Android常见面试题

- 下面异常是属于Runtime Exception 的是(abcd)(多选)

A、ArithmeticException　　B、IllegalArgumentException　　C、NullPointerException　　D、BufferUnderflowException

解析： Java提供了两类主要的异常:runtime exception和checked exception。checked 异常也就是我们经常遇到的IO异常，以及SQL异常都是这种异常。对于这种异常，JAVA编译器强制要求我们必需对出现的这些异常进行catch。所以，面对这种异常不管我们是否愿意，只能自己去写一大堆catch块去处理可能的异常。。。

出现运行时异常后，系统会把异常一直往上层抛，一直遇到处理代码。如果没有处理块，到最上层，如果是多线程就由Thread.run()抛出，如果是单线程就被main()抛出。抛出之后，如果是线程，这个线程也就退出了。如果是主程序抛出的异常，那么这整个程序也就退出了。运行时异常是Exception的子类，也有一般异常的特点，是可以被Catch块处理的。只不过往往我们不对他处理罢了。也就是说，你如果不对运行时异常进行处理，那么出现运行时异常之后，要么是线程中止，要么是主程序终止。

编译时被检查的异常和运行时异常的区别：　编译被检查的异常在函数内被抛出，函数必须要声明，否编译失败。　声明的原因：是需要调用者对该异常进行处理。　运行时异常如果在函数内被抛出，在函数上不需要声明。

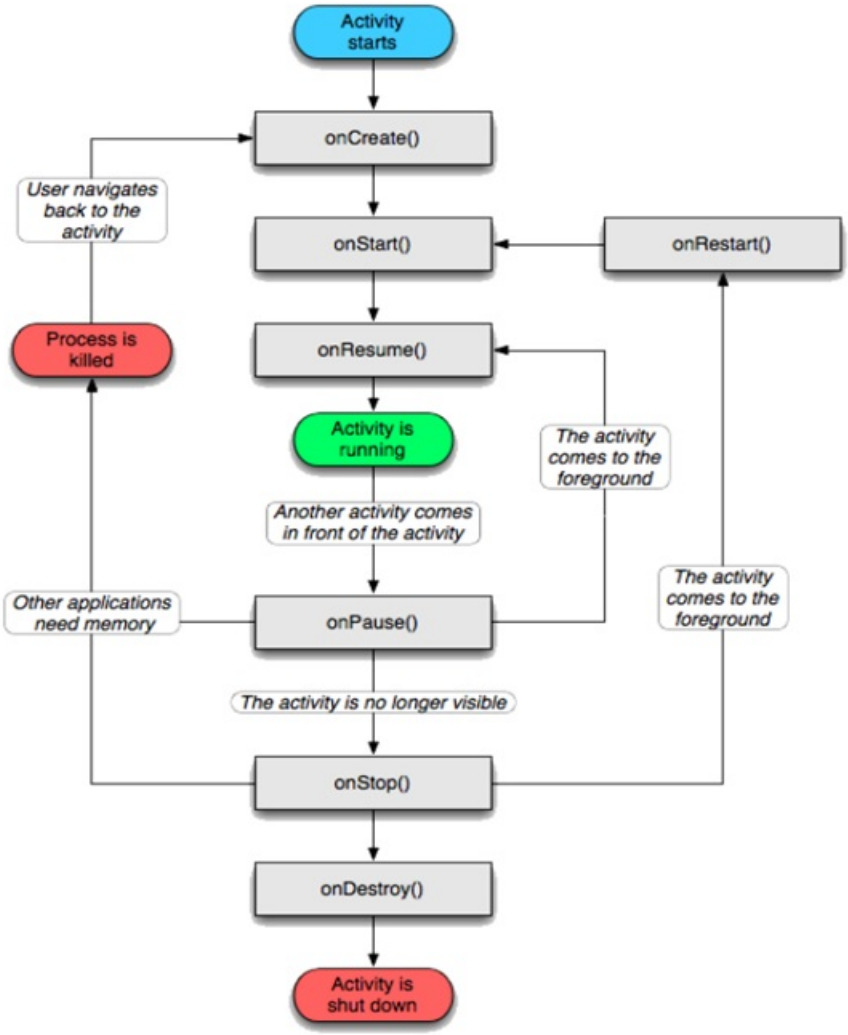
- Math.round(11.5)等于多少(). Math.round(-11.5)等于多少(c)

