```
import Compressor
import UIKit
//: # Compressor Operation
//: Continuing from the challenge in the previous video, your challenge
    for this video is to use an `NSOperationQueue` to decompress a
    collection of compressed images.

//: The `ImageDecompressor` class is as before
class ImageDecompressor: NSOperation {
  var inputPath: String?
  var outputImage: UIImage?

  override func main() {
    guard let inputPath = inputPath else { return }

    if let decompressedData = Compressor.loadCompressedFile(inputPath) {
      outputImage = UIImage(data: decompressedData)
    }
  }
}

//: The following two arrays represent the input and output collections:
let compressedFilePaths = ["01", "02", "03", "04", "05"].map {
  NSBundle.mainBundle().pathForResource("sample_\($0)_small", ofType:
      "compressed")
}
var decompressedImages = [UIImage]()

//: Create your implementation here:

//: Create the queue with the default constructor
let decompressionQueue = NSOperationQueue()
let appendQueue = NSOperationQueue()
appendQueue.maxConcurrentOperationCount = 1

//: Create a filter operations for each of the iamges, adding a
    completionBlock
for compressedFile in compressedFilePaths {

  let decompressionOp = ImageDecompressor()

  decompressionOp.inputPath = compressedFile

  decompressionOp.completionBlock = {
    guard let output = decompressionOp.outputImage else { return }
    appendQueue.addOperationWithBlock {
      decompressedImages.append(output)
    }
  }
  decompressionQueue.addOperation(decompressionOp)
}

//: Need to wait for the queue to finish before checking the results
decompressionQueue.waitUntilAllOperationsAreFinished()
```

```
//: Inspect the decompressed images
decompressedImages

//: # Erratum
//: The code in this video assumes that Swift Array is threadsafe. This
    is an incorrect assumption, and therefore you might notice that the
    size of filteredImages changes each time you run the code. This is
    due to a race condition.
//: The problem comes from two separate threads attempting to append
    images to the array at the same time, and the solution is to prevent
    that from happening. To that end add a new serial NSOperationQueue
    whose sole responsibility is to manage additions to the images
    array.
/// let appendQueue = NSOperationQueue()
/// appendQueue.maxConcurrentOperationCount = 1

//: Then push the append() calls onto this serial queue:

/// appendQueue.addOperationWithBlock {
///     filteredImages.append(output)
/// }

//: The fact that appendQueue is serial means that only one append() can
    occur at once, making it impossible for the race condition to occur.
```