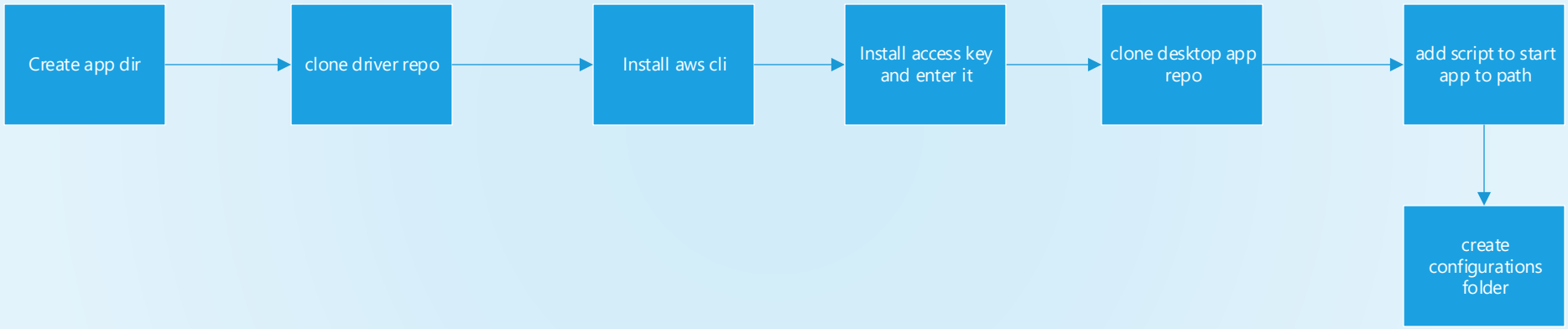