  A、11 ,-11 B、11 ,-12 C、12 ,-11 D、12 ,-12

  解析： Math.ceil()用作向上取整。 Math.floor()用作向下取整。 Math.round() 我们数学中常用到的四舍五入取整。

- 对一些资源以及状态的操作保存，最好是保存在生命周期的哪个函数中进行(d)

  A、onPause() B、onCreate() C、 onResume() D、onStart()



解析：

系统杀死程序会调用onSaveInstanceState(Bundle)进行数据保存，这里保存的数据会出现在在程序下一次onStart（Bundle）这个Bundle中，onStart时可以将Bundle中数据取出。

- Intent传递数据时，下列的数据类型哪些可以被传递(abcd)(多选)

A、Serializable B、charsequence C、Parcelable D、Bundle

解析：

Serializable :将 Java 对象序列化为二进制文件的 Java 序列化技术是 Java系列技术中一个较为重要的技术点，在大部分情况下，开发人员只需要了解被序列化的类需要实现 Serializable 接口，使用ObjectInputStream 和 ObjectOutputStream 进行对象的读写。

charsequence : 实现了这个接口的类有：CharBuffer、String、StringBuffer、StringBuilder这个四个类。 CharBuffer为nio里面用的一个类，String实现这个接口理所当然，StringBuffer也是一个CharSequence，StringBuilder是Java抄袭C#的一个类，基本和StringBuffer类一样，效率高，但是不保证线程安全，在不需要多线程的环境下可以考虑。 提供这么一个接口，有些处理String或者StringBuffer的类就不用重载了。但是这个接口提供的方法有限，只有下面几个：charat、length、subSequence、toString这几个方法，感觉如果有必要，还是重载的比较好，避免用instaneof这个操作符。

Parcelable : android提供了一种新的类型：Parcel。本类被用作封装数据的容器，封装后的数据可以通过Intent或IPC传递。 除了基本类型以 外，只有实现了Parcelable接口的类才能被放入Parcel中。 是GOOGLE在安卓中实现的另一种序列化,功能和Serializable相似,主要是序列化的方式不同 Bundle是将数据传递到另一个上下文中或保存或回复你自己状态的数据存储方式。它的数据不是持久化状态。

- 下列属于SAX解析xml文件的优点的是(b)

  A、将整个文档树在内存中，便于操作，支持删除，修改，重新排列等多种功能 B、不用事先调入整个文档，占用资源少 C、整个文档调入内存，浪费时间和空间 D、不是长久驻留在内存，数据不是持久的，事件过后，若没有保存数据，数据就会消失

解析： 在Android中提供了三种解析XML的方式:SAX(Simple API XML),DOM(Document Objrect Model),以及Android推荐的Pull解析方式。

SAX： 是事件驱动型XML解析的一个标准接口，简单地说就是对文档进行顺序扫描，当扫描到文档（document）开始与结束、元素（element）开始与结束、文档（document）结束等地方时通知事件处理函数，由事件处理函数做相应动作，然后继续同样的扫描，直至文档结束。

DOM：即对象文档模型，它是将整个XML文档载入内存(所以效率较低，不推荐使用)，使用DOM API遍历XML树、检索所需的数据，每一个节点当做一个对象。

Pull：运行方式与 SAX 解析器相似。它提供了类似的事件，SAX解析器的工作方式是自动将事件推入事件处理器进行处理，因此你不能控制事件的处理主动结束；而Pull解析器的工作方式为允许你的应用程序代码主动从解析器中获取事件，正因为是主动获取事件，因此可以在满足了需要的条件后不再获取事件，结束解析。pull是一个while循环，随时可以跳出，而sax是只要解析了，就必须解析完成。

- 在android中使用Menu时可能需要重写的方法有(ac)。(多选)

  A、onCreateOptionsMenu() B、onCreateMenu() C、onOptionsItemSelected() D、onItemSelected()

  解析： android中有三种菜单

  1.选项菜单Options menus :一个Activity只能有一个选项菜单，在按下Menu键时，显示在屏幕下方。

  ```
  重写 onCreateContextMenu 用以创建上下文菜单
  重写 onContextItemSelected 用以响应上下文菜单
  ```

