1 什么是 Object 类

- Object 类是所有类的父类，如果一个类没有显式继承任何类，则会隐式继承 Object 类

2 Object 类的适用场景

- 任何情况，任何引用类型都可以直接调用 Object 类中的方法
- Object类位于java.lang包中，java.lang包包含着Java最基础和核心的类，在编译时会自动导入

3 Object 类中方法解析

3.1 getClass() 方法

- 获取类的类对象（Class 对象）
- 通过该类的 Class 对象，可以查看该类的一些基本信息，反射就用到了 Class 对象

```
//返回的是一个类对象（Class 对象：jvm在类加载时会为每个类生成一个Class<A>的Class对象在Java堆中，每个A类型的实例都要通过这个Class对象来进行实例化），所以这个 Class 对象是唯一的，只有第一次加载的时候才会生成
//每个类的运行时的类型信息就是用Class对象表示的
public final native Class<?> getClass();
```

3.2 clone() 方法

- Java 浅/深克隆、克隆数组
- 如果要调用 clone() 方法，该对象必须实现 Clonealbe 接口，不然会报 CloneNotSupportedException 异常
- 不建议通过这个方法来实现对象的拷贝（浅拷贝或深拷贝）—— 通过拷贝工厂来实现对象的拷贝

```
//返回一个对象的拷贝：这个拷贝存在于堆中地址不同于被拷贝的对象
//但是仍然是浅拷贝：拷贝对象中关联的对象，它在堆中的地址仍然是被拷贝对象对应对象的地址
protected native Object clone() throws CloneNotSupportedException;
//建议通过工厂的形式得到复制对象
public static Yum newInstance(Yum yum);//可以通过这个方法来返回一个对象的拷贝
```

3.3 hashCode() 方法

- 本地 hashCode() 方法，返回的是对象的哈希码值（注：这个值并不是地址值，只能保证相同类的不同实例返回的值是不同的，不同类的实例返回的值可能相同）
- 我们重写该方法的时候也要保证以上情况，并且要遵守如下的 Object 规范
  - 在Java应用程序程序执行期间，对于同一对象多次调用hashCode()方法时，其返回的哈希码是相同的，前提是将对象进行equals比较时所用的标尺信息未做修改。在Java应用程序的一次执行到另外一次执行，同一对象的hashCode()返回的哈希码无须保持一致
  - 如果两个对象相等（依据：调用equals()方法），那么这两个对象调用hashCode()返回的哈希码也必须相等
  - 反之，两个对象调用hasCode()返回的哈希码相等，这两个对象不一定相等
- 通过以上规范可以看出，当我们重写了 equals() 方法时，必须重写 hashCode() 方法

```
//hashCode 值主要用于散列表中(hashMap等)
public native int hashCode();
```

3.4 finalize() 方法

- 终结方法通常是不可预测的，也是很危险的，一般情况下不必要使用，这里不分析了，可以查看 Effective Java 中的相关说明

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

3.5 toString() 方法

- 默认返回 className + @ + hashCode 的十六机制形式
- 建议每个子类都重写该方法

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

3.6 notify()、notifyAll()、wait(..)

- 参见 Java - sleep yield wait notify notifyAll join 方法功能解析

3.7 equals() 方法

- 参见 java == 与 equals() 方法的区别

参考

Effective Java 中文第二版 根类Object详细分析 Java基础12：深入理解Class类和Object类 JDK 1.8u171