Object在Java中是所有类的父类，我们又形象的称之为超类。

可能上述的有点抽象不太好理解，那么如果这样描述：extends Object，是否能够理解呢，也就是说我们所写的每个类其实都隐形的写着这句话继承着Object类，但是这不需要我们写出来。

那么既然我们的类都继承了这个Object类，那它到底有些啥子用处呢？自然它里面有很多方法，这样我们自己类都会默认带有了作为父类Object的这些方法，使用这些方法以及重写这些方法才是我们使用超类的魅力所在。

那么，其实其中常用的方法也就三种，我们也就长话短说，只介绍这三种方法。

equals

这个方法大家应该都不陌生，相信大家都被那道比较字符串是否相等的题目坑过。那道题如果使用“==”比较，只会比较地址，地址不相等字符串相等也不等，正确的比较字符串是否相等就得用这里的equals，当然那里用的是String的equals。这样说不知道大家能不能转过弯，正如上文所述，使用超类的方法以及重写才是使用超类的魅力，String中就对equals进行了重写，能够比较字符串是否匹配，那么超类的equals长啥样呢，这里我直接截图源码（增加可信度嘛）。

```
public boolean equals(Object obj) {
    return (this == obj);
}
```

不知道这个地方大伙能否看懂，比如我定义了A a1，A a2。我这样写：

a1.equals(a2);

那么a2显然是这边的obj，a1作为方法调用者，this指代的是谁？那显然就是a1嘛，那么代码里的意思显然就是比较a1 == a2。那么就清晰了，超类最原始的equals方法仅仅是比较比较双方的地址是否一样。

那么，说到这里大家应该也明白了equals存在的功效其实就是为了比较两个对象的某些属性啥啥的，这样在开发时有特殊需求的话重写该方法即可。

finalize

这个方法如果学过C++的小伙伴应该比较清楚，只学过Java的小伙伴可能就有点懵了，没事，听我慢慢道来。

说到这个方法，可能我们又得扯到Java一个重要机制，GC，Java也正因为有了这个机制让很多没有C++基础的小伙伴不会太清楚这个方法。

我们知道，程序在运行过程中会定义很多个类，有很多类用完之后并不会再用到，这样这些类就成为了垃圾，是垃圾就得处理掉，不处理掉只会再内存累计，最后形容下，boom！！！

在C++中有个概念叫析构函数，每次使用资源后必须主动调用析构函数进行释放资源，而在Java中通过GC机制我们基本上不需要管，它会自动检测哪些类为垃圾然后处理掉。

有人要问了，这上面扯了这么多和finalize有啥关系，别急，下面就是了，说到finalize，先上一下源码的截图。

```
*/
protected void finalize() throws Throwable { }
```

可以看到这就是个空方法，啥都没有，我们怎么用呢，我就这么说吧，这个方法不需要我们主动调用，它就是在我们上述的GC中自动调用的，那么怎么演示呢，容我慢慢列出。

我们先重写一下该方法，不然啥内容没有，也看不出啥。

```
@Override
public void finalize(){
    System.out.println("你在调用finalize哦...");
}
```

看着这几行代码，是不是又能复习点啥呢，对，重写方法可以改变修饰符，也可以去掉后面的throws相关的。

再看我们怎么调用，虽说GC是自动识别的，但是我们强大的Java肯定得留后门不是，如下代码。

```
public static void main(String[] args) {
    new Outer().sysEdge();

    System.gc();
}
```

我们看结果。

```
我是一个匿名内部类...
你在调用finalize哦...
```

看，是不是在GC进行时自动调用的finalize。

toString

这个方法大家就更熟悉了，长话短说，先贴源码。

```
    public String toString() {
        return getClass().getName() + "@" + Integer.toHexString(hashCode());
    }
```

这就是为什么我们平常调用这个方法会返回如下格式结果的原因。

```
com.xyz.innerClass.Outer@1540e19d
```

类的全限定类名+@+对象地址。

那么这个方法的作用就显然了，主要作用还是在对它的重写，可以在调用这个类的时候就输出了关于这个类你想知道的信息，全凭自己处置。