Android基础面试题

没有删这套题，虽然都是网上找的，在刚开始找工作的时候这套题帮了我很多，那时候`Android`刚起步，很多家都是这一套面试题，我都是直接去了不看题画画一顿就写完了，哈哈
现在估计没有公司会用这种笔试题了。还是留下来吧，回忆一下。

1. 下列哪些语句关于内存回收的说明是正确的? (b)
   A、 程序员必须创建一个线程来释放内存
   B、 内存回收程序负责释放无用内存
   C、 内存回收程序允许程序员直接释放内存
   D、 内存回收程序可以在指定的时间释放内存对象
2. 下面异常是属于Runtime Exception 的是（abcd）(多选)
   A、ArithmeticException
   B、IllegalArgumentException
   C、NullPointerException
   D、BufferUnderflowException
3. 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
4. 下列程序段的输出结果是：(b)

```
void complicatedexpression_r(){
    int x=20, y=30;
    Boolean b;
    b=x>50&&y>60||x>50&&y<-60||x<-50&&y>60||x<-50&&y<-60;
    System.out.println(b);
}
```

   A、true　B、false　C、1　D、011.activity
5. 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()
6. 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
7. android 中下列属于Intent的作用的是(c)
   A、实现应用程序间的数据共享
   B、是一段长的生命周期，没有用户界面的程序，可以保持应用在后台运行，而不会因为切换页面而消失
   C、可以实现界面间的切换，可以包含动作和动作数据，连接四大组件的纽带
   D、处理一个应用程序整体性的工作
8. 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
9. 下面的对自定style的方式正确的是(a)
   A、

```
<resources>
    <style name="myStyle">
        <item name="android:layout_width">fill_parent</item>
    </style>
</resources>
```

B、

```
<style name="myStyle">
            <item name="android:layout_width">fill_parent</item>
</style>
```

C、

```
<resources>
    <item name="android:layout_width">fill_parent</item>
</resources>
```

D、

```
<resources>
    <style name="android:layout_width">fill_parent</style>
</resources>
```

10. 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()
11. 在SQL Server Management Studio 中运行下列T-SQL语句，其输出值（c）
    `SELECT @@IDENTITY`
    A、 可能为0.1
    B、 可能为3
    C、 不可能为-100
    D、 肯定为0
12. 在SQL Server 2005中运行如下T-SQL语句，假定SALES表中有多行数据，执行查询之后的结果是（d）。

```
BEGIN TRANSACTION A
   Update SALES Set qty=30 WHERE qty<30
BEGIN TRANSACTION B
   Update SALES Set qty=40 WHERE qty<40
   Update SALES Set qty=50 WHERE qty<50
   Update SALES Set qty=60 WHERE qty<60
COMMIT  TRANSACTION B
COMMIT TRANSACTION A
```

A、SALES表中qty列最小值大于等于30
B、SALES表中qty列最小值大于等于40
C、SALES表中qty列的数据全部为50
D、SALES表中qty列最小值大于等于60

13. 在android中使用SQLiteOpenHelper这个辅助类时，可以生成一个数据库，并可以对数据库版本进行管理的方法可以是(ab)
    A、getWriteableDatabase()
    B、getReadableDatabase()
    C、getDatabase()
    D、getAbleDatabase()
14. 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
15. 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
16. 下面在AndroidManifest.xml文件中注册BroadcastReceiver方式正确的(a)
    A、

```
<receiver android:name="NewBroad">
   <intent-filter>
       <action
          android:name="android.provider.action.NewBroad"/>
       <action>
   </intent-filter>
</receiver>
```

    B、

```
<receiver android:name="NewBroad">
   <intent-filter>
       android:name="android.provider.action.NewBroad"/>
   </intent-filter>
</receiver>
```

    C、

```
<receiver android:name="NewBroad">
   <action
         android:name="android.provider.action.NewBroad"/>
    <action>
</receiver>
```

    D、

```
<intent-filter>
   <receiver android:name="NewBroad">
       <action>
          android:name="android.provider.action.NewBroad"/>
       <action>
   </receiver>
</intent-filter>
```

