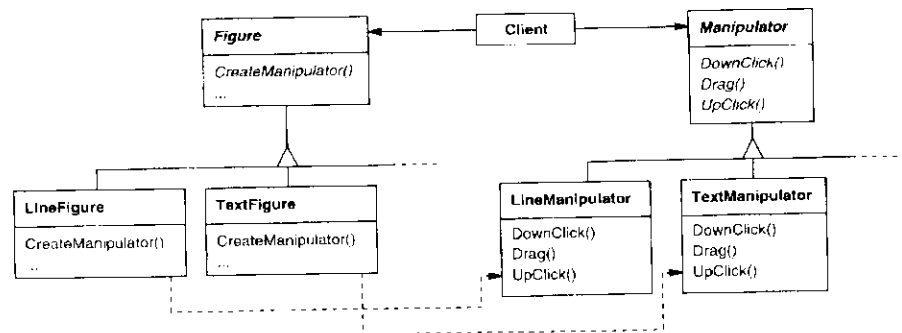


that's needed. Different figures will use different Manipulator subclasses to handle particular interactions. The resulting Manipulator class hierarchy parallels (at least partially) the Figure class hierarchy:



The Figure class provides a CreateManipulator factory method that lets clients create a Figure's corresponding Manipulator. Figure subclasses override this method to return an instance of the Manipulator subclass that's right for them. Alternatively, the Figure class may implement CreateManipulator to return a default Manipulator instance, and Figure subclasses may simply inherit that default. The Figure classes that do so need no corresponding Manipulator subclass—hence the hierarchies are only partially parallel.

Notice how the factory method defines the connection between the two class hierarchies. It localizes knowledge of which classes belong together.

Implementation

Consider the following issues when applying the Factory Method pattern:

1. *Two major varieties.* The two main variations of the Factory Method pattern are (1) the case when the Creator class is an abstract class and does not provide an implementation for the factory method it declares, and (2) the case when the Creator is a concrete class and provides a default implementation for the factory method. It's also possible to have an abstract class that defines a default implementation, but this is less common.

The first case *requires* subclasses to define an implementation, because there's no reasonable default. It gets around the dilemma of having to instantiate unforeseeable classes. In the second case, the concrete Creator uses the factory method primarily for flexibility. It's following a rule that says, "Create objects in a separate operation so that subclasses can override the way they're created." This rule ensures that designers of subclasses can change the class of objects their parent class instantiates if necessary.

2. *Parameterized factory methods.* Another variation on the pattern lets the factory method create *multiple* kinds of products. The factory method takes a