```swift
//
//  RAM.swift
//  SimulatedPageAllocation
//
//  Created by Brandon Baars on 4/3/18.
//  Copyright © 2018 Brandon Baars. All rights reserved.
//

import Foundation

class RAM {

    public var RAM: [Frame]
    private var freeRAM: [Frame]
    private var RAMSize: Int

    init(sizeOfRam size: Int) {

        RAM = [Frame]()
        freeRAM = [Frame]()
        RAMSize = size

        for index in 0..<size {
            let newFrame = Frame(frameSize: 512.0, isOccupied: false, index:
             index, type: .empty, data: nil)
            RAM.append(newFrame)
            freeRAM.append(newFrame)
        }
    }

    public func changeFrameSize(withSize size: Double) {
        for frame in 0..<RAM.count {
            let _ = RAM[frame].setFrameSize(newSize: size)
        }
    }

    public func addProcessToRam(withProcess process: ProcessData) {

        // check amount of frames needed vs. what's actually free
        let totalFrames = process.numOfDataPages + process.numOfCodePages
        freeRAM = self.RAM.filter {!$0.isOccupied}

        var codeCount = 0
        var dataCount = 0

        if freeRAM.count < totalFrames {
            return
        } else {

            // look through all free ram and add the process
            // add the code and data to RAM and update the
```

```
            // values accordingly.
            for (index, frame) in freeRAM.enumerated() {
                if index >= totalFrames {
                    break
                } else {
                    RAM[frame.index].isOccupied = true
                    RAM[frame.index].data = process

                    // if it's our code part, add our code
                    // part to the associated frame
                    if codeCount < process.numOfCodePages {
                        RAM[frame.index].type = .code
                        process.addFrameToPageTable(withFrame:
                         RAM[frame.index])
                        codeCount += 1

                    // add our data part to the associated frame
                    } else {
                        RAM[frame.index].type = .data
                        process.addFrameToPageTable(withFrame:
                         RAM[frame.index])
                        dataCount += 1
                    }
                }
            }
        }
    }

    // reset each frame in our array
    public func resetRAM() {
        for index in 0..<RAMSize {
            RAM[index].resetFrame(withIndex: index)
        }
    }
}
```