List中的RetainAll(), RemoveAll(), Remove()

retainAll(list) 这个方法的作用是求两个集合的交集。 项目中用到的是ArrayLis，所以看了一下的他的源码，原来很简单。 如果list.retainAll() 直接按Ctrl和右键点击进去是:

```
   public boolean retainAll(Collection<?> collection);
```

一个简单的接口方法。 什么地方去实现了这个方法呢。 在Android Studio中点击右边的绿色小圆圈会有所有关于他的他实现，或者查询Java的官方API一样。 查看ArrayList的源码接口

```
  public class ArrayList<E> extends AbstractList<E> implements Cloneable,
  Serializable, RandomAccess
```

实现了AbstractList，但是这个AbstractList总并没有retainAll这个方法，继续向上寻找 AbstractList这个类

```
  public abstract class AbstractList<E> extends AbstractCollection<E>
```

这个抽象类中还没有，接着往上找AbstractColletion这个类，返现这个类中有retainAll的实现

```
   public boolean retainAll(Collection<?> collection) {
      boolean result = false;
      Iterator<?> it = iterator();
      while (it.hasNext()) {
          if (!collection.contains(it.next())) {
              it.remove();
              result = true;
          }
      }
      return result;
  }
```

从代码中可以看出，遍历参数中的集合，如果当前集合中包含参数中的元素，就从当前集合中移除。 那么contains是怎么实现的呢？

```
      public boolean contains(Object object);
```

只是一个接口而已。 在ArrayList中已经实现了这个方法：

```
     */
  @Override public boolean contains(Object object) {
      Object[] a = array;
      int s = size;
      if (object != null) {
          for (int i = 0; i < s; i++) {
              if (object.equals(a[i])) {
                  return true;
              }
          }
      } else {
          for (int i = 0; i < s; i++) {
              if (a[i] == null) {
                  return true;
              }
          }
      }
      return false;
```

```
}
```

所以最终是看你自定义类中的equals这个方法，所以我们可以实现equals这个方法来控制retainAll，来寻找我们所需要的交集。

removeAll 和remove(object) 类似

```
    public boolean removeAll(Collection<?> collection) {
    boolean result = false;
    Iterator<?> it = iterator();
    while (it.hasNext()) {
        if (collection.contains(it.next())) {
            it.remove();
            result = true;
        }
    }
    return result;
}
```

也是调用了contains这个方法