* **ControlRecognitionProtocol** => already done by Moritz with testing the constructors

(Rm: object\_id’s are now pid’s of the actors since every object we want to display is itself an actor described with an

ProcessID (=pid)

* + **All Constructor of the message classes:**
  + CreateNewVirtualObject(int \_messageID, PID \_pidSender)
  + SetActiveVirtualObject(int \_messageID, PID \_pidSender)
  + SetInactiveVirtualObject(int \_messageID, PID \_pidSender)
  + StartVirtualObject(int \_messageID, PID \_pidSender)
  + StopVirtualObject(int \_messageID, PID \_pidSender)
  + KillVirtualObject(int \_messageID, PID \_pidSender)
  + RequestAllVirtualObjects(int \_messageID, timespan Time, PID \_pidSender)

If one actor does not reply in the given timespan Time we then have an error or we do whatever we want

* + *ClickVirtualObject(int \_messageID, PID \_pidSender) // after step one*
  + **All responds:**
  + RespondCreateNewVirtualObject(int \_messageID, PID \_pidSender);
  + RespondSetActiveVirtualObject(int \_messageID, PID \_pidSender)
  + RespondSetInactiveVirtualObject(int \_messageID, PID \_pidSender)
  + RespondStartVirtualObject(int \_messageID, PID \_pidSender)
  + RespondStopVirtualObject(int \_messageID, PID \_pidSender)
  + RespondKillVirtualObject(int \_messageID, PID \_pidSender)
  + RespondRequestAllVirtualObjects(int \_messageID, timespan Time, PID \_pidSender)

If one actor does not reply in the given timespan Time we then have an error or we do whatever we want

* + *RespondClickVirtualObject(int \_messageID,* PID \_pidSender*) // after step one*
  + **All fails:**
  + FailToCreateNewVirtualObject(int \_messageID, PID \_pidSender);
  + FailToSetActiveVirtualObject(int \_messageID, PID \_pidSender)
  + FailToSetInactiveVirtualObject(int \_messageID, PID \_pidSender)
  + FailToStartVirtualObject(int \_messageID, PID \_pidSender)
  + FailToStopVirtualObject(int \_messageID, PID \_pidSender)
  + FailToKillVirtualObject(int \_messageID, PID \_pidSender)
  + FailToRequestAllVirtualObjects(int \_messageID, PID \_pidSender, timespan Time)

If one actor does not reply in the given timespan Time we then have an error or we do whatever we want

* + *FailToClickVirtualObject(int \_messageID,* PID \_pidSender*) // after step one*
* **VisualizationCodeObjectProtocol**
  + **All Constructor of the message classes:**
  + StartVirtualObject(int \_messageID, PID \_pidSender)
  + StopVirtualObject(int \_messageID, PID \_pidSender)
  + SetActiveVirtualObject(int \_messageID, PID \_pidSender)
  + SetInactiveVirtualObject(int \_messageID, PID \_pidSender)
  + KillVirtualObject(int \_messageID, PID \_pidSender)
  + RequestCheckVirtualObject(int \_messageID, PID \_pidSender)
  + *ClickVirtualObject(int* \_messageID*,* PID \_pidSender*) // after step one*
  + **All responds:**
  + RespondStartVirtualObject(int \_messageID, PID \_pidSender)
  + RespondStopVirtualObject(int \_messageID, PID \_pidSender)
  + RespondSetActiveVirtualObject(int \_messageID, PID \_pidSender)
  + RespondSetInactiveVirtualObject(int \_messageID, PID \_pidSender)
  + RespondKillVirtualObject(int \_messageID, PID \_pidSender)
  + RespondRequestCheckVirtualObject(int \_messageID, PID \_pidSender)
  + *RespondClickVirtualObject(int* \_messageID*,* PID \_pidSender*) // after step one*
  + **All fails:**
  + FailToStartVirtualObject(int \_messageID, PID \_pidSender)
  + FailToStopVirtualObject(int \_messageID, PID \_pidSender)
  + FailToSetActiveVirtualObject(int \_messageID, PID \_pidSender)
  + FailToSetInactiveVirtualObject(int \_messageID, PID \_pidSender)
  + FailToKillVirtualObject(int \_messageID, PID \_pidSender)
  + FailToRequestCheckVirtualObject(int \_messageID, PID \_pidSender)
  + *FailToClickVirtualObject(int* \_messageID*,* PID \_pidSender*) // after step one*
* CodeObjectCameraFeedProtocol
  + TBD (= to be defined)
* ControlPositionProtocol *// after step one*
* ControlInputProtocol *// after step one*
* PositionBluetoothScannerProtocol *// after step one*
* PositionBluetoothDeviceProtocol *// after step one*
* BluetoothScannerInterfaceProtocol *// after step one*
* BluetoothDeviceInterfaceProtocol *// after step one*
* InputGestureProtocol *// after step one*
* InputSpeechProtocol *// after step one*
* GestureCameraFeedProtocol *// after step one*
* SpeechRecognitionMicrophoneProtocol *// after step one*