17. 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
18. 我们都知道Hanlder是线程与Activity通信的桥梁,如果线程处理不当，你的机器就会变得越慢，那么线程销毁的方法是(a)
    A、onDestroy()
    B、onClear()
    C、onFinish()
    D、onStop()
19. 下面退出Activity错误的方法是(c)
    A、finish()
    B、抛异常强制退出
    C、System.exit()
    D、onStop()
20. 下面属于android的动画分类的有(ab)(多项)
    A、Tween B、Frame C、Draw D、Animation
21. 下面关于Android dvm的进程和Linux的进程,应用程序的进程说法正确的是(d)
    A、DVM指dalivk的虚拟机.每一个Android应用程序都在它自己的进程中运行,不一定拥有一个独立的Dalvik虚拟机实例.而每一个DVM都是在Linux 中的一个进程,
    所以说可以认为是同一个概念.
    B、DVM指dalivk的虚拟机.每一个Android应用程序都在它自己的进程中运行,不一定拥有一个独立的Dalvik虚拟机实例.而每一个DVM不一定都是在Linux 中的一个进程,
    所以说不是一个概念.
    C、DVM指dalivk的虚拟机.每一个Android应用程序都在它自己的进程中运行,都拥有一个独立的Dalvik虚拟机实例.而每一个DVM不一定都是在Linux 中的一个进程,
    所以说不是一个概念.
    D、DVM指dalivk的虚拟机.每一个Android应用程序都在它自己的进程中运行,都拥有一个独立的Dalvik虚拟机实例.而每一个DVM都是在Linux 中的一个进程,
    所以说可以认为是同一个概念.
22. Android项目工程下面的assets目录的作用是什么(b)
    A、放置应用到的图片资源。
    B、主要放置多媒体等数据文件
    C、放置字符串，颜色，数组等常量数据
    D、放置一些与UI相应的布局文件，都是xml文件

23. 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
24. 下列对android NDK的理解正确的是(abcd)
    A、NDK是一系列工具的集合
    B、NDK 提供了一份稳定、功能有限的 API 头文件声明。
    C、使 “Java+C” 的开发方式终于转正，成为官方支持的开发方式
    D、NDK 将是 Android 平台支持 C 开发的开端
25. android中常用的四个布局是framlayout，linenarlayout，relativelayout和tablelayout。
26. android 的四大组件是activiey，service，broadcast和contentprovide。
27. java.io包中的objectinputstream和objectoutputstream类主要用于对对象(Object)的读写。
28. android 中service的实现方法是：startservice和bindservice。
29. 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onrestart,onresume,onpause,onstop。
30. android的数据存储的方式sharedpreference,文件,SQlite,contentprovider,网络。
31. 当启动一个Activity并且新的Activity执行完后需要返回到启动它的Activity来执行 的回调函数是startActivityResult()。
32. 请使用命令行的方式创建一个名字为myAvd,sdk版本为2.2,sd卡是在d盘的根目录下,名字为scard.img， 并指定屏幕大小HVGA._____。
33. 程序运行的结果是：_good and gbc__。

```
public class Example{
    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';
    }
}
```

34. 在android中，请简述jni的调用过程。(8分)
    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
35. 简述Android应用程序结构是哪些?(7分)
    Android应用程序结构是：
    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)

36. 请继承SQLiteOpenHelper实现：(10分)
    1）.创建一个版本为1的“diaryOpenHelper.db”的数据库
    2）.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)
    3）.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,String name,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 table diary"+
                    "("+
                    "_id integer primary key autoincrement,"+
                    "topic varchar(100),"+
                    "content varchar(1000)"+
                    ")";
        db.execSQL(sql);
    }
    //若数据库版本有更新，则调用此方法
    public void onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion)
    {

        String sql = "drop table if exists diary";
        db.execSQL(sql);
        this.onCreate(db);
    }
}
```

37. 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。(10分)

```
public class ProgressBarStu extends Activity {

    private ProgressBar progressBar = null;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.progressbar);
        //从这到下是关键
        progressBar = (ProgressBar)findViewById(R.id.progressBar);

