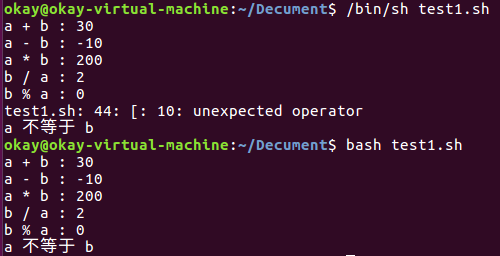
**2018.5.16**

问题1：

****

在一些用法上面，用/bin/sh编译，会出问题，百度说用bash？这些有什么不同么？那怎么知道什么时候用什么来编译？

问题2：重定向，通俗一点是啥意思啊？

解答：

1. sh和bash是两种shell解析器。它们可能会有一定的差异，对语法上可能会有一定的差异，但是基本语法是一样的。然后我们一般不直接使用bash或sh。一般是在.sh文件中加#! bin/bash, 正常在ubuntu上我们都是用bash解析器，然后再命令行上使用./xxx.sh运行脚本。所以以后都用bash的解析器。也不要像上面那样运行了。
2. 重定向简单理解就是要把打印的内容输出到哪里。比如正常情况下，脚本的中输出的内容都是输出到窗口上，如果你使用./test.sh > test 来重定向，那输出就会到test文件中，窗口上就不会有输出了。(当然输入也有重定向就是了，可以先理解输出)

**2018.5.3**

问题1：runOnUiThread(),这个里面，你说这个不属于创建一个新的线程，是因为它是内部调用类的源代码的意思么？ ( <https://blog.csdn.net/xlh1191860939/article/details/52895016?locationNum=1&fps=1> )

问题2：网络编程这节，需要自己去弄个Apache服务器么？

解答：

1. 第一、runOnUiThread()是Activity中的方法，和startActivity方法一样，是android提供给我们的API中的方法。

第二、runOnUiThread()可以在主线程中调用，也可以在子线程中调用。但是在主线程上调用runOnUiThread()没有意义啊，因为主线程本来就是可以更新UI啊(runOnUiThread()主要的用处就是去更新UI), 所以runOnUiThread()一般在子线程中调用。

第三、runOnUiThread()内部的机制也是Handler

1. 不用弄Apache服务器，网上有很多服务器接口，阿里云和百度都有提供一些开发者API(有些是免费的)，以后可以使用它们来学习。

**2018.5.2**

问题1：

private void showResponse(final *String response*){  
 runOnUiThread(new *Runnable*() {  
 @Override  
 public void run() {  
 //在这里进行UI操作，将结果显示在界面上  
 reponseText.setText(*response*);  
 }  
 });  
}

为啥这个子线程里面不用.start（）？？

解答：

1. 该方法在哪里被调用？
2. runOnUiThread()方法是用来干嘛的？
3. runOnUiThread()方法传入的Runnable是运行在哪个线程呢？

先解答第一点：

private void sendRequestWithHttpURLConnection(){

new Thread(new Runnable(){

……

showResponse(respone.toString());

}).start();

}

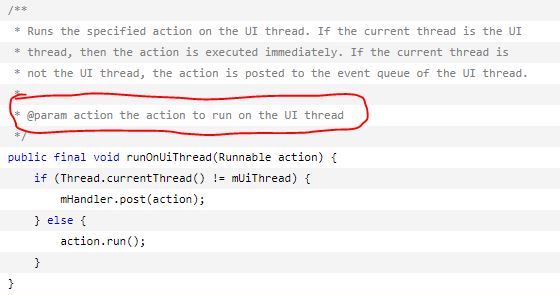
sendRequestWithHttpURLConnection()方法启动了一个子线程，然后调用了showRespone()方法。

然后第二点：

runOnUiThread()方法是用来更新UI的，而Android对UI的更新只能在UI线程(或者也叫做主线程)。

最后一点:

我把这个方法的截了图



红色框框中对参数做了说明，运行在UI线程上。一个应用启动后就是运行在UI线程上的，所以不需要我们手动去创建。

总结：

1. runOnUiThread()方法是Activity的方法，用来更新UI线程。
2. runOnUiThread()方法只在子线程中被调用。
3. runOnUiThread()中的参数是运行在UI线程上的
4. 更新UI有两种方法: (1)使用Handler (2) 使用runOnUiThread()

去总结一下上面两种**更新UI的用法**

1. Handler机制

①在**主线程**中定义Handler，Handler mainHandler= new Handler（）；

在其他地方定义，获取主线程的looper和Queue，Handler mainHandler= new Handler（Looper.getMainLooper（））；