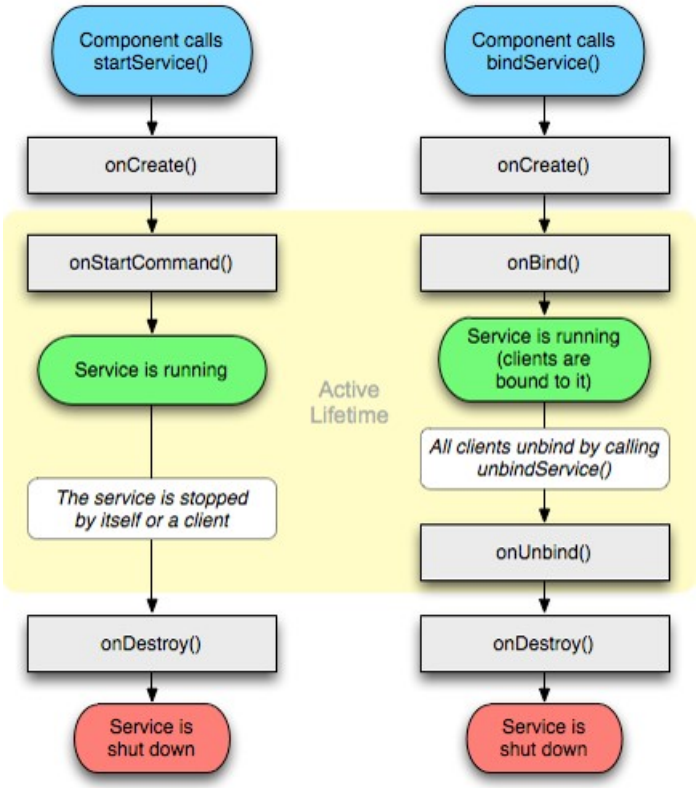
  2.上下文菜单Context menus :为Activity中的任何一个视图注册一个上下文菜单，“长按”出现。

  ```
  重写 onCreateOptionsMenu 用以创建选项菜单
  重写 onOptionsItemSelected 用以响应选项菜单
  ```

  3.弹出式菜单Popup menus :依赖于Activity中的某个一个视图

- android 关于service生命周期的onCreate()和onStart()说法正确的是(ad)(多选题)

  A、当第一次启动的时候先后调用onCreate()和onStart()方法 B、当第一次启动的时候只会调用onCreate()方法 C、如果service已经启动，将先后调用onCreate()和onStart()方法 D、如果service已经启动，只会执行onStart()方法，不在执行onCreate()方法



解析：

1). 被启动的服务的生命周期：如果一个Service被某个Activity 调用 Context.startService 方法启动，那么不管是否有Activity使用bindService绑定或unbindService解除绑定到该Service，该Service都在后台运行。如果一个Service被startService 方法多次启动，那么onCreate方法只会调用一次，onStart将会被调用多次（对应调用startService的次数），并且系统只会创建Service的一个实例（因此你应该知道只需要一次stopService调用）。该Service将会一直在后台运行，而不管对应程序的Activity是否在运行，直到被调用stopService，或自身的stopSelf方法。当然如果系统资源不足，android系统也可能结束服务。

2). 被绑定的服务的生命周期：如果一个Service被某个Activity 调用 Context.bindService 方法绑定启动，不管调用 bindService 调用几次，onCreate方法都只会调用一次，同时onStart方法始终不会被调用。当连接建立之后，Service将会一直运行，除非调用Context.unbindService 断开连接或者之前调用bindService 的 Context 不存在了（如Activity被finish的时候），系统将会自动停止Service，对应onDestroy将被调用。

3). 被启动又被绑定的服务的生命周期：如果一个Service又被启动又被绑定，则该Service将会一直在后台运行。并且不管如何调用，onCreate始终只会调用一次，对应startService调用多少次，Service的onStart便会调用多少次。调用unbindService将不会停止Service，而必须调用 stopService 或 Service的 stopSelf 来停止服务。 4). 当服务被停止时清除服务：当一个Service被终止（1、调用stopService；2、调用stopSelf；3、不再有绑定的连接（没有被启动））时，onDestroy方法将会被调用，在这里你应当做一些清除工作，如停止在Service中创建并运行的线程。 特别注意：