        Thread thread = new Thread(new Runnable() {

            @Override
            public void run() {
                int progressBarMax = progressBar.getMax();
                try {
```

```
                    while(progressBarMax!=progressBar.getProgress())
                    {

                        int stepProgress = progressBarMax/10;
                        int currentprogress = progressBar.getProgress();

progressBar.setProgress(currentprogress+stepProgress);
                        Thread.sleep(1000);
                    }

                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

            }
        });

        thread.start();
        //关键结束
    }
 }
```

38. 如何退出Activity？如何安全退出已调用多个Activity的Application？
    对于单一Activity的应用来说，退出很简单，直接finish()即可。
    当然，也可以用killProcess()和System.exit()这样的方法。
    但是，对于多Activity的应用来说，在打开多个Activity后，如果想在最后打开的Activity直接退出，上边的方法都是没有用的，因为上边的方法都是结束一个Activity而已。
    当然，网上也有人说可以。
    就好像有人问，在应用里如何捕获Home键，有人就会说用keyCode比较KEYCODE_HOME即可，而事实上如果不修改framework，根本不可能做到这一点一样。
    所以，最好还是自己亲自试一下。
    那么，有没有办法直接退出整个应用呢？
    在2.1之前，可以使用ActivityManager的restartPackage方法。
    它可以直接结束整个应用。在使用时需要权限android.permission.RESTART_PACKAGES。
    注意不要被它的名字迷惑。
    可是，在2.2，这个方法失效了。
    在2.2添加了一个新的方法，killBackgroundProcesses()，需要权限android.permission.KILL_BACKGROUND_PROCESSES。
    可惜的是，它和2.2的restartPackage一样，根本起不到应有的效果。
    另外还有一个方法，就是系统自带的应用程序管理里，强制结束程序的方法，forceStopPackage()。
    它需要权限android.permission.FORCE_STOP_PACKAGES。
    并且需要添加android:sharedUserId="android.uid.system"属性
    同样可惜的是，该方法是非公开的，他只能运行在系统进程，第三方程序无法调用。
    因为需要在Android.mk中添加LOCAL_CERTIFICATE := platform。
    而Android.mk是用于在Android源码下编译程序用的。
    从以上可以看出，在2.2，没有办法直接结束一个应用，而只能用自己的办法间接办到。
    现提供几个方法，供参考：

    - 抛异常强制退出：
      该方法通过抛异常，使程序Force Close。
      验证可以，但是，需要解决的问题是，如何使程序结束掉，而不弹出Force Close的窗口。
    - 记录打开的Activity：
      每打开一个Activity，就记录下来。在需要退出时，关闭每一个Activity即可。

- 发送特定广播：
  在需要结束应用时，发送一个特定的广播，每个Activity收到广播后，关闭即可。
- 递归退出
  在打开新的Activity时使用startActivityForResult，然后自己加标志，在onActivityResult中处理，递归关闭。
  除了第一个，都是想办法把每一个Activity都结束掉，间接达到目的。
  但是这样做同样不完美。
  你会发现，如果自己的应用程序对每一个Activity都设置了nosensor，在两个Activity结束的间隙，sensor可能有效了。
  但至少，我们的目的达到了，而且没有影响用户使用。
  为了编程方便，最好定义一个Activity基类，处理这些共通问题。

39. 请介绍下Android中常用的五种布局。
    FrameLayout（框架布局），LinearLayout （线性布局），AbsoluteLayout（绝对布局），RelativeLayout（相对布局），TableLayout（表格布局）

40. 请介绍下Android的数据存储方式。
    - SharedPreferences方式
    - 文件存储方式
    - SQLite数据库方式
