9.6 Class `Object`

As we discussed earlier in this chapter, all classes in Java inherit directly or indirectly from class `Object` (package `java.lang`), so its 11 methods (some are overloaded) are inherited by all other classes. Figure 9.12 summarizes `Object`’s methods. We discuss several `Object` methods throughout this book (as indicated in Fig. 9.12).

Method	Description
`equals`	This method compares two objects for equality and returns `true` if they’re equal `false` otherwise. The method takes any `Object` as an argument. When objects particular class must be compared for equality, the class should override method `equ` compare the contents of the two objects. For the requirements of implementing this r (which include also overriding method `hashCode`), refer to the method’s document `http://docs.oracle.com/javase/8/docs/api/java/lang/Object` The default `equals` implementation uses operator `==` to determine whether two refe refer to the same object in memory. Section 14.3.3 demonstrates class `String`’s `ec` method and differentiates between comparing `String` objects with `==` and with `eq`
`hashCode`	Hashcodes are `int` values used for high-speed storage and retrieval of information s a hashtable data structure (see Section 16.10). This method is also called as part `Object`’s default `toString` method implementation.
`toString`	This method (introduced in Section 9.4.1) returns a `String` representation of an obj default implementation of this method returns the package name and class name o object’s class typically followed by a hexadecimal representation of the value returne object’s `hashCode` method.
`wait`, `notify`, `notifyAll`	Methods `notify`, `notifyAll` and the three overloaded versions of `wait` are rel multithreading, which is discussed in Chapter 23.
`getClass`	Every object in Java knows its own type at execution time. Method `getClass` (us Sections 10.5 and 12.5) returns an object of class `Class` (package `java.lang`)

	contains information about the object's type, such as its class name (returned by `Cl` method `getName`).
`finalize`	This `protected` method is called by the garbage collector to perform terminat housekeeping on an object just before the garbage collector reclaims the object's me Recall from Section 8.10 that it's unclear whether, or when, `finalize` will be call this reason, most programmers should avoid method `finalize`.
`clone`	This `protected` method, which takes no arguments and returns an `Object` refer makes a copy of the object on which it's called. The default implementation perform called shallow copy—instance-variable values in one object are copied into another o the same type. For reference types, only the references are copied. A typical overri `clone` method's implementation would perform a deep copy that creates a new obj each reference-type instance variable. Implementing `clone` correctly is difficult. F reason, its use is discouraged. Some industry experts suggest that object serialization be used instead. We discuss object serialization in Chapter 15. Recall from Chapter arrays are objects. As a result, like all other objects, arrays inherit the members of `Object`. Every array has an overridden `clone` method that copies the array. Howe the array stores references to objects, the objects are not copied—a shallow copy performed.

Fig. 9.12

`Object` methods.