**Refactoring**

**Improving the Design of Existing Code**

1. **Chapter 1: Refactoring: The First Example**

* When you have to add a feature to a program but the code is not structured in convenient way => Refactor the program to make it easy to add a feature first, then add features.
* Before you start refactoring, make sure you have a solid suite of test. This test must be self-checking
* When working with long function, find point that can separate different part of function. Then turn a chunk of code into its own function *(Extract Function)*
* Refactoring change programs in a small test. If we make mistake, it easy to find what is the problem
* A fool can write code for computer to understand. A Good programmer can write code that human can understand easily
* Get Rid of temporary variable because it create a lot of local scope, that make refactor is hard *(Replace Temp with Query, Inline Variable, Change Function Declaration)*
* If loop statement have number of local variables => can split it and move declaration next to that loop for readability *(Split Loop, Split Statement*)
* If we need to think about performance => refactor first, then care about perf after. Because refactor code is easy to adapt

1. **Chapter 2: Principle in Refactoring**
2. **Chapter 3: Bad smells in Code**
3. **Chapter 4: Building Test**
4. **Chapter 5: Introduce The catalog**
5. **Chapter 6: A First set of Refactoring**
6. **Chapter 7: Encapsulation**
7. **Chapter 8: Moving feature**
8. **Chapter 9: Organizing Data**
9. **Chapter 10: Simplifying Condition Logic**
10. **Chapter 11: Refactoring APIs**
11. **Chapter 12: Dealing with Inheritance**
12. **Techniques**

* Extra Function:
  + Problem: You have a code fragment that can be grouped together.
  + Solution: Move this code to a separate new method (or function) and replace the old code with a call to the method.

**void** **printOwing**() {

printBanner();

// Print details.

System.out.println("name: " + name);

System.out.println("amount: " + getOutstanding());

}

* + Convert into:

**void** **printOwing**() {

printBanner();

printDetails(getOutstanding());

}

**void** **printDetails**(**double** outstanding) {

System.out.println("name: " + name);

System.out.println("amount: " + outstanding);

}

* Replace Temp with Query:
  + Problem: You place the result of an expression in a local variable for later use in your code.
  + Solution: Move the entire expression to a separate method and return the result from it. Query the method instead of using a variable. Incorporate the new method in other methods, if necessary.

**double** **calculateTotal**() {

**double** basePrice = quantity \* itemPrice;

**if** (basePrice > 1000) {

**return** basePrice \* 0.95;

}

**else** {

**return** basePrice \* 0.98;

}

}

**double** **calculateTotal**() {

**if** (basePrice() > 1000) {

**return** basePrice() \* 0.95;

}

**else** {

**return** basePrice() \* 0.98;

}

}

**double** **basePrice**() {

**return** quantity \* itemPrice;

}

* Inline Variable:
  + Problem: You have a temporary variable that’s assigned the result of a simple expression and nothing more.
  + Solution: Replace the references to the variable with the expression itself.

**boolean** **hasDiscount**(Order order) {

**double** basePrice = order.basePrice();

**return** basePrice > 1000;

}

**boolean** **hasDiscount**(Order order) {

**return** order.basePrice() > 1000;

}

* Change Function Declaration:
  + Problem: Remove Parameter of function