    - 内容提供器（Content provider）方式
    - 网络存储方式

41. 请介绍下ContentProvider是如何实现数据共享的。
    创建一个属于你自己的Content provider或者将你的数据添加到一个已经存在的Content provider中，前提是有相同数据类型并且有写入Content provider的权限。

42. 如何启用Service，如何停用Service。
    Android中的service类似于windows中的service，service一般没有用户操作界面，它运行于系统中不容易被用户发觉，
    可以使用它开发如监控之类的程序。
    一。步骤
    第一步：继承Service类
    public class SMSService extends Service { }
    第二步：在AndroidManifest.xml文件中的节点里对服务进行配置:

    二。Context.startService()和Context.bindService
    服务不能自己运行，需要通过调用Context.startService()或Context.bindService()方法启动服务。这两个方法都可
    以启动Service，但是它们的使用场合有所不同。
    1.使用startService()方法启用服务，调用者与服务之间没有关连，即使调用者退出了，服务仍然运行。
    使用bindService()方法启用服务，调用者与服务绑定在了一起，调用者一旦退出，服务也就终止。
    2.采用Context.startService()方法启动服务，在服务未被创建时，系统会先调用服务的onCreate()方法，
    接着调用onStart()方法。如果调用startService()方法前服务已经被创建，多次调用startService()方法并
    不会导致多次创建服务，但会导致多次调用onStart()方法。
    采用startService()方法启动的服务，只能调用Context.stopService()方法结束服务，服务结束时会调用
    onDestroy()方法。

    3.采用Context.bindService()方法启动服务，在服务未被创建时，系统会先调用服务的onCreate()方法，
    接着调用onBind()方法。这个时候调用者和服务绑定在一起，调用者退出了，系统就会先调用服务的onUnbind()方法，

。接着调用onDestroy()方法。如果调用bindService()方法前服务已经被绑定，多次调用bindService()方法并不会
导致多次创建服务及绑定(也就是说onCreate()和onBind()方法并不会被多次调用)。如果调用者希望与正在绑定的服务
解除绑定，可以调用unbindService()方法，调用该方法也会导致系统调用服务的onUnbind()-->onDestroy()方法。

三。Service的生命周期

1. Service常用生命周期回调方法如下：
   onCreate() 该方法在服务被创建时调用，该方法只会被调用一次，无论调用多少次startService()或bindService()方法，
   服务也只被创建一次。 onDestroy()该方法在服务被终止时调用。
2. Context.startService()启动Service有关的生命周期方法
   onStart() 只有采用Context.startService()方法启动服务时才会回调该方法。该方法在服务开始运行时被调用。
   多次调用startService()方法尽管不会多次创建服务，但onStart() 方法会被多次调用。
3. Context.bindService()启动Service有关的生命周期方法
   onBind()只有采用Context.bindService()方法启动服务时才会回调该方法。该方法在调用者与服务绑定时被调用，
   当调用者与服务已经绑定，多次调用Context.bindService()方法并不会导致该方法被多次调用。
   onUnbind()只有采用Context.bindService()方法启动服务时才会回调该方法。该方法在调用者与服务解除绑定时被调用。

备注：

1. 采用startService()启动服务
   Intent intent = new Intent(DemoActivity.this, DemoService.class);
   startService(intent);
2. Context.bindService()启动
   Intent intent = new Intent(DemoActivity.this, DemoService.class);
   bindService(intent, conn, Context.BIND_AUTO_CREATE);
   //unbindService(conn);//解除绑定

43. 注册广播有几种方式，这些方式有何优缺点？请谈谈Android引入广播机制的用意。
Android广播机制（两种注册方法）
在android下，要想接受广播信息，那么这个广播接收器就得我们自己来实现了，我们可以继承BroadcastReceiver，就可以有一个广播接受器了。
有个接受器还不够，我们还得重写BroadcastReceiver里面的onReceiver方法，当来广播的时候我们要干什么，这就要我们自己来实现，
不过我们可以搞一个信息防火墙。具体的代码：

