In this figure, X is base class and classes XY and XZ are derived from it. The XFactory class decides which of these subclasses to return depending on the arguments you give it. On the right, we define a `getClass` method to be one that passes in some value *abc,* and that returns some instance of the class **x**. Which one it returns doesn't matter to the programmer since they all have the same methods, but different implementations. How it decides which one to return is entirely up to the factory. It could be some very complex function, but it is often quite simple.

Sample Code

Let's consider a simple case in which we could use a Factory class. Suppose we have an entry form and we want to allow the user to enter his name either as "firstname lastname" or as "lastname, firstname." We'll make the further simplifying assumption that we will always be able to decide the name order by whether there is a comma between the last and first name.

This is a pretty simple sort of decision to make, and you could make it with a simple *if* statement in a single class, but let's use it here to illustrate how a factory works and what it can produce. We'll start by defining a simple base class that takes a String and splits it (somehow) into two names:

```
public class Namer {
    //base class extended by two child classes
    protected String last;     //split name
    protected String first;    //stored here

    public String getFirst() {
        return first;          //return first name
    }
    public String getLast() {
        return last;           //return last name
    }
}
```

In this base class, we don't compute any names as such, but we do provide implementations of the `getFirst` and `getLast` methods. We'll store the split first and last names in the Strings *first* and *last,* and, since the subclasses will need access to these variables, we'll make them *protected*.

The Two Subclasses

Now we can write two very simple subclasses that split the name into two parts in the constructor. In the FirstFirst class, we make the simplifying assumption that everything before the last space is part of the first name.