REFACTORING TECHNIQUES

Composing Methods

Name	Description
Extract Method	You have a code fragment that can be grouped together. Turn this fragment into a method whose name explains the purpose of this method.
Inline Method	A method's body is just as clear as its name. Put the method's body into the body of its callers and remove the method.
Inline Temp	You have a temp that is assigned to once with a simple expression, and the temp is getting in the way of other refactorings. Replace all references to that temp with the expression.
Replace Temp With Query	You are using a temporary variable to hold the result of an expression. Extract the expression into a method. Replace all references to the temp with the expression. The new method can then be used in other methods.
Introduce Explaining Variable	You have a complicated expression. Put the result of the expression, or parts of the expression, in a temporary variable with a name that explains the purpose.
Split Temporary Variable	You have a temporary variable assigned to more than once, but is not a loop variable nor a collecting temporary variable. Make a separate temporary variable for each assignment.
Remove Assignments to Parameters	The code assigns to a parameter. Use a temporary variable instead. Motivated by manipulation that might not be observed due to a changes made to the object a reference points to.
Replace Method With Method Object	You have a long method that uses local variables in such a way that you cannot apply Extract Method. Turn the method into its own object so that all the local variables becomes fields on that object. You can then decompose the method into other methods on the same object.
Substitute Algorithm	You want to replace an algorithm with one that is clearer. Replace the body of the method with the new algorithm.

Moving Features Between Objects

Name	Description
Move method	A method is, or will be, using or used by more features of another class than the class on which it is defined. Create a new method with a similar body in the class it uses most. Either turn the old method into a simple delegation, or remove it altogether.
Move field	A field, is or will be, used by another class more than the class on which it is defined. Create a new field in the target class, and change all its users.
Extract Class	You have one class doing work that should be done by two. Create a new class and move the relevant fields and methods from the old class into the new class.
Inline Class	A class isn't doing very much. Move all its features into another class and delete it.

Hide Delegate	A client is calling a delegate class of an object. Create methods on the server to hide the delegate.
Remove Middle Man	A class is doing too mcuh simple delegation. Get the client to call the delegate directly.
Introduce Foreign Method	A server class you are using needs an additional method, but you can't modify the class. Create a method in the client class with an instnace of the server class as its first method.
Introduce Local Extension	A server class you are using needs several additional methods, but you can't modify the class. Create a new class that contains those extract methods. Make this extension class a subclass or wrapper of the original.

<u>Organisation Data</u>

Name	Description
Self-Encapsulate Field	You are accessing a field directly, but the coupling to the field is becoming awkward. Create getting and setting methods for the field and use only those to access the field.
Replace Data Value with Objects	You have a data item that needs additional data or behaviour. Turn the data item into an object.
Change Value to Reference	You have a class with many equal instances that you want to replace with a single object. Turn the object into a reference object.
Change Reference to Value	You have a reference object that is small, immutable, and awkward to manage. Turn it into a value object.
Replace Array With Object	You have an array in which certain elements mean different things. Replace the array with an object that has a field for each element.
Duplicate Observed Data	You have domain data avaialble only in a GUI control, and domain methods need access. Copy the data to a domain object. Set up an observer to synchronize the two pieces of data.
Change Unidirectional Association to Bidirectional	You have two classes that need to use each other's features, but there is only a one-way link. Add back pointers, and change modifiers to update both sets.
Change Bidirectional Association to Unidirectional	You have a two-way association but one class no longer needs features from the other. Drop the unneeded end of the association.
Replace Magic Number with Symbolic Constant	You have a literal number with a particular meaning. Create a constant, name it after the meaning, and replace the number with it.
Ecnapsulate Field	There is a public field. Make it private and provide accessors.
Encapsulate Collection	A method returns a collection. Make it return a read-only view and provide add/remove methods.
Replace Record with Data Class	You need to interface with a record structure in a traditional programming environment. Make a dumb data object for the record.
Replace Type Code with Class	A class has a numeric type code that does not affect its behaviour. Replace the number with a new class.
Replace Type Code with Subclasses	You have an immutable type code that affects the behaviour of a class. Replace the type code with subclasses.
Replace Type Code with State/Strategy	You have a type code that affects the behaviour of a class, but you cannot use subclassing. Replace the type code with a state object.
Replace Subclass with Fields	You have subclasses that vary only in mehtods that return constant data. Change the methods to superclass fields and eliminate the

	subclasses.