1、你应当知道在调用 bindService 绑定到Service的时候，你就应当保证在某处调用 unbindService 解除绑定（尽管 Activity 被 finish 的时候绑定会自动解除，并且Service会自动停止）； 2、你应当注意 使用 startService 启动服务之后，一定要使用 stopService停止服务，不管你是否使用bindService； 3、同时使用 startService 与 bindService 要注意到，Service 的终止，需要unbindService与stopService同时调用，才能终止 Service，不管 startService 与 bindService 的调用顺序，如果先调用 unbindService 此时服务不会自动终止，再调用 stopService 之后服务才会停止，如果先调用 stopService 此时服务也不会终止，而再调用 unbindService 或者 之前调用 bindService 的 Context 不存在了（如Activity 被 finish 的时候）之后服务才会自动停止； 4、当在旋转手机屏幕的时候，当手机屏幕在“横”“竖”变换时，此时如果你的 Activity 如果会自动旋转的话，旋转其实是 Activity 的重新创建，因此旋转之前的使用bindService 建立的连接便会断开（Context 不存在了），对应服务的生命周期与上述相同。 5、在 sdk 2.0 及其以后的版本中，对应的 onStart 已经被否决变为了 onStartCommand，不过之前的 onStart 任然有效。这意味着，如果你开发的应用程序用的 sdk 为 2.0 及其以后的版本，那么你应当使用 onStartCommand 而不是 onStart。

- 下面是属于GLSurFaceView特性的是(abc)(多选)

A、管理一个surface，这个surface就是一块特殊的内存，能直接排版到android的视图view上。 B、管理一个EGL display，它能让

opengl把内容渲染到上述的surface上。 C、让渲染器在独立的线程里运作，和UI线程分离。 D、可以直接从内存或者DMA等硬件接口取得图像数据

解析： Android游戏当中主要的除了控制类外就是显示类View。SurfaceView是从View基类中派生出来的显示类。android游戏开发中常用的三种视图是：view、SurfaceView和GLSurfaceView。

View：显示视图，内置画布，提供图形绘制函数、触屏事件、按键事件函数等；必须在UI主线程内更新画面，速度较慢。

SurfaceView：基于view视图进行拓展的视图类，更适合2D游戏的开发；是view的子类，类似使用双缓机制，在新的线程中更新画面所以刷新界面速度比view快。

GLSurfaceView：基于SurfaceView视图再次进行拓展的视图类，专用于3D游戏开发的视图；是SurfaceView的子类，openGL专用。

GLSurfaceView提供了下列特性： 1.管理一个surface，这个surface就是一块特殊的内存，能直接排版到android的视图view上。 2.管理一个EGL display，它能让opengl把内容渲染到上述的surface上。 3.用户自定义渲染器(render)。 4 . 让渲染器在独立的线程里运作，和UI线程分离。 5.支持按需渲染(on-demand)和连续渲染(continuous)。 6.一些可选工具，如调试。

- 关于ContenValues类说法正确的是(a)