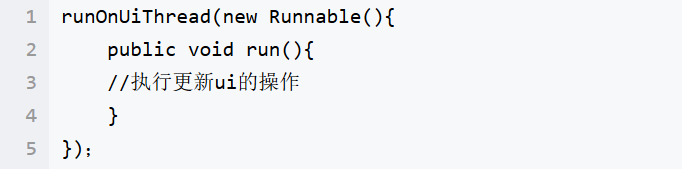
②获取到Handler后，用handler.post（Runnable r）方法把消息处理放在该handler依附的消息队列中（主线程消息队列）



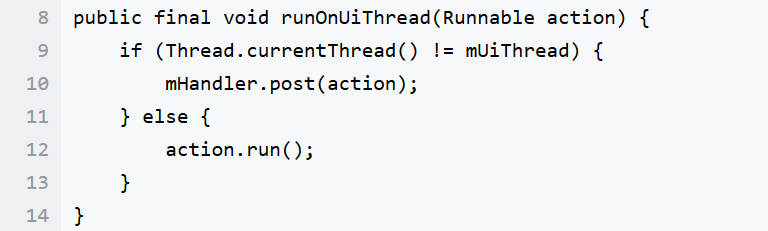


1. 使用runOnUiThread（）

利用Activity. runOnUiThread(Runnable)把更新ui的代码创建在Runnable中，然后在需要更新ui时，把这个Runnable对象传给Activity. runOnUiThread(Runnable)对象，就能在ui程序中被调用。如果当前线程时UI线程，那么行动是立即执行。如果当前线程不是UI线程，操作是发布到事件队列的UI线程。



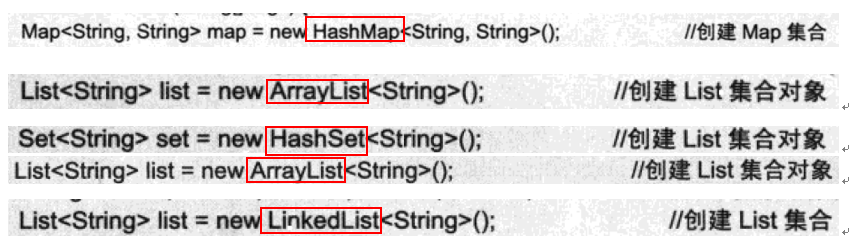
new了一个Runnable接口的子类的对象



如果当前不是主线程的话，则调用mHandler.post ( )方法。（内部）

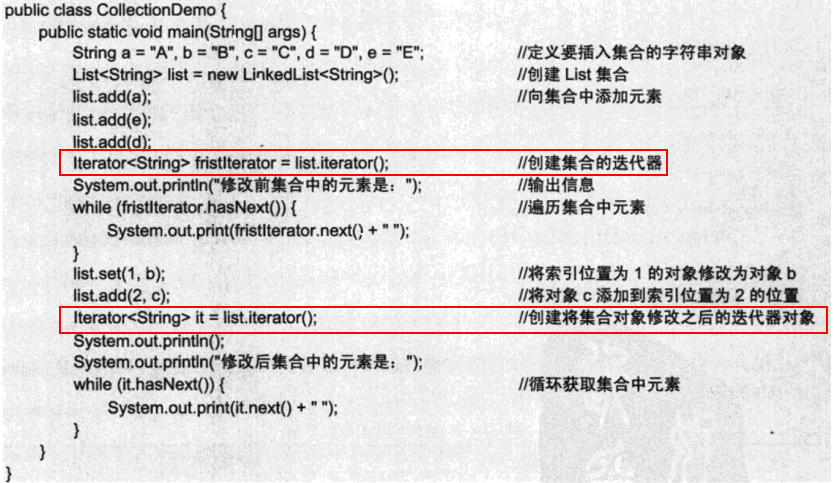
OK, 回到原来的问题，首先那个并没有创建线程，第二，遇到这种不知道的方法，可以先查具体用法和含义。而且你不觉得跟你平常用得线程不一样吗C:\Users\YONGDA~1.YAN\AppData\Local\Temp\SGPicFaceTpBq\14456\025F3483.png

**2018.4.19**

问题1：

这些都是新建集合，圈起来那个怎么确定的啊？

问题2 ：



为什么要创建迭代器？不是已经新建完集合了么？

解答：

1. 红色圈起来的都是前面的子类或者实现了前面的接口。所以后面的类的选择有两个原则：
2. 必须是实现前面的接口或者继承前面的类
3. 要根据你的需要来选择子类，比如ArrayList和LinkedList都是实现List接口。它们两个的差别在于数据结构方式不同，比如LinkedList内部使用的链表。你可以上网查看它们的**差异**，然后**根据实际需求来选择**。
4. **迭代器是用来遍历集合中的元素的。**当你调用它的next()，它就返回集合中的下一个元素给你。你要理清两者的关系，迭代器是集合特用的一种东西，专门用来遍历集合的(遍历就是把集合的所有元素从头到尾的读出来)。

