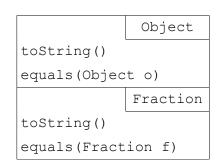
Object Interaction in Java – Inheritance & Methods

For any class of objects we define, Java automatically provides a toString method for that class. Although we did not previously discuss how this was accomplished, it can now be explained in the context of inheritance. All classes inherit from the Object class, and that includes methods such as toString. Similarly, all classes can use the equals method, which is also defined in the Object class.

Recall that the default toString method (from the Object class) returned a string containing the *identifier* of the class and a *reference* to the object location in memory. This is not particularly useful in most situations, but we can *override* this default behaviour with a custom definition, a task we undertook with the Fraction class.

The toString method for the Fraction class was written to return the numerator and denominator of the fraction. We also performed an override of the equals method, where two Fraction objects were equal if their *reduced values* were equal.



A Fraction Object

Note that both of these methods are *instance methods*. When a call is made to an instance method, Java looks for a method with the appropriate signature starting with the current class, and then working upward in the hierarchy of the object.

Example 1 – If we were to write

```
Fraction f = new Fraction (2, 3);
f.toString();
```

we would invoke the toString method of the Fraction class. If, on the other hand, we had not written a toString method for the Fraction class, this call would search upwards through the hierarchy, looking for a toString method with the same signature. It would find, and use, the default toString method defined in the Object *superclass*.