一、判断题（认为陈述正确的打“✓”，陈述错误的打“×”）

1．String 类不可以继承。（ ✓ ）

2．匿名内部类可以继承非外部类。（ × ）

3．当一个线程进入一个对象的一个 synchronized 方法后，其它线程还可以进入此对象的其它方法。（ × ）

4. AndroidManifest.xml 对程序的 Layout 没有影响。（ × ）

5．aapt 是用来编译 AIDL 文件的。（ × ）

6．如果 Service 被 Context.startService(Intent, Bundle)方法启动，那么无论是否有活动绑定到该 Service，Service 都在后台运行。（ ✓）

7. 如果一个类被声明为 final，意味着它不能再派生出新的子类，不能作为父类被继承。因此一个类不能既被声明为 abstract 的，又被声明为 final 的。（ ✓ ）

8．Android 中， Looper 类不但包含了一个 MessageQueue，而且可以处理通过 Handler 类对象传递的 Message 或 Runnable 对象。（ ✓ ）

9．Java 中不会存在内存泄漏。（ × ）

10．SAX 解析器将 XML 整棵树被载入内存，能随机访问节点。（ × ）

二、选择题（有单选的，也有多选的）

1．下面那些类继承自 Collection 接口（ A， B ）

A． List B. Set C. Map

2. 那些数据类型可以作用在关键字 switch 上（ A，D，E，F ）

A．int B. string C. long

D. short E. char F. byte

3. try {}里有一个 return 语句，那么紧跟在这个 try 后的 finally {}里的 code 会不会被执行，什么时候被执行，在 return 前还是后?（ B ）

A. 不会执行 B. 在 return 之前 C. 在 return 之后

4. Android 平台存储数据的途径有（ A， B， C， F）

A．SharedPreference B. 文件 C. DataBase

D. Intent E. Parcel F. ContentProvider

5. Android 常用的 Widget 有：（ A， B， C， E， F）

A． Button B. ListView C. ProgressBar

D． Binder E. DatePicker F. SurfaceView

6．假如 name 是 String 类型的变量，下面有几种判断 name 为 null 或空值方法，那些是恰当的：（ C， D ）

A．

```
if (name != "") {
    //do something
}
```

B．

```
if (!name.equals("")) {
    //do something
}
```

C．

```
if (name != null && !name.equals("")) {
    //do something
}
```

D．

```
if (!"".equals(name)) {
    //do something
}
```

7．以下关于 Dalvik 虚拟机论述错误的是　　（　A，B，D ）

A．　Dalvik 是运行时环境是基于堆栈的。

B．　所有 Android 应用都运行在同一个 Dalvik 虚拟机实例里。

C．　Dalvik 只能执行 DEX 文件格式。

D．　最新版 Androdi 中，Dalvik 仍不支持 JIT。

8．Android IDL 可以使用的类型　　（ A， B， C， D， E， F， G）

A．　Java 基本类型　　B.　String　　C.　CharSequence

D.　Parcelable 对象　　E.　List　　F.　Map

G.　其他 AIDL 接口

9. 我们使用 AlterDialog，经常见到如下形式的代码：

```
new AlertDialog.Builder(mContext).
        setIcon(android.R.drawable.ic_dialog_alert).
        setCancelable(false).
        setTitle(mTitle).
        setMessage(mMessage).
        setPositiveButton(mPositiveTitle, mPositiveListener).
        setNegativeButton(mNegativeTitle, mNegativeListener).
        setOnCancelListener(mCancelListener).
        create();
```

