How TDD changed my life

Write a failing test first  
Write simplest code to make it pass  
Refactor

Not just unit testing

Testing is not a separate process  
Happens many times every minute

First change: Code breaking code  
-Looney Tune Code: Code breaking code, circle

Second change: Hard to reproduce bugs

Solid Design Patterns for mere mortals

S – Single Responsibility Principle  
 -Do one thing well  
O – Open Closed Principle  
 -Open for extension, Closed for modification  
L – Liskov Substitution Principle  
 -Derived classes can stand in for Base classes  
I – Interface Segregation Principle  
 -Make interfaces fine grained and client specific  
D – Dependency Inversion Principle  
 -Dependent on abstractions, not concrete implementations  
 -New is glue  
-----------  
Additional consideration  
-Clipboard inheritance is an anti-pattern  
-Boy scout principle  
 -clean up after yourself  
 -clean up after others  
-YAGNI – Ya aint gonna need it  
-Separation of concerns  
 -Focusing ones attention upon some aspect – Edsger Dijkstra

Design Patterns – General reusable solution to a common problem  
 -Conceptual  
 -Defined by purpose and structure  
 -Method of communication  
 -Support SOLID Dev  
 -NOT CODE  
Types of Design Patterns  
-**Creational**  
 -Singleton  
 -Ensures class has only one instance with a single access point  
 -Simple Factory  
 -Encapsulates Object Creation in one place  
 -Factory Method  
 -Uses methods to create objects without specifying exact class  
-**Structural**  
 -Adapter  
 -Converts the interface of a class into another interface the client expects   
 -Façade   
 -Provides a simplified interface to a larger body of code  
 -Decorator  
 -attaches additional responsibilities to an object a runtime without effecting other objects of the same class  
-**Behavior**  
 -Command  
 -Encapsulates a request as an object  
 -Strategy

**“Head First Design Patterns” – Get this book**

**Applying design patterns to solve everyday problems**Bridge Pattern  
 -Uses an interface to isolate your code from different implementations  
 -allows to easily switch out implementations behind the bridge  
 -allows testing without dependency  
 -Sets us up to use dependency injection

Decorator Pattern  
 -Wrap one object inside another  
 -Implement the same interface as the object they are decorating  
 -Objects can be decorated multiple times  
 -Useful for crosscutting concerns

Chain of responsibility   
 -Series of multiple handlers that can each handle a request

Template Pattern  
 -Define base class, This defines high level steps  
 -