在影像處理軟體中處理複數個工作項目時，如果每個項目的運算量或記憶體用量較大，又可能會在運算過程中動態添加/移除工作項目的時候，其中一種做法是，將所有的工作項目排入佇列 (queue) 中，固定只處理佇列中的第一個項目，處理完成的項目就會從佇列中被移除，而新增的工作項目則會附加到佇列的最後，以此來達成先加入的工作項目就先被處理 (first-in-first-out) 的效果。

請使用***照片處理function***以及***array***段落定義的array，實作一個class來完成上述的佇列結構。處理照片佇列的過程中，必須要確保同一時間，只能夠對一張照片進行處理；此外，先被選擇的照片必須先被處理。
在這個佇列結構對應的array中，每個element主要的內容，是待處理照片的路徑，而不是待處理的影像處理程序，至於是否需要其他內容，可以自行決定；此佇列結構的class需要有以下公開的方法 (method)，私有的屬性 (property)或方法可自行添加：

方法	**意義**
isEmpty(): Boolean	檢查佇列是否爲空(表示已經沒有任何待處理的照片，背景執行緒也沒有處理任何照片了)。
addToQueue(imagePath: String)	使用者選取了某張照片時，會呼叫此方法，並傳入該照片的路徑(imagePath)。此照片將會被排入待處理的佇列中
removeFromQueue(imagePath: String)	使用者反選取了某張照片時，會呼叫此方法，並傳入該照片的路徑(imagePath)。此照片將從待處理的佇列中被移除 請注意，如果即將被移除的照片已經進行影像處理的程序，從佇列中移除照片，並不會中止對應的處理程序

在程式中，外部的程式碼只會透過addToQueue/removeFromQueue來對佇列的內容進行修改，並根據isEmpty的值來判斷，是否所有選擇的照片都已經處理完成。因此，必須自行決定在class中觸發影像處理的時機。

*撰寫的pseudo code中，可以使用一般程式語言通用的if/else、switch/case、while或for loop語法，也可以自定義function來將邏輯判斷或運算式包裝起來。
宣告變數時請用let，如果變數值足以表明該變數型別，就不必特別宣告型別。宣告class，可參考*array***段落的範例
未列在照片處理function***以及***array***中的function，除非是自定義，否則不能使用。
****假設只有影像後處理會在背景執行緒執行，其餘程式碼皆在主執行緒上，因此不必考慮class中的不同方法對於共同的變數會發生競爭情況 (race condition)。

照片處理function：

定義	回傳

function imageProcess(imagePath: String, callback: function)
 	
在主執行緒呼叫此function後，影像將會在背景執行緒中進行處理。
imagePath爲待處理的影像路徑。
callback爲背景執行緒完成影像處理後觸發的function，會回到主執行緒來執行。
imageProcess爲全域函數，可在程式中的任意地方直接呼叫。
 	
範例：
`//範例中的x是用來說明主執行緒與背景執行緒的執行順序`
 	
`let imagePath = 'path/to/image';`
`let x = 0;`
`//主執行緒呼叫imageProcess，背景執行緒開始進行`
`//影像處理`
`imageProcess(imagePath, function() {`
`  //背景執行緒執行完後，觸發callback function`
`  x = x + 1; //x = 2`
`});`
`//主執行緒完成imageProcess呼叫後，立刻執行`
`//底下程式碼`
`x = x + 1; //x = 1`
	無回傳值

array：

與C/C++或Java語言中嚴格的定義不同，此處的array長度可以在程式執行過程中動態增加或減少，不必考量初始化就需指定array長度的限制，但若存取的index超過array的長度，仍然會出錯。

假設array有以下的屬性/方法可使用：

屬性	**意義**
length: int	array長度

方法	**意義**
insertAt(element: Element, index: int)	在array中指定的index插入新的元素
removeFrom(index: int)	從array中指定的index移除元素

範例：

```
//宣告SampleObject，作爲下述陣列的元素型別
class SampleObject {
    let name: String;
    let age: int;
    constructor(name: String, age: int) {
        this.name = name;
        this.age = age;
    }
}
//宣告空陣列，每個元素的型別爲SampleObject
let sampleArray = [SampleObject];
//sampleArray.length = 0

//在index = 0 插入第1個元素
sampleArray.insertAt(new SampleObject('A', 10), 0);
//sampleArray.length = 1
//sampleArray[0] = { 'name': 'A', 'age': 10 }

//在index = 0 插入第2個元素
sampleArray.insertAt(new SampleObject('B', 20), 0);
//sampleArray.length = 2
//sampleArray[0] = { 'name': 'B', 'age': 20 }
//sampleArray[1] = { 'name': 'A', 'age': 10 }

//在index = 1 插入第3個元素
sampleArray.insertAt(new SampleObject('C', 30), 1);
//sampleArray.length = 3
//sampleArray[0] = { 'name': 'B', 'age': 20 }
//sampleArray[1] = { 'name': 'C', 'age': 30 }
//sampleArray[2] = { 'name': 'A', 'age': 10 }

//移除index = 1的元素
sampleArray.removeFrom(1);
//sampleArray.length = 2
//sampleArray[0] = { 'name': 'B', 'age': 20 }
//sampleArray[1] = { 'name': 'A', 'age': 10 }
```