```
public class SmsBroadCastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle bundle = intent.getExtras();
        Object[] object = (Object[])bundle.get("pdus");
        SmsMessage sms[]=new SmsMessage[object.length];
        for(int i=0;i<object.length;i++) {
            sms[0] = SmsMessage.createFromPdu((byte[])object[i]);
            Toast.makeText(context, "来
自"+sms[i].getDisplayOriginatingAddress()+" 的消息
是："+sms[i].getDisplayMessageBody(), Toast.LENGTH_SHORT).show();
        }
        //终止广播，在这里我们可以稍微处理，根据用户输入的号码可以实现短信防火墙。
        abortBroadcast();
```

```
    }
}
```

当实现了广播接收器，还要设置广播接收器接收广播信息的类型，这里是信息：android.provider.Telephony.SMS_RECEIVED
我们就可以把广播接收器注册到系统里面，可以让系统知道我们有个广播接收器。这里有两种，一种是代码动态注册：

```
//生成广播处理
smsBroadCastReceiver = new SmsBroadCastReceiver();
//实例化过滤器并设置要过滤的广播
IntentFilter intentFilter = new
IntentFilter("android.provider.Telephony.SMS_RECEIVED");
//注册广播
BroadCastReceiverActivity.this.registerReceiver(smsBroadCastReceiver,
intentFilter);
```

一种是在AndroidManifest.xml中配置广播

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="spl.broadCastReceiver"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity android:name=".BroadCastReceiverActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />

            </intent-filter>
        </activity>

        <!--广播注册-->
        <receiver android:name=".SmsBroadCastReceiver">
            <intent-filter android:priority="20">
                <action
android:name="android.provider.Telephony.SMS_RECEIVED"/>
            </intent-filter>
        </receiver>

    </application>

    <uses-sdk android:minSdkVersion="7" />

    <!-- 权限申请 -->
    <uses-permission android:name="android.permission.RECEIVE_SMS"></uses-
permission>
 </manifest>
```

两种注册类型的区别是：

- 第一种不是常驻型广播，也就是说广播跟随程序的生命周期。

- 第二种是常驻型，也就是说当应用程序关闭后，如果有信息广播来，程序也会被系统调用自动运行。

44. 请解释下在单线程模型中Message、Handler、Message Queue、Looper之间的关系。

Handler简介：

一个Handler允许你发送和处理Message和Runable对象，这些对象和一个线程的MessageQueue相关联。每一个线程实例和一个单独的线程以及该线程的MessageQueue相关联。

当你创建一个新的Handler时，它就和创建它的线程绑定在一起了。这里，线程我们也可以理解为线程的MessageQueue。从这一点上来看，

Handler把Message和Runable对象传递给MessageQueue，而且在这些对象离开MessageQueue时，Handler负责执行他们。

Handler有两个主要的用途：（1）确定在将来的某个时间点执行一个或者一些Message和Runnable对象。（2）在其他线程（不是Handler绑定线程）中排入一些要执行的动作。

Scheduling Message，即（1），可以通过以下方法完成：

post(Runnable):Runnable在handler绑定的线程上执行，也就是说不创建新线程。

postAtTime(Runnable,long):

postDelayed(Runnable,long):

sendEmptyMessage(int):

sendMessage(Message):

sendMessageAtTime(Message,long):

sendMessageDelayed(Message,long):

post这个动作让你把Runnable对象排入MessageQueue,MessageQueue受到这些消息的时候执行他们，当然以一定的排序。sendMessage这个动作允许你把Message对象排成队列，

这些Message对象包含一些信息，Handler的hanlerMessage(Message)会处理这些Message.当然，handlerMessage(Message)必须由Handler的子类来重写。这是编程人员需要作的事。

当posting或者sending到一个Hanler时，你可以有三种行为：当MessageQueue准备好就处理，定义一个延迟时间，定义一个精确的时间去处理。

后两者允许你实现timeout,tick,和基于时间的行为。

当你的应用创建一个新的进程时，主线程（也就是UI线程）自带一个MessageQueue，这个MessageQueue管理顶层的应用对象（像activities,broadcast receivers等）和

主线程创建的窗体。你可以创建自己的线程，并通过一个Handler和主线程进行通信。这和之前一样，通过post和sendmessage来完成，差别在于在哪一个线程中执行这么方法。

在恰当的时候，给定的Runnable和Message将在Handler的MessageQueue中被Scheduled。

Message简介：

Message类就是定义了一个信息，这个信息中包含一个描述符和任意的数据对象，这个信息被用来传递给Handler.Message对象提供额外的两个int域和一个Object域，

