**OOP vs FP**

**Composition vs Inheritance:**

Inheritance: super class extended to smaller pieces

Composition: smaller pieces combined to build something bigger

Structure of code: what it is has or what it does?

**Solution to inheritance problem**

Solve the inheritance problem by removing all the methods from class and using functions, add the necessary functions where required.

Small small components/functions combined into different boxes. New component can be added easily

Structure of code: what it is?

**Inheritance problem**

**Tight coupling problem** (parent and child) – having a small change in a parent class has rippling effect in all children

**Fragile base class problem** – due to tight coupling problem it may break the code

**Hierarchy class problem** - > unused methods are inherited

Fp(paradigm-rules)

Avoiding side effects and writing pure functions

Code is a combination of functions

Data is immutable – functions cannot change outside environment

Output value of functions simply depends on the given args

Functions are first class citizens

Functions manipulate data structures like arrays... Trees

Oop(paradigm-rules)

Orgainizing the code into units

Object is a box containing information and operations

Information – attributes or states

Opeations on state – methods

Objects are first class citizens

Fp

Many operations on fixed data

State less – don’t modify data

Functioins are pure – no side effects – don’t make an impact on codes outside

More declarative – what we want to be doing

Works well on lot of operations

Data and operations are kept separately for clear understanding

Oop

Few operations on common data

Statefull – modify data

Side effect – methods manipulate internal state

Imperative – more on how it should be doing

Works well on objects with many things but less operations

Bring together data and operations into a single box – orgranized and clear

**JavaScript allows both the paradigm**