver，其继承LifecycleObserver接口:

LifecycleEventObserver lifecycleEventObserver = new LifecycleEventObserver() {

@Override

public void onStateChanged(@NonNull LifecycleOwner source, @NonNull Lifecycle.Event event) {

switch (event.ordinal()) {

case 0:

onCreate();

break;

case 2:

onResume();

break;

case 3:

onPause();

break;

case 5:

onDestroy();

break;

}

}

};

上述接口根据返回的Lifecycle.Event枚举类的对应项来进行判断实现对应的生命周期实现，枚举类Lifecycle.Event对应项如下所示:

ON\_CREATE,

ON\_START,

ON\_RESUME,

ON\_PAUSE,

ON\_STOP,

ON\_DESTROY,

ON\_ANY;

②通过LifecycleEvent实现，案例如下所示：

@OnLifecycleEvent(value = Lifecycle.Event.ON\_CREATE)

public void onCreate() {

Log.i(TAG, "onCreate");

}

接下来就是添加观察者LifecycleObserver操作，①②对象实现方式如下所示：

①处理方式如下:

if (null != lifecycle) {

lifecycle.addObserver(lifecycleEventObserver);

}

②处理方式如下：

if (null != lifecycle) {

lifecycle.addObserver(this);

}

那么this对象，可作为当前类实现接口LifecycleObserver即可。