* **Service:**
  + Members:
    - **Event logger** e – view – helps us see if the service did what we wanted it to do
    - **Installer**
    - **Service Status**
    - **ILogging modal** logger
    - **Server**(logger)
  + The Service is the main – create server and logging model in the onStart()

Logger.onMsgEevnt += onMsg (after the service creates the logger in onStart())

* + **Void onMsg(msg)** {e.WriteEntry(msg)}
* **Logging Modal:**
  + Member:
    - Public Event: onMsgEvent (member)
  + **log(msg)** {onMsgEvent.Invoke(this, msg)}
* **Image Modal:** 
  + **addFile** – check if outputDir exists, if not – create it. Check date of picture from path, move picture from path to directory outputDir in relevant year and month directories.

Result – false/true

Return value – string – specific error

* + The class Image Model will have more functions besides addFile, but addFile is the only one in the interface
* **Server**:
  + Members:
    - **Ilogger**
    - **IController(**ImageModel)
    - **Event onCommand**
    - **Dictionary** of commands (right now just has closeHandler)
  + Server creates IImage Modal – class that handles file system (server will pass this to handlers etc.)
  + Every handler uses the same controller
  + In this targil we don’t need to create TCP
  + Server sends commands to all of the handlers, each handler checks if the command is relevant to it (checks if the path is in its directory)
  + **createHandler(directory)** {Handler h = new Handler(directory, controller); onCommand += h. onCommandReceived (); h.onClose += onCloseServer}
  + **sendCommand()** {onCommand(“\*”, CloseHandler)} – closes handlers
  + **onCloseServer(sender, …)** {h = sender; onCommand -= h. onCommandReceived; onCommand -= h.onCloseServer;} – handler will call this function to tell server it closed
* **Handler**
  + Members:
    - **Directory path**
    - **Controller**
    - **FileSystemWatcher**
    - **event onClose**
  + when handler realize that there’s a new picture in folder, FileSystemWatcher will invoke an event (automatically) – need to wrap it in a command and call controller.executeCommand(command)
  + **onCommandReceived (command)** {check if command is meant for its directory, if yes – handle command (for now will just be to close handler)}
  + **closeHandler()** – close FileSystemWatcher and invoke onClose event
* **Controller**
  + Member:
    - **IImage Modal**
  + **executeCommand()** {Decipher command and call relevant function in image modal in new task (thread)}

appConfig – has paths to directories. service reads from appConfig.

(can check if this works by using the directory downloads in our computer)