这可以让你在大多数情况下不用作分配的动作。尽管Message的构造函数是public的，但是获取Message实例的最好方法是调用Message.obtain(),

或者Handler.obtainMessage()方法，这些方法会从回收对象池中获取一个。

MessageQueue简介：

这是一个包含message列表的底层类。Looper负责分发这些message。Messages并不是直接加到一个MessageQueue中，而是通过MessageQueue.IdleHandler关联到Looper。

你可以通过Looper.myQueue()从当前线程中获取MessageQueue。

Looper简介：

Looper类被用来执行一个线程中的message循环。默认情况，没有一个消息循环关联到线程。在线程中调用prepare()创建一个Looper，然后用loop()来处理messages，直到循环终止。

大多数和message loop的交互是通过Handler。

下面是一个典型的带有Looper的线程实现。

```
class LooperThread extends Thread {
  public Handler mHandler;

  public void run() {
```

```
        Looper.prepare();

        mHandler = new Handler() {
            public void handleMessage(Message msg) {
                // process incoming messages here
            }
        };

        Looper.loop();
    }
}
```

45. AIDL的全称是什么？如何工作？能处理哪些类型的数据？
    AIDL的英文全称是Android Interface Define Language
    当A进程要去调用B进程中的service时，并实现通信，我们通常都是通过AIDL来操作的
    A工程：
    首先我们在net.blogjava.mobile.aidlservice包中创建一个RemoteService.aidl文件，在里面我们自定义一个接口，含有方法get。ADT插件会在gen目录下自动生成一个RemoteService.java文件，该类中含有一个名为RemoteService.stub的内部类，该内部类中含有aidl文件接口的get方法。
    说明一：aidl文件的位置不固定，可以任意
    然后定义自己的MyService类，在MyService类中自定义一个内部类去继承RemoteService.stub这个内部类，实现get方法。在onBind方法中返回这个内部类的对象，系统会自动将这个对象封装成IBinder对象，传递给他的调用者。
    其次需要在AndroidManifest.xml文件中配置MyService类，代码如下：

    为什么要指定调用AIDL服务的ID,就是要告诉外界MyService这个类能够被别的进程访问，只要别的进程知道这个ID，正是有了这个ID,B工程才能找到A工程实现通信。
    说明：AIDL并不需要权限
    B工程：
    首先我们要将A工程中生成的RemoteService.java文件拷贝到B工程中，在bindService方法中绑定aidl服务
    绑定AIDL服务就是将RemoteService的ID作为intent的action参数。
    说明：如果我们单独将RemoteService.aidl文件放在一个包里，那个在我们将gen目录下的该包拷贝到B工程中。如果我们将RemoteService.aidl文件和我们的其他类存放在一起，那么我们在B工程中就要建立相应的包，以保证RmoteService.java文件的报名正确，我们不能修改RemoteService.java文件
    bindService(new Inten("net.blogjava.mobile.aidlservice.RemoteService"), serviceConnection, Context.BIND_AUTO_CREATE);
    ServiceConnection的onServiceConnected(ComponentName name, IBinder service)方法中的service参数就是A工程中MyService类中继承了RemoteService.stub类的内部类的对象。

46. 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

47. 系统上安装了多种浏览器，能否指定某浏览器访问指定页面？请说明原由。
    通过直接发送Uri把参数带过去，或者通过manifest里的intentfilter里的data属性