A、他和Hashtable比较类似，也是负责存储一些名值对，但是他存储的名值对当中的名是String类型，而值都是基本类型

B、他和Hashtable比较类似，也是负责存储一些名值对，但是他存储的名值对当中的名是任意类型，而值都是基本类型

C、他和Hashtable比较类似，也是负责存储一些名值对，但是他存储的名值对当中的名，可以为空，而值都是String类型

D、他和Hashtable比较类似，也是负责存储一些名值对，但是他存储的名值对当中的名是String类型，而值也是String类型

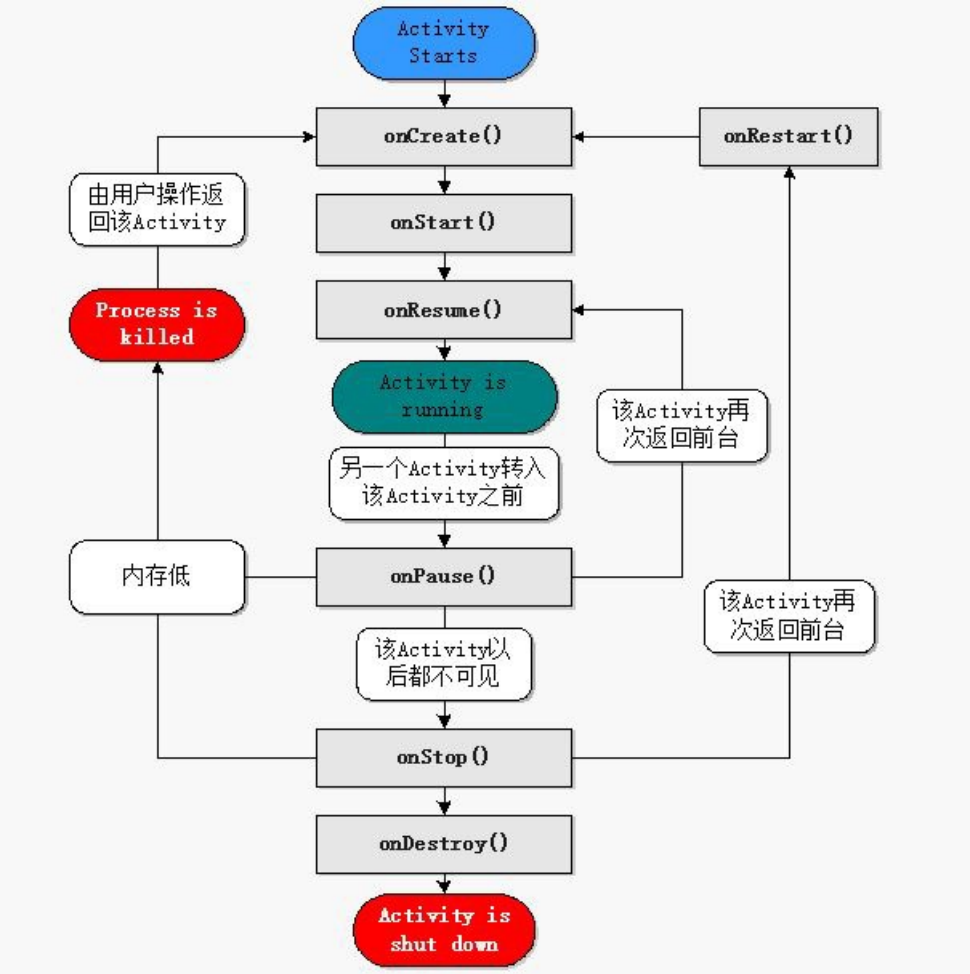
解析： ContentValues 和HashTable类似都是一种存储的机制 但是两者最大的区别就在于，contenvalues Key只能是String类型，values只能存储基本类型的数据，像string，int之类的，不能存储对象这种东西。ContentValues 常用在数据库中的操作。

HashMap是Hashtable的轻量级实现（非线程安全的实现），他们都完成了Map接口，主要区别在于HashMap允许空（null）键值（key）,由于非线程安全，效率上可能高于Hashtable。HashMap允许将null作为一个entry的key或者value，而Hashtable不允许。

- 下面退出Activity错误的方法是(d)

A、finish() B、抛异常强制退出 C、System.exit() D、onStop()

解析：



finish()：在你的activity动作完成的时候，或者Activity需要关闭的时候，调用此方法。当你调用此方法的时候，系统只是将最上面的Activity移出了栈，并没有及时的调用onDestory（）方法，其占用的资源也没有被及时释放。因为移出了栈，所以当你点击手机上面的“back”按键的时候，也不会再找到这个Activity。finish函数仅仅把当前Activity退出了，但是并没有释放他的资源。安卓系统自己决定何时从内存中释放应用程序。当系统没有可用内存到时候，会按照优先级，释放部分应用。

onDestory()：系统销毁了这个Activity的实例在内存中占据的空间。 在Activity的生命周期中，onDestory()方法是他生命的最后一步，资源空间等就被回收了。当重新进入此Activity的时候，必须重新创建，执行onCreate()方法。

System.exit(0)：退出整个应用程序（不仅仅是当前activity）。将整个进程直接Kill掉。

- 关于res/raw目录说法正确的是(a)

A、 这里的文件是原封不动的存储到设备上不会转换为二进制的格式 B、这里的文件是原封不动的存储到设备上会转换为二进制的格式 C、 这里的文件最终以二进制的格式存储到指定的包中 D、这里的文件最终不会以二进制的格式存储到指定的包中

```
解析：
res/raw和assets的相同点：
两者目录下的文件在打包后会原封不动的保存在apk包中，不会被编译成二进制。

res/raw和assets的不同点：
1.res/raw中的文件会被映射到R.java文件中，访问的时候直接使用资源ID即R.id.filename；assets文件夹下的文件不会被映射到R.
```