**2018.4.18**

问题1：在项目里面的时候，直接就会说继承某个类啥的，或者适配器里面的泛型（继承RecyclerView.Adapter，泛型指定为<FruitAdapter.ViewHolder>）,这些是固定的么？见多就懂了？

问题2：适配器就是Listview这边才会用到的？啥时候要用到适配器，它的作用是啥？（好像很早以前也问过==、）

解答：

1. RecyclerView的适配器都是继承RecyclerView.Adapter， 适配器中的泛型指定为适配器中定义的**内部类**。

比如：

public class FruitAdapter extends RecyclerView.Adapter<FruitAdapter.ViewHolder>{

static class ViewHolder extends RecycleView. ViewHolder{

………..

}

……….

}

泛型FruitAdapter.ViewHolder是指FruitAdapter中的内部类ViewHolder.

再举个例子：

public class MyAdapter extends RecyclerView.Adapter<MyAdapter.MyViewHolder>{

………

public static class MyViewHolder extends RecyclerView.ViewHolder {

……….

}

……….

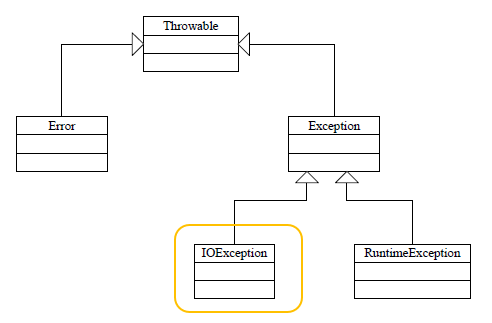
}

把红色的部分一一对应上，就知道怎么指定泛型了。

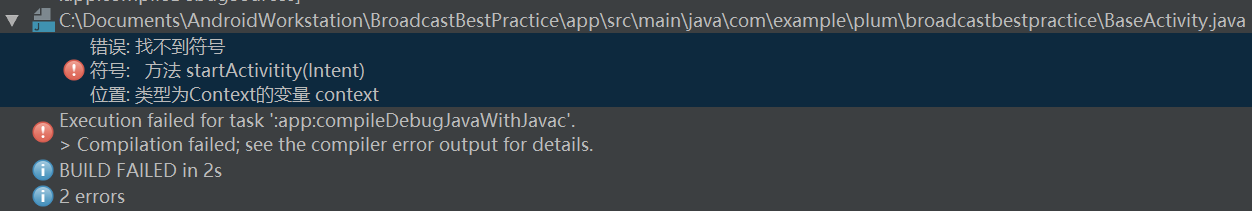
1. RecyclerView也用到适配器啊，只是用到的是RecyclerView.Adapter。ListView有很多种适配器，ArrayAdapter，BaseAdapter，SimpleAdapter等，还有其他一些控件也会使用到适配器，但是那些控件目前比较少用到。只要你用到RecyclerView和ListView这种列表控件，都会用到适配器。适配器说简单点就是用**来生成列表中的每一个项(item), 只要用到列表，都要用到适配器**。(会用RecyclerView后就不要再用ListView了)

**2018.4.13**

问题1：在java异常里面，需要去记有哪些异常类么？不然怎么知道什么时候用哪一种异常来处理？（我们所要处理的异常就是圈起来部分（检查异常），未检查异常可以直接忽略么？）



问题2: 新建一个广播接收器和新建一个类，继承自BroadcastReceiver，并重写父类的onReceiver（）方法，两者有什么不同，不都是为了创建一个广播接收器么？

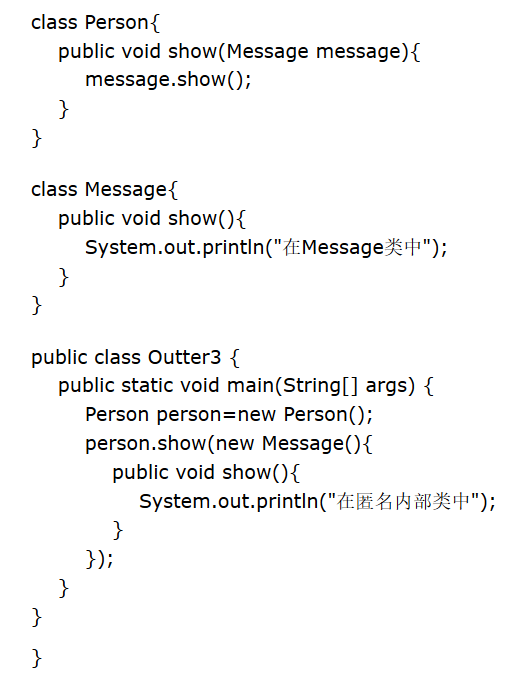
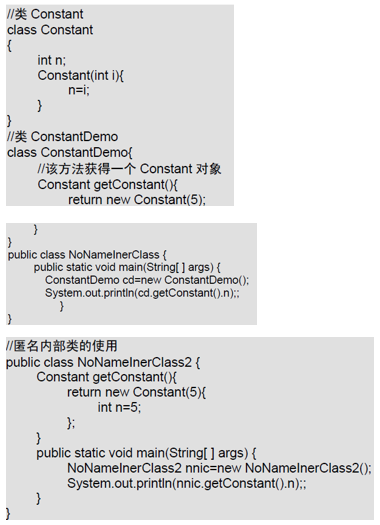
问题3：代码出现这个错误：

