try-with-resource语法

```
1. public static void main(String[] args) {
2.     FileInputStream inputStream = null;
3.     try {
4.         inputStream = new FileInputStream(new File("test"));
5.         System.out.println(inputStream.read());
6.     } catch (IOException e) {
7.         throw new RuntimeException(e.getMessage(), e);
8.     } finally {
9.         if (inputStream != null) {
10.            try {
11.                inputStream.close();
12.            } catch (IOException e) {
13.                throw new RuntimeException(e.getMessage(),
e);
14.            }
15.         }
16.     }
17. }
```

新语法之后

```
1. public static void main(String[] args) {
2.     try (FileInputStream inputStream = new FileInputStream(new
File("test"))) {
3.         System.out.println(inputStream.read());
4.     } catch (IOException e) {
5.         throw new RuntimeException(e.getMessage(), e);
6.     }
7. }
```

在JDK7以前，Java没有自动关闭外部资源的语法特性，直到JDK7中新增了try-with-resource语法，才实现了这一功能。
那什么是try-with-resource呢？简而言之，当一个外部资源的句柄对象（比如FileInputStream对象）实现了AutoCloseable接口，那么就可以将上面的板式代码简化为上面形式

Objects类

```
1. //比较两个对象是否相等（首先比较内存地址，然后比较a.equals(b)，只要符合
其中之一返回true）
2. public static boolean equals(Object a, Object b);
3.
4. //深度比较两个对象是否相等(首先比较内存地址,相同返回true;如果传入的是数
组，则比较数组内的对应下标值是否相同)
5. public static boolean deepEquals(Object a, Object b);
6.
7. //返回对象的hashCode，若传入的为null，返回0
```

```
8. public static int hashCode(Object o);
9.
10. //返回传入可变参数的所有值的hashCode的总和（这里说总和有点牵强，具体参考Arrays.hashCode()方法）
11. public static int hash(Object... values);
12.
13. //返回对象的String表示，若传入null，返回null字符串
14. public static String toString(Object o)
15.
16. //返回对象的String表示，若传入null，返回默认值nullDefault
17. public static String toString(Object o, String nullDefault)
18.
19. //使用指定的比较器c 比较参数a和参数b的大小（相等返回0，a大于b返回整数，a小于b返回负数）
20. public static <T> int compare(T a, T b, Comparator<? super T> c)
21.
22. //如果传入的obj为null抛出NullPointerException,否者返回obj
23. public static <T> T requireNonNull(T obj)
24.
25. //如果传入的obj为null抛出NullPointerException并可以指定错误信息message,否者返回obj
26. public static <T> T requireNonNull(T obj, String message)
27.
28. ------------------------------以下是jdk8新增方法----------------------------
29.
30. //判断传入的obj是否为null，是返回true,否者返回false
31. public static boolean isNull(Object obj)
32.
33. //判断传入的obj是否不为null，不为空返回true,为空返回false （和isNull()方法相反）
34. public static boolean nonNull(Object obj)
35.
36. //如果传入的obj为null抛出NullPointerException并且使用参数messageSupplier指定错误信息,否者返回obj
37. public static <T> T requireNonNull(T obj, Supplier<String> messageSupplier)
```