Classes and Objects in Java – Class Methods

Although we generally use instance methods when dealing with objects, it is possible, and sometimes more appropriate, to use *class methods* instead. A class method is something we have used previously, in classes constructed to hold our `main` method. The Java keyword `static` distinguishes a class method from an instance method.

For example, suppose we want a class method for our `Fraction` class that multiplies two fractions together and places the product in a new `Fraction` object. In addition, we will include the instance method times for comparison.

```
class Fraction
{
      private int num;
      private int den;

      // class method
      public static Fraction product (Fraction f1, Fraction f2)
      {
            Fraction result = new Fraction();
            total.num = f1.num * f2.num;
            total.den = f1.den * f2.den;
            return result;
      }

      // instance method
      public Fraction times (Fraction other)
      {
            Fraction result = new Fraction();
            result.num = this.num * other.num;
            result.den = this.den * other.den;
            return result;
      }
}
```

Suppose we want to call product from out main method. The syntax required to invoke this method depends on the way our program is organized – specifically, the location of our `main` method.

If the main method is in the `Fraction` class, then we would call the product method in the same way we have used in the past:

```
Fraction f = product(g, h);
```

Usually, however, we do not organize our programs this way. In most cases, our `main` method will refer to one or more classes that have been separately developed, residing in their own files. In this more common situation, where the main method is completely separate from the `Fraction` class, we would need to write:

```
Fraction f = Fraction.product(g, h);
```

You might recognize this instruction syntax from our use of other classes such as `Math` or `String`. In order to use the many class methods provided by these classes, we must explicitly identify the class in our instruction:

```
<class identifier>.<method identifier>(<parameter list>);
```

This is in contrast to the instance methods, which are invoked the same way regardless of the organization of our program.

```
Fraction f = g.times(h);
```

In this example, we will get identical results regardless of the implementation we choose. There is arguably an advantage to the instance method, since the syntax is independent of program organization, but this is a minor issue.