解答：

1. 异常不用去记，现在集成开发环境(IDE)都会提示你要加try{} catch(){} 。一般文件流操作或网络操作时会比较经常用到。未检测异常不会强制你去捕获，只有在真正运行时才能知道错误，但是一般是忽略，当报错一般说明代码没处理好。
2. 新建一个广播接收器，是IDE帮你创建一个类并继承BroadcastReceiver，并且会帮你在AndroidManifest.xml进行注册，这些都是IDE自动帮我们完成的。而新建一个类再去继承，这些动作就要我们手动完成了
3. 方法名写错

**2018.4.12**

问题1：

匿名内部类是没有类名的内部类？圈起来部分不就是匿名类么？不是有类名

然后右边代码的匿名类使用没有很懂。。。为啥两个分号？？

问题2：



<intent-filter> 不是只有在主活动的时候才出现的么？

问题3：

活动的启动模式，就了解下原理的示意图，ok么？

解答：

1. 圈起来的内容等价于：

class NewMessage extents Message{

public void show(){

System.out.println(“在匿名内部类中”);

}

}

person.show((new NewMessage()).show());

或者

NewMessage message = new NewMessage();

person.show(message.show());

里面重写Message的show方法，什么情况下才会重写呢？就是继承父类时才会去重写父类方法。但是框框内的代码却省去了继承的动作，也就是没有显式去写继承，而是通过new父类，然后重写父类方法。这种形式就是创建了一个匿名类，就是我们看不到真正定义的类。

两个分号应该是写错，没有任何意义。

1. 每个Activity, Service或者BroadcastReceiver都可以定义<intent-filter>。它主要是用在隐式Intent。用来过滤一些action和category等。
2. 用得不多，但是Activity跳转比较多时要注意，不要让Activity一直重复创建。面试时问的可能性还是蛮大的。但是目前先了解吧。

**2018.4.11**

问题1：内部类在应用中会使用很多么？看教程好乱，不是很理解……还有clone的话，是不是懂下啥格式就好了？

问题2：

直线的颜色不是这样改么？？

解答:

1. 内部类在监听器时使用的比较多，特别是匿名内部类。你在学习控件使用时，也提到了。在Android中一般就是监听器时喜欢用内部类。

clone这个东西可以不用掌握，基本不用。

1. 直线的颜色就是这样改，但是我试了一下你的布局，确实没变，我再查看一下是什么原因导致的。

**2018.4.10**

问题1：



这个难道不是应该改为Time opt.time；？？

解答：

上面代码的意思是：定义了一个ObjectParamTransfer类，里面有个成员变量time,它的类型是Time; 然后在main方法中定义了ObjectParamTransfer对象opt，而这个对象中有个time的成员，它的类型在类定义时已经被确定的。

而你的那种写法是想说time的类型不确定，所以要加Time。从来就没有这种写法，因为类中的成员都是确定类型的。

**2018.4.9**

1、



2、



3、



两者有啥区别，我怎么判定是要用哪种变量？（只知道说一个就是局部使用，一个是可以全局使用）

解答：

1. 不会，array1指向的对象和twoD[0]是两个不一样的存储空间。



就是看到的情况是：







即array1[0]和array2[0]的值都改变了，所以当你执行：



 //此时的twoD1[0]的值改变了，那么array1的值难道不是随着它而改变了？？

解答：

你上面的例子是一位数组的，而下面的是二维数组，这可能就是你不理解的问题所在吧。上面的一位数组，array1和array2存放的是地址，array[0]存放的是数值；

而下面的是二维数组，array1, twoD1[0], twoD1[4]都是存放地址，array1[0]或者twoD1[0][1]才是真正存放数值的。



如果只是数值的拷贝，那就完全可以理解了！（maybe又是转牛角尖……）

**ok啦，就是地址和数值的问题！！！**

1. 新建Java Project就会自动生成JRE System Library,可能环境搭建有问题
2. 局部变量只能在方法中使用，如果想在其他方法中也使用这个变量时，要把它定义为全局变量。可以直接使用全局的就ok了