Simplifying Conditional Expressions

Name	Description
Decompose Conditional	You have a complicated conditional (if-then-else) statement. Extract methods from the condition, then part and else parts.
Consolidate Conditional Expression	You have a sequence of conditional tests with the same result. Combine them into a single conditional expression and extract it.
Consolidate Duplicate Conditional Fragments	The same fragment of code is in all branches of a conditional expression. Move it outside of the expression.
Remove Control Flag	You have a variable that is acting as a control flag for a series of boolean statements. Use a break or return instead.
Replace Nested Conditional with Guard Clauses	A method has conditional behaviour that does not make clear the normal path of execution. Use guard clauses for all the special cases.
Replace Conditiional with Polymorphsism	You have a conditional that chooses different behaviour depending on the type of object. Move each leg of the conditional to an overriding method in a subclass. Make the original method abstract.
Introduce Null Object	You have repeated checks for null value. Replace the null value with a null object.
Introduce Assertion	A section of code assumes something about the state of the program. Make the assumption explicit with an assertion.

Making Method Calls Simpler

Name	Description
Rename method	The name of a method does not reveal its purpose. Change the name of the method.
Add parameter	A method needs more information from its caller. Add a parameter for an object that can pass on this information.
Remove parameter	A parameter is no longer used by the method body. Remove it.
Separate Query from Modifier	You have a method that returns a value but also changes the state of an object. Create two methods, one for the query and one for the modification.
Parameterise Method	Several methods do similar things but with different values contained in the method body. Create one method that uses a parameter for the different values.
Replace Parameter with Explicit Methods	You have a method that runs different code depending on the values of an enumerated parameter. Create a separate method for each value of the parameter.
Preserve Whole Object	You are getting several values from an objecta n passing these values as parameters in a method call. Send the whole object instead.
Replace Parmeter With Method	An object invokes a method, then passes the result as a parameter for a method. The receiver can also invoke this method. Remove the parameter and let the receiver invoke its method.
Introduce Parameter Object	You have a group of parameters that naturally go together. Replace them with an object.
Remove Setting Method	A field should be set at creation time and never altered. Remove

	any setting method for that field (put it into the constructor if need be)
Hide Method	A method is not used by any other class. Make the method private.
Replace Constructor with Facftory Method	You want to do more than simple construction when you create an object. Replace the constructor with a factory method.
Encapsulate Downcast	A method returns an object that needs to be downcasted by its callers. Move the downcast to within the method.
Replace Error code with Exception	A method returns a special code to indicate an error. Throw an exception instead.
Replace Exception with Test	You are throwing a checked exception on a condition the caller could have checked first. Change the caller to make the test first.

Dealing with Generalisation

Name	Description
Pull Up Field	Two subclasses have the same field. Move the field to the superclass.
Pull Up Method	You have methods with identical results on subclasses. Move them to the superclass.
Pull Up Constructor Body	You have constructors on subclassees with mostly identitcal bodies. Create a superclass constructor; call this from the subclass methods.
Push Down Method	Behaviour on a superclass is relevant only for some of its subclasses. Move it to those subclasses.
Push Down Field	A field is used only by some subclasses. Move the field to those subclasses.
Extract Subclass	A class has features that are used only in some instances. Create a subclass for that subset of features.
Extract Superclass	You have two clasess with similar features. Create a superclass and move the common features to the superclass.
Extract Interface	Several clients use the same subset of a class' interface, or two classes have part of their interfaces in common.
Collapse Hierarchy	A superclass and subclass are not very different. Merge them together.
Form Template Method	You have two methods in subclasses that perform similar steps in the same order, yet the steps are different. Get the steps into methods with the same signature, so that the original methods become the same. Then you can pull them up.
Replace Inheritance with Delegation	A subclass uses only part of the superclasses interface or does not want to inherit or does not want to inherit data. Create a field for the superclass, adjust methods to delegate to the superclass, and remove the subclassing.
Replace Delegation with Inheirtance	You're using delegation and are often writing many simple delegations for the entire interface. Make the delegating class a subclass of the delegate.