- android中常用的四个布局是framlayout，linenarlayout，relativelayout和tablelayout。
- android 的四大组件是activiey，service，broadcast和contentprovider。
- activity一般会重载7个方法用来维护其生命周期，除了onCreate(),onStart(),onDestory() 外还有onpause,onresume,onstop，onrestart。
- android的数据存储的方式sharedpreference,文件,SQlite,contentprovider,网络。

- 程序运行的结果是：good and gbc

```
public classExample{
   String str=new String("good");
   char[]ch={'a','b','c'};
   public static void main(String args[]){
   Example ex=new Example();
   ex.change(ex.str,ex.ch);
   System.out.print(ex.str+" and ");
   Sytem.out.print(ex.ch);
   }
   public void change(String str,char ch[]){
   str="test ok";
   ch[0]='g';
   }
   }
```

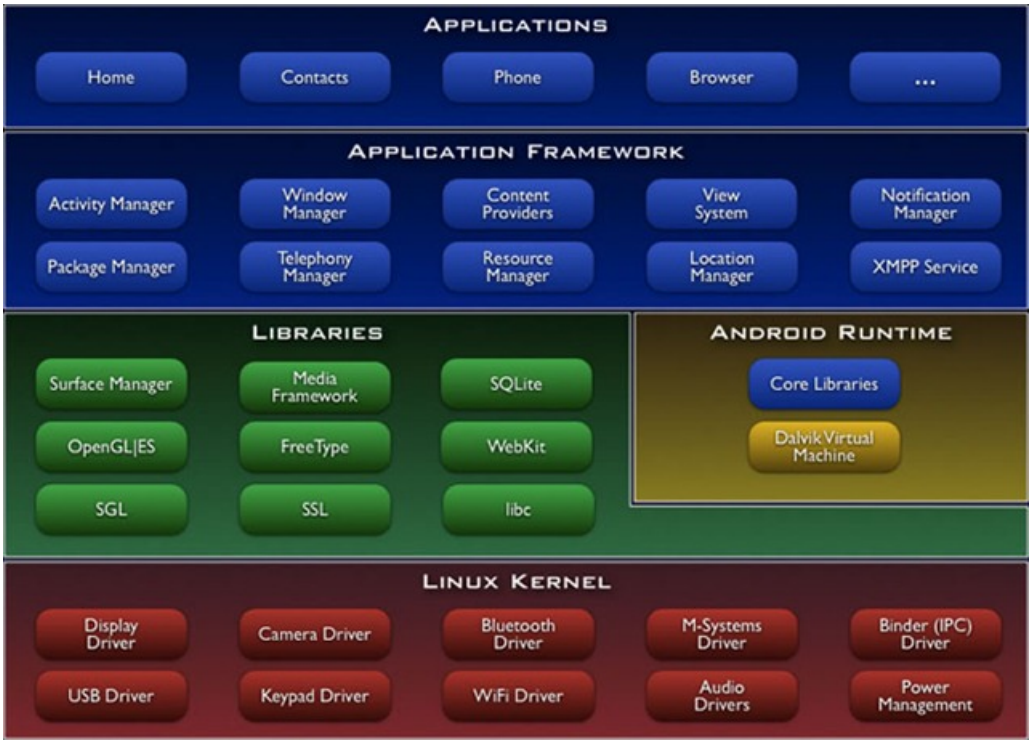
```
解析：
public void change(String str,char ch[])
str是按值传递，所以在函数中对它的操作只生效于它的副本，与原字符串无关。
ch是按址传递，在函数中根据地址，可以直接对字符串进行操作。
```

- 在android中，请简述jni的调用过程。

```
1)安装和下载Cygwin，下载 Android NDK
2)在ndk项目中JNI接口的设计
3)使用C/C++实现本地方法
4)JNI生成动态链接库.so文件
5)将动态链接库复制到java工程，在java工程中调用，运行java工程即可
```

- Android应用程序结构：

  Linux Kernel(Linux内核)、Libraries(系统运行库或者是c/c++核心库)、Application Framework(开发框架包)、Applications (核心应用程序)