48. 你如何评价Android系统？优缺点。

答：Android平台手机 5大优势：

- 开放性
  在优势方面，Android平台首先就是其开发性，开发的平台允许任何移动终端厂商加入到Android联盟中来。显著的开放性可以使其拥有更多的开发者，
  随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。开放性对于Android的发展而言，有利于积累人气，这里的人气包括消费者和厂商，
  而对于消费者来讲，随大的受益正是丰富的软件资源。开放的平台也会带来更大竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。
- 挣脱运营商的束缚
  在过去很长的一段时间，特别是在欧美地区，手机应用往往受到运营商制约，使用什么功能接入什么网络，几乎都受到运营商的控制。从去年iPhone 上市，
  用户可以更加方便地连接网络，运营商的制约减少。随着EDGE、HSDPA这些2G至3G移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈，
  当你可以通过手机IM软件方便地进行即时聊天时，再回想不久前天价的彩信和图铃下载业务，是不是像噩梦一样？互联网巨头Google推动的Android终端天生就有网络特色，
  将让用户离互联网更近。
- 丰富的硬件选择
  这一点还是与Android平台的开放性相关，由于Android的开放性，众多的厂商会推出千奇百怪，功能特色各具的多种产品。功能上的差异和特色，
  却不会影响到数据同步、甚至软件的兼容，好比你从诺基亚 Symbian风格手机 一下改用苹果 iPhone，同时还可将Symbian中优秀的软件带到iPhone上使用、
  联系人等资料更是可以方便地转移，是不是非常方便呢?
- 不受任何限制的开发商
  Android平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的阻扰，可想而知，会有多少新颖别致的软件会诞生。但也有其两面性，血腥、暴力、情色方面的程序和游戏如可控制正是留给Android难题之一。
- 无缝结合的Google应用
  如今叱诧互联网的Google已经走过10年度历史，从搜索巨人到全面的互联网渗透，Google服务如地图、邮件、搜索等已经成为连接用户和互联网的重要纽带，
  而Android平台手机将无缝结合这些优秀的Google服务。

再说Android的5大不足：

- 安全和隐私
  由于手机 与互联网的紧密联系，个人隐私很难得到保守。除了上网过程中经意或不经意留下的个人足迹，Google这个巨人也时时站在你的身后，
  洞穿一切，因此，互联网的深入将会带来新一轮的隐私危机。
- 首先开卖Android手机的不是最大运营商
  众所周知，T-Mobile在23日，于美国纽约发布 了Android首款手机G1。但是在北美市场，最大的两家运营商乃AT&T和Verizon，
  而目前所知取得Android手机销售权的仅有 T-Mobile和Sprint，其中T-Mobile的3G网络相对于其他三家也要逊色不少，因此，用户可以买账购买G1，
  能否体验到最佳的3G网络服务则要另当别论了!
- 运营商仍然能够影响到Android手机
  在国内市场，不少用户对购得移动定制机不满，感觉所购的手机被人涂画了广告一般。这样的情况在国外市场同样出现。
  Android手机的另一发售运营商Sprint就将在其机型中内置其手机商店程序。
- 同类机型用户减少
  在不少手机论坛都会有针对某一型号的子论坛，对一款手机的使用心得交流，并分享软件资源。而对于Android平台手机，由于厂商丰富，
  产品类型多样，这样使用同一款机型的用户越来越少，缺少统一机型的程序强化。举个稍显不当的例子，现在山寨机泛滥，品种各异，

就很少有专门针对某个型号山寨机的讨论和群组，除了哪些功能异常抢眼、颇受追捧的机型以外。

- 过分依赖开发商缺少标准配置
  在使用PC端的Windows Xp系统的时候，都会内置微软Windows Media Player这样一个浏览器程序，用户可以选择更多样的播放器，
  如Realplay或暴风影音等。但入手开始使用默认的程序同样可以应付多样的需要。在Android平台中，由于其开放性，软件更多依赖第三方厂商，
  比如Android系统的SDK中就没有内置音乐 播放器，全部依赖第三方开发，缺少了产品的统一性。

