# **Refactoring**

Makes code understandable and easy to extend

Makes it quicker to create complicated system

You remove excess code so it is easier to understand & modify

By refactoring code you can better understand other codes

You eliminate the having to remember what old code does

You can write code quicker

Bad smells are common design problem

Bad design is normally unclear, complicated or duplicated

|  |  |
| --- | --- |
| What are bad smells | |
| Duplicated codes | Alternative classes with different interfaces |
| Long methods | Lazy classes |
| Complex conditional statement | Large classes |
| Primitive obsession وسواس اولیه | Switch statement |
| Indecent exposure | Combinatorial explosions انفجار ترکیبی |
| Solution sprawl | Oddball solution |

More constructors -> more problems

General purpose constructors save the day.

If you have to change the class to add features your design is bad.

**Replace conditions with polymorphism and strategy pattern**

I actually could eliminate the subclass

That would make the design better because it reduces coupling

**Eliminate duplicate codes with template method**

Generalization is used to transform specific code into general purpose code.

When objects perform similar steps in the same order:

Implement the similar code in a template method

Allow sub class to override the behavior that varies.

**Replace implied primitive trees with the composite pattern**

The composite pattern can make any type of tree structure.

* The Builder pattern allows you to build complex objects in a series of steps.
* Define an object class of a specific type (sandwich)
* Create an abstract class that contains all of the methods that each class of type sandwich must implement
* The director asks for the type of sandwich you want, initializes it and provides it.

Complete code = better understanding code

**Replace conditionals with command pattern**

Refactoring with visual studio

* [Rename](https://docs.microsoft.com/en-us/cpp/ide/refactoring/rename?view=vs-2017) (Ctrl+R Ctrl+R)
* [Extract Function](https://docs.microsoft.com/en-us/cpp/ide/refactoring/extract-function?view=vs-2017)(Ctrl+R Ctrl+M) **Edit > Refactor > Extract Function (Experimental)**.
* [Implement Pure Virtuals](https://docs.microsoft.com/en-us/cpp/ide/refactoring/implement-pure-virtuals?view=vs-2017)(Ctrl+.) **Quick Actions and Refactorings** menu and select **Implement all Pure Virtuals for class 'ClassName'**
* [Create Declaration / Definition](https://docs.microsoft.com/en-us/cpp/ide/refactoring/create-declaration-definition?view=vs-2017)(Ctrl+.) **Quick Actions and Refactorings** menu and select **Create Declaration / Definition**
* [Move Function Definition](https://docs.microsoft.com/en-us/cpp/ide/refactoring/move-definition-location?view=vs-2017)(Ctrl+.) **Quick Actions and Refactorings** menu and select **Move Definition Location**
* [Convert to Raw String Literal](https://docs.microsoft.com/en-us/cpp/ide/refactoring/convert-to-raw-string-literal?view=vs-2017)(Ctrl+.) **Quick Actions and Refactorings** menu and select **Convert to Raw String Literal**
* [Change Signature](https://docs.microsoft.com/en-us/cpp/ide/refactoring/change-signature?view=vs-2017) **Edit > Refactor > Change Signature**.