# INTRODUCING FRAMECHAINS 0

-YON-
Columbia University, 2024

## Abstract

The world was revolutionized by the invention of Blockchains by a paper written by an anonymous author under the alias Satoshi Nakamoto. The paper offered critical solutions that enabled a decentralized cash system that registered transactions into blocks and chained them into a central immutable ledger, hence the name Blockchain.

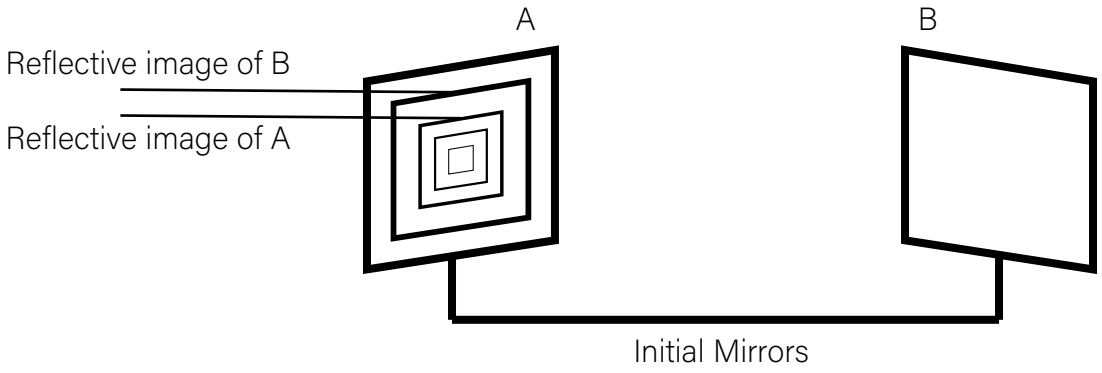
This paper offers an iteration of the ledger which was described as a Linked List, into an a new datastructure, Frames. A Frame is a datastructure representing two mirrors facing each other. When two mirrors are faced, the reflection of the one is observable in the other, which contains the reflection of the other, through which the one is observable. As such, it creates an infinite loop of reflections that are naturally linked to each other. This paper offers an implementation of this phenomenon in Java and allow each unit to hold data to make them a utilizable component for a ledger creating a chain of Frames - Framechains.

## Frames

Below is a basic implementation of a Frame.

```
public class Frame {
    private int frames; //the number of frames to be contained inside a single Frame
    private Frame[] contained; //A recursive call to the class Frame as an array
    private int value; //Data to be contained in the Frame instance, integer in this instance
    private boolean activated; //Activation handle controller

    public Frame(int frames) {
        this.frames = frames;
        contained = new Frame[frames];
        activated = false;
    } //Constructs a frame with a certain number of reflective instances, only when the Frame is activated
is when the internal Frames are constructed.

    public int activate() {
        for (int i = 0; i < frames; i++) {
            Frame newFrame = new Frame(frames);
            contained[i] = newFrame;
        }
        activated = true;
        return frames;
    } //Handle to prevent an infinite recursive call

    public Frame[] enterFrame() {
        if (!activated) {
            System.out.println("Trying to reach an unactivated Frame");
            System.exit(1);
        }
        return contained;
    } //A means to access the internal Frames of an activated Frame.

//written by -YON-
//This Frame, intentionally doesn't include an exitFrame tool or function to make it uni-directional.
//This Frame, intentionally doesn't include a setValue tool or function to not limit readers to any type of
data
//Basis for FrameChain - an iteration over the Blockchain ledger.
```

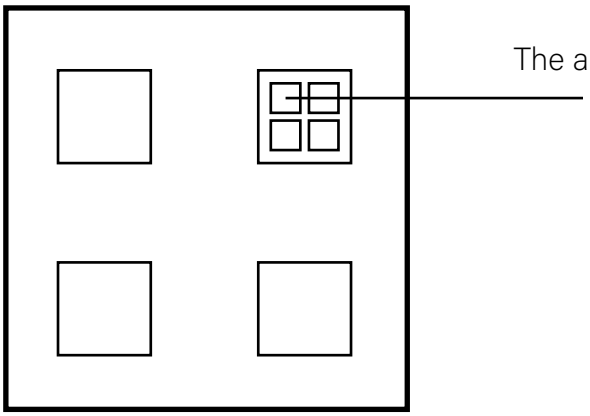
The Frame uses a controlled recursion without a base case to allow the implementation of the infinte reflective nature of mirrors facing each other. If only two mirrors faced each other, the image at a certain level would only be a single frame. Here is a diagram:



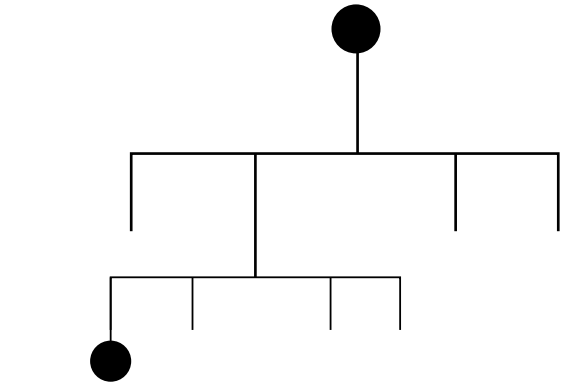
Initial Mirrors

Inside a computer system, the number of images can be controlled, as done by the variable frames in the above code. Thus, below is a simple execution of the above Class with 4 images inside the Frame.

```
public static void main(String[] args) {
    Frame myFrame = new Frame(4);
    myFrame.activate();
    Frame[] internalFrames = myFrame.enterFrame();
    internalFrames[1].activate();
    Frame[] internalInternalFrames = internalFrames[1].enterFrame();
    internalInternalFrames[0].activate();
}
```



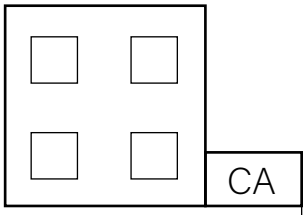

If the above Frame was to be represented into a tree, it would look like the following -



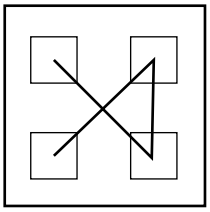
# Possible Variations of Frames

## Configurable Frames

A configuration array is included with the data stored in each Frame to give the included Frames a certain sequence.



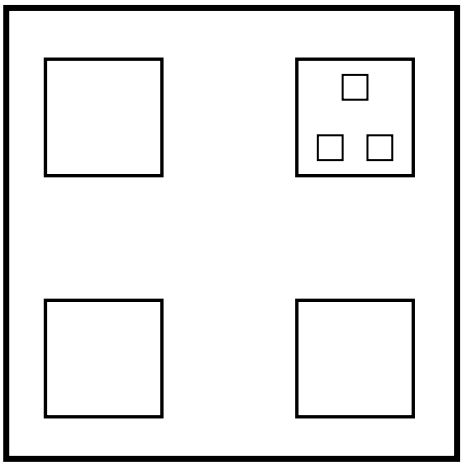

A configuration Array



[1, 4, 2, 3]

## Varying Frames

A Frame structure with varying number of Internal Frames in different Frames.



A Framechain ledger with varying Frames can allow the possibility of Branching out into numerous separate systems using Varying Frames. This can allow the construction of complex ledgers that can work as a complex Data system.