Replace Inheritance with Composition

(all of each class, one at-a-time recipe)

1a	Replace constructor with factory method on each of the concrete subclasses **C1, C2, C3…etc**	We want to get control of the instantiation of the subclasses and understand what public interface is required of the subclasses by their callers
1b	Change the type returned by the factory method to be the super type **A** (the abstract class that the concrete classes inherit from). This may cause compile breaks. Investigate and fix any compile breaks. At this stage if there is too much that is broken you might want to see if there's an easier refactoring step you can take first.	
2	Create a new concrete subclass **D** (the delegator) of the abstract class **A**. The new subclass should also have a field of type **A** – and should implement the abstract methods by delegating to its field **A**.	This new class **D** will eventually replace abstract class **A.** The new class gives us a way to migrating the subclasses **C1, C2**… bit-of-bit towards composition
3	Extract an interface from the abstract class **A** and change **D** to use the interface type instead of the abstract type **A** for its delegate field.	This interface is the strategy interface. Each of the subclasses will eventually be migrated to implement this new interface (instead of inheriting from the superclass **A**)
4	Do each concrete subclass **C1, C2..** one-by-one. Change **C1** to implement the strategy instead of extending the superclass. Change its factory method to return an instance of the delegator **D** wrapping an instance of itself. At this stage you may find other couplings between the subclasses and their superclass which you will need to address.	The subclasses can be migrated one-by-one, with successful commits between each refactoring.
5	Merge the delegating subclass **D** and the superclass **A**.	Since **D** is the only subclass of **A** there is no need for 2 separate classes.
6	Fix any remaining naming. Review the tests – should the tests be testing the delegator **D** alone, the subclasses (now strategies) alone, or a combination of the two?	

Replace Inheritance with Composition

(all classes, a bit at-a-time recipe)

1a	Replace constructor with factory method on each of the concrete subclasses **C1, C2, C3…etc**	We want to get control of the instantiation of the subclasses and understand what public interface is required of the subclasses by their callers
1b	Change the type returned by the factory method to be the super type **A** (the abstract class that the concrete classes inherit from). This may cause compile breaks. Investigate and fix any compile breaks. At this stage if there is too much that is broken you might want to see if there's an easier refactoring step you can take first.	
2	Implement the abstract methods on the superclass **A** by declaring & calling new abstract methods that are exactly the same except for their name. Change the subclasses **C1,C2..** to implement the new abstract methods instead of the original abstract methods.	This step is all about creating a separate interface between the public world of **A** and its subclasses.
3	In each subclass, look for field references from the superclass and extract them as parameters of the new abstract methods. You can lean on the compiler to check whether you have fully decoupled the subclasses by temporarily removing the inheritance relationship and seeing what is broken (and then putting the inheritance relationship back). It is important to try to make sure you have decoupled the subclasses from their superclass before proceeding to the next step.	This continues to separate the subclasses from their inheritance relationships
4	Extract the strategy interface from the superclass – the methods for the interface are the ones you created in step (2).	
5	Add a field to the superclass for an instance of the strategy interface. Use the strategy object in the superclass instead of having abstract methods, and stop the superclass from implementing the strategy interface. Change the superclass to not be abstract. At this point there will be compile breaks. Fix the compile breaks by changing the subclasses to implement the strategy interface, and change their factory methods to return a composed object – the original superclass and its subclass.	
6	Fix any remaining naming. Review the tests – should the tests be testing the superclass alone (now no longer a superclass), the subclasses (now strategies) alone, or a combination of the two?	