Big Refactoring

Name	Description
Tease Apart Inheritance	You have an inheritance hierarchy that is doing two jobs at once. Create two hierarchites and use delegation to invoke one from the

	other.
Converty Procedural Design to Objects	You have code written in procedural style. Turn the data records into objects, break up the behaviour, and move the behaviour to the objects.
Separate Domain from Presentation	You have GUI classees that contain domain logic. Separate the domain logic into separate domain classes.
Extract Hierarchy	You have a class that is doing too much work, at least in part through many conditional statements. Create a hierarchy of classes in which each subclass represents a special case.

Signs of code that might need refactoring

Name	Description	Solution(s)
Duplicated Code	Demonstrated if cut-and-paste is done more than once.	Extract Method, Extract Class, Pull UP Method, Form Template Method
Long Method	If you are scrolling screen on screen when going through the same method, you might have a method too long	Extract Method, Replace Temp with Query
Large Class	May show up as too many instance variables. You might need to better decompose to serveral classes or else.	Extract Class, Extract Subclass, Extract Interface, Replace Data Value with Object
Long Parameter List	Parameter lists are much too large or if you have optional parameters. Demonstrated if some of the parameters are not being used.	Replace Parameter with Method, Introduce Parameter Object, Preserve Whole Object
Divergent Change	When a single class is changed too frequently when changes are requested. Functionality may need to be extracted into separate classes or methods.	Extract Class
Shotgun Surgery	When a change in the system environmetn changes occurs, you have to edit many many classes.	Move Method, Move Field, Inline class
Feature Envy	When another class is depending on another one to provide a certain functionality, another class might actually need to perform that functionality.	Move Method, Move Field, Extract Method
Data Clumps	These are scattered bits of data that belong together but are persisted in different classes/methods.	Extract Class, Introduce Parameter Object, Preserve Whole Object
Primitive Obsesssion	The relunctance to move to objects and keep fields as separate primitive types	
Switch Statements	Too many switch statements show procedural statements.	Replace Conditional with Polymorphism, Replace Type Code with Subclasses, Replace Type Code with State/Strategy, Replace Parameter with Explicit Methods, Introduce Null Objects
Parallel Inheritance Hierarchies	Special case of shotgun surgery in which everytime you subclass one class you will have to subclass another.	Move Method, Move Field

Lazy Class	A class that doesn’t pull its weight.	Inline Class, Collapse Hierarchy
Speculative Generality	The use of abstract classes or super methods which must be overriden to be useful. Signs to look for if the only things using it are test cases.	Collapse Hierarchy, Inline Class, Remove Parameter, Rename Method
Temporary Field	An instance variable that is set only in certain circumstances.	Extract Class, Introduce Null Object
Message Chains	Demonstrated when a client asks one object for antoher object, which the client then asks for yet another object, and so on. There is strong coupling here.	Hide Delegate
Middle Man	If a method is simply passing parameters to another one without providing real functionality, may be a good candidate for refactoring	Remove Middle Man, Inline Method, Replace Delegation with Inheritance
Inappropriate Intimacy	When classes deal with private or protected variables in another class far too frequently. Common on inheritance hierarchies.	Move Method, Move Field, Chagne Bidirectional Association to Unidirectional, Replace Inheritance with Delegation, Hide Delegate
Alternative Classes with Different Interfaces	When another class doing the same job is created just for a different signature. This should be replaced with overloading and merging to a single class.	Rename Method, Move Method
Incomplete Library Class	When library functionality doesn’t provide the complete set required (usually when you are given a library to work with external to your organisation or project).	Introduce Foreign Method, Introduce Local Extension
Data Class	Dumb data holders may provide better functionality if given more things to do.	Move Method, Encapsulate Field, Encapsulate Collection
Refused Bequest	When a subclass doesn’t want methods inherent in the superclass.	Repleace Inheritance with delegation
Comments	While comments are great, refactoring may make most comments superfluous. If a method is heavily commented, it may need refactoring	Extract Method, Introduce Assertion

“Any fool can write code that a computer can understand. Good programers write code that humans can understand.”

“Three strikes and you refactor.”