* 1. Object Oriented Design

See Design Patterns document.

# Functional Programming

Functional programming is a style of programming where solutions are simple, isolated functions, without any side effects outside of the function scope.

INPUT -> PROCESS -> OUTPUT

Functional programming is about:

1) Isolated functions - there is no dependence on the state of the program, which includes global variables that are subject to change

2) Pure functions - the same input always gives the same output

3) Functions with limited side effects - any changes, or mutations, to the state of the program outside the function are carefully controlled

One of the core principle of functional programming is to **not change things**. Changes lead to bugs. It's easier to prevent bugs knowing that your functions don't change anything, including the function arguments or any global variable.

in functional programming, changing or altering things is called mutation, and the outcome is called a side effect. A function, ideally, should be a **pure function**, meaning that it does not cause any side effects.

principles for functional programming:

1) Don't alter a variable or object - create new variables and objects and return them if need be from a function.

2) Declare function arguments - any computation inside a function depends only on the arguments, and not on any global object or variable.