**Chapter 3 – 4 Summary**

**Chapter 3 Bad Smells in Code**

duplicated code is one number in the stink parade. If you see the same code structure in more than one place, you can be sure that your program will be better if you find a way to unify them. The simplest duplicated code problem is when you have the same expression in two methods of the same class. Then all you have to do is Extract Method and invoke the code from both places

**Eliminate the duplicate using the following method:**

1. **Extract Method**
2. **Form Template Method**

**Extract Method** is to separate the similar bits from the different bits.

**Form Template Method** you can use this if the methods do the same thing with a different algorithm

Using Extract Class in one class and then use the new component in the other. Another possibility is that the method really belongs only in one of the classes and should be invoked by the other class or that the method belongs in a third class that should be referred to by both of the original classes. You have to decide where the method makes sense and ensure it is there and nowhere else.

Long Method

The object programs that live best and longest are those with short methods. Programmers new to objects often feel that no computation ever takes place, that object programs are endless sequences of delegation.

The early days of programming people have realized that the longer a procedure is, the more difficult it is to understand. Older languages carried an overhead in subroutine calls, which deterred people from small methods. Modern OO languages have pretty much eliminated that overhead for in-process calls.

Ninety-nine percent of the time, all you have to do to shorten a method is Extract Method. Find parts of the method that seem to go nicely together and make a new method.

Large Class

When a class attempts to do too much, it typically manifests as an excessive number of instance variables. Duplicating code is unavoidable when a class has too many instance variables. Extract Class can be used to gather together a large number of variables. Choose variables that will work well together in the component that is appropriate for each.

If your huge class is a GUI class, you may need to divide data and behaviour object. This may necessitate storing some duplicate data in both places as well as keeping the data in both places sync. This is suggested by Duplicate Observed Data.

Long Parameter List

Long parameter lists are difficult to grasp because they become inconsistent and difficult to use, and because they are always changing as additional data is required. Most modifications are erased by passing objects because it is much more likely that you will only need to make a few of queries to get a new piece of data.

Use Replace Parameter with Method when you can get the data in one parameter by making a request of an object you already know about. This object might be a field or it might be another 66 parameter.

Use Preserve Whole Object to take a bunch of data gleaned from an object and replace it with the object itself. If you have several data items with no logical object, use Introduce Parameter Object.

Divergent Change

Divergent change happens when one class is frequently altered in various ways for distinct purposes reasons.

Shotgun Surgery

* Similar to divergent change but is the opposite.
* You can tell when you have to make a lot of small modifications to a lot of different classes every time you make a change.
* Divergent change is one class that suffers many kinds of changes, and shotgun surgery is one change that alters many classes.

Data Clumps

* Data items tend to be like children; they enjoy hanging around in groups together

Steps to look where the clumps appear:

1. Extract Class
2. Introduce Parameter Object or Preserve Whole Object

Primitive Obsession

There are two kinds of Data:

1. Records types allow you to structure data into meaningful groups.

* Records always carry a certain amount of overhead.

1. Primitive types are your building blocks.