请问这是哪个设计模式的体现：　　（　A　）

A．　Builder　　B. Prototype　　C. Factory Method　　D. Abstract Factory

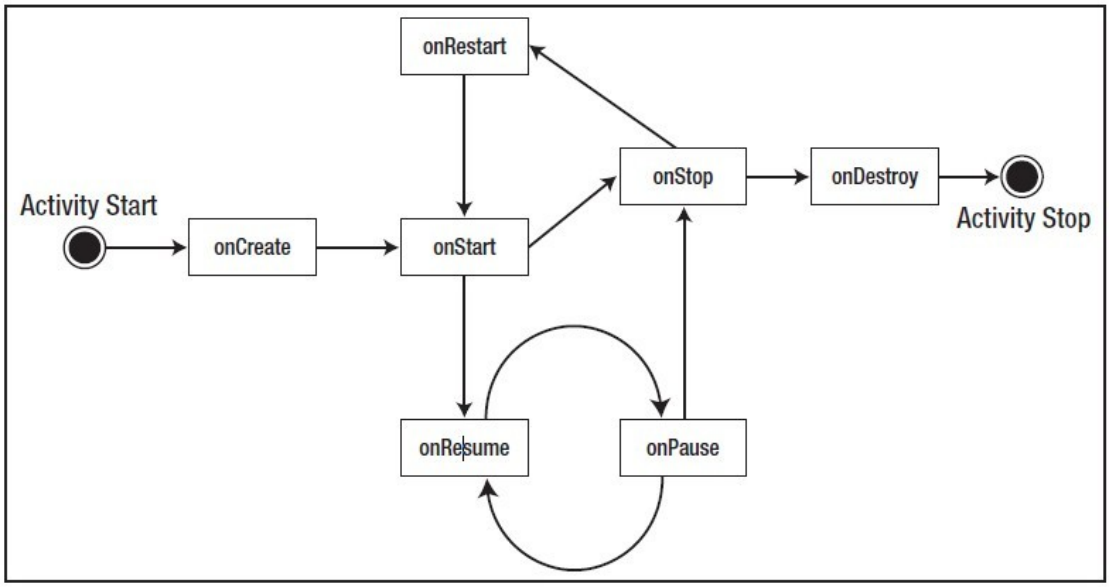
10. Android 的 IPC 通讯机制：　　（　B　）

A．　Dbus　　B. Binder　　C. socket　　D. named pipe

三、简答题

1． 请简单说明 Android 中 Activity 生命周期。

涉及 Create、Start、Pause、Resume、Stop、Destory，等几个状态。状态转换图如下：



2． sleep() 和 wait() 有什么区别?

sleep()方法是使线程停止一段时间的方法。在 sleep 时间间隔期满后，线程不一定立即恢复执行。这是因为在那个时刻，其它线程可能正在运行而且没有被调度为放弃执行，除非
(a)“醒来”的线程具有更高的优先级
(b)正在运行的线程因为其它原因而阻塞。

wait()是线程交互时，如果线程对一个同步对象 x 发出一个 wait()调用，该线程会暂停执行，被调对象进入等待状态，直到被唤醒或等待时间到。

3． Singleton 模式主要作用是保证在 Java 应用程序中，一个类 Class 只有一个实例存在。请写一个 Singleton 类出来。

一般 Singleton 模式通常有几种种形式:
第一种形式: 定义一个类，它的构造函数为 private 的，它有一个 static 的 private 的该类变量，在类初始化时实例话，通过一个 public 的 getInstance 方法获取对它的引用,继而调用其中的方法。

```
public class Singleton {
    private Singleton(){}
    //在自己内部定义自己一个实例，是不是很奇怪？
    //注意这是 private 只供内部调用
    private static Singleton instance = new Singleton();
```

```
    //这里提供了一个供外部访问本 class 的静态方法，可以直接访问
    public static Singleton getInstance() {
        return instance;
     }
}
```

第二种形式:

```
public class Singleton {
    private static Singleton instance = null;
    public static synchronized Singleton getInstance() {
    //这个方法比上面有所改进，不用每次都进行生成对象，只是第一次
    //使用时生成实例，提高了效率！
    if (instance==null)
        instance＝new Singleton();
return instance;        }
}
```

其他形式:

定义一个类，它的构造函数为 private 的，所有方法为 static 的。

一般认为第一种形式要更加安全些