49. 什么是ANR 如何避免它?
    答：ANR：Application Not Responding，五秒
    在Android中，活动管理器和窗口管理器这两个系统服务负责监视应用程序的响应。当出现下列情况时，Android就会显示ANR对话框了：
    对输入事件(如按键、触摸屏事件)的响应超过5秒
    意向接受器(intentReceiver)超过10秒钟仍未执行完毕
    Android应用程序完全运行在一个独立的线程中(例如main)。这就意味着，任何在主线程中运行的，需要消耗大量时间的操作都会引发ANR。
    因为此时，你的应用程序已经没有机会去响应输入事件和意向广播(Intent broadcast)。
    因此，任何运行在主线程中的方法，都要尽可能的只做少量的工作。特别是活动生命周期中的重要方法如onCreate()和 onResume()等更应如此。
    潜在的比较耗时的操作，如访问网络和数据库;或者是开销很大的计算，比如改变位图的大小，需要在一个单独的子线程中完成(或者是使用异步请求，如数据库操作)。
    但这并不意味着你的主线程需要进入阻塞状态已等待子线程结束 -- 也不需要调用Therad.wait()或者Thread.sleep()方法。
    取而代之的是，主线程为子线程提供一个句柄(Handler)，让子线程在即将结束的时候调用它(xing:可以参看Snake的例子，这种方法与以前我们所接触的有所不同)。
    使用这种方法涉及你的应用程序，能够保证你的程序对输入保持良好的响应，从而避免因为输入事件超过5秒钟不被处理而产生的ANR。
    这种实践需要应用到所有显示用户界面的线程，因为他们都面临着同样的超时问题。

50. 什么情况会导致Force Close ?如何避免?能否捕获导致其的异常?
    答：一般像空指针啊，可以看起logcat，然后对应到程序中 来解决错误

51. 如何将SQLite数据库(dictionary.db文件)与apk文件一起发布?
    解答：可以将dictionary.db文件复制到Eclipse Android工程中的res aw目录中。所有在res aw目录中的文件不会被压缩，这样可以直接提取该目录中的文件。
    可以将dictionary.db文件复制到res aw目录中

52. 如何将打开res aw目录中的数据库文件?
    解答：在Android中不能直接打开res aw目录中的数据库文件，而需要在程序第一次启动时将该文件复制到手机内存或SD卡的某个目录中，然后再打开该数据库文件。复制的基本方法是使用getResources().openRawResource方法获得res aw目录中资源的 InputStream对象，然后将该InputStream对象中的数据写入其他的目录中相应文件中。
    在Android SDK中可以使用SQLiteDatabase.openOrCreateDatabase方法来打开任意目录中的SQLite数据库文件。

53. Android引入广播机制的用意?

    - 从MVC的角度考虑(应用程序内)
      其实回答这个问题的时候还可以这样问，android为什么要有那4大组件，现在的移动开发模型基本上也是照搬的web那一套MVC架构，只不过是改了点嫁妆而已。
      android的四大组件本质上就是为了实现移动或者说嵌入式设备上的MVC架构，它们之间有时候是一种相互依存的关系，有时候又是一种补充关系，
      引入广播机制可以方便几大组件的信息和数据交互。
    - 程序间互通消息(例如在自己的应用程序内监听系统来电)

- 效率上(参考UDP的广播协议在局域网的方便性)
- 设计模式上(反转控制的一种应用，类似监听者模式)

54. Android dvm的进程和Linux的进程, 应用程序的进程是否为同一个概念
    DVM指dalivk的虚拟机。每一个Android应用程序都在它自己的进程中运行，都拥有一个独立的Dalvik虚拟机实例。而每一个DVM都是在Linux 中的一个进程，
    所以说可以认为是同一个概念。
55. 说说mvc模式的原理，它在android中的运用
    MVC(Model_view_contraller)"模型*视图*控制器"。 MVC应用程序总是由这三个部分组成。Event(事件)导致Controller改变Model或View，或者同时改变两者。
    只要 Controller改变了Models的数据或者属性，所有依赖的View都会自动更新。类似的，只要Contro
56. DDMS和TraceView的区别?
    DDMS是一个程序执行查看器，在里面可以看见线程和堆栈等信息，TraceView是程序性能分析器。
57. java中如何引用本地语言
    可以用JNI（java native interface java 本地接口）接口 。
58. 谈谈Android的IPC（进程间通信）机制
    IPC是内部进程通信的简称， 是共享"命名管道"的资源。Android中的IPC机制是为了让Activity和Service之间可以随时的进行交互，故在Android中该机制，
    只适用于Activity和Service之间的通信，类似于远程方法调用，类似于C/S模式的访问。通过定义AIDL接口文件来定义IPC接口。Servier端实现IPC接口，
    Client端调用IPC接口本地代理。