- 请继承SQLiteOpenHelper实现创建一个版本为1的“diaryOpenHelper.db”的数据库，同时创建一个 “diary” 表(包含一个_id主键并自增长，topic字符型100长度， content字符型1000长度)，在数据库版本变化时请删除diary表，并重新创建出diary表。

```
public class DBHelper extends SQLiteOpenHelper{
    public final static String DATABASENAME ="diaryOpenHelper.db";
    public final static int DATABASEVERSION =1;
    //创建数据库
    public DBHelper(Context context,Stringname,CursorFactory factory,int version)
    {
    super(context, name, factory,version);
    }
    //创建表等机构性文件
    public void onCreate(SQLiteDatabase db)
    {
    String sql ="create tablediary"+
    "("+
    "_idinteger primary key autoincrement,"+
    "topicvarchar(100),"+
    "contentvarchar(1000)"+
    ")";
    db.execSQL(sql);
    }
    //若数据库版本有更新，则调用此方法
    public void onUpgrade(SQLiteDatabasedb,int oldVersion,int newVersion)
    {
    String sql = "drop table ifexists diary";
    db.execSQL(sql);
    this.onCreate(db);
    }
    }
```

- 页面上现有ProgressBar控件progressBar，请用书写线程以10秒的的时间完成其进度显示工作。

```
public class ProgressBarStu extends Activity {

    private ProgressBar progressBar = null;
    protected void onCreate(BundlesavedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.progressbar);
    //从这到下是关键
    progressBar = (ProgressBar)findViewById(R.id.progressBar);
    Thread thread = new Thread(newRunnable() {
    @Override
    public void run() {
    int progressBarMax =progressBar.getMax();
    try {
    while(progressBarMax!=progressBar.getProgress())
    {
    intstepProgress = progressBarMax/10;
    intcurrentprogress = progressBar.getProgress();
    progressBar.setProgress(currentprogress+stepProgress);
    Thread.sleep(1000);
    }
    } catch(InterruptedException e) {
    // TODO Auto-generatedcatch block
    e.printStackTrace();
    }
    }
    });
    thread.start();
    //关键结束
    }
    }
```

- onFreeze() renamed to onSaveInstanceState()，以便恢复在onCreate(Bundle)里面设置的状态。
- 如果后台的Activity由于某原因被系统回收了，onSaveInstanceState()在被系统回收之前（onPause()前面）保存当前状态。

  当你的程序中某一个Activity A在运行时，主动或被动地运行另一个新的Activity B，这个时候A会执行onSaveInstanceState()。B完成以后又会来找A，这个时候就有两种情况：一是A被回收，二是A没有被回收，被回收的A就要重新调用onCreate()方法，不同于直接启动的是这回onCreate()里是带上了参数savedInstanceState;而没被收回的就直接执行onResume()，跳过onCreate()了。

- ContentProvider：

  提供了我们在应用程序之前共享数据的一种机制，而我们知道每一个应用程序都是运行在不同的应用程序的，数据和文件在不同应用程序之间达到数据的共享不是没有可能，而是显得比较复杂，而正好Android中的ContentProvider则达到了这一需求，比如有时候我们需要操作手机里的联系人，手机里的多媒体等一些信息，我们都可以用到这个ContentProvider来达到我们所需。

1）、ContentProvider为存储和获取数据提供了统一的接口。ContentProvide对数据进行封装，不用关心数据存储的细节。使用表的形式来组织数据。 2）、使用ContentProvider可以在不同的应用程序之间共享数据。 3）、Android为常见的一些数据提供了默认的ContentProvider（包括音频、视频、图片和通讯录等）。 总的来说使用ContentProvider对外共享数据的好处是统一了数据的访问方式。

Uri为系统的每一个资源给其一个名字，比方说通话记录。每一个ContentProvider都拥有一个公共的URI，这个URI用于表示这个ContentProvider所提供的数据。

- 请解释下Android程序运行时权限与文件系统权限的区别。

```
运行时权限Dalvik( android授权)
文件系统 linux 内核授权
```

- 什么是ANR 如何避免它?

```
在Android里，应用程序的响应性是由Activity Manager和Window Manager系统服务监视的。当它监测到以下情况中的一个时，

在5秒内没有响应输入的事件（例如，按键按下，屏幕触摸）
BroadcastReceiver在10秒内没有执行完毕。

Android应用程序通常是运行在一个单独的线程（例如，main）里。这意味着你的应用程序所做的事情如果在主线程里占用了太长的时

在主线程里尽量的少做事情，比如高耗时的计算和网络、数据库等潜在的耗时操作都应该放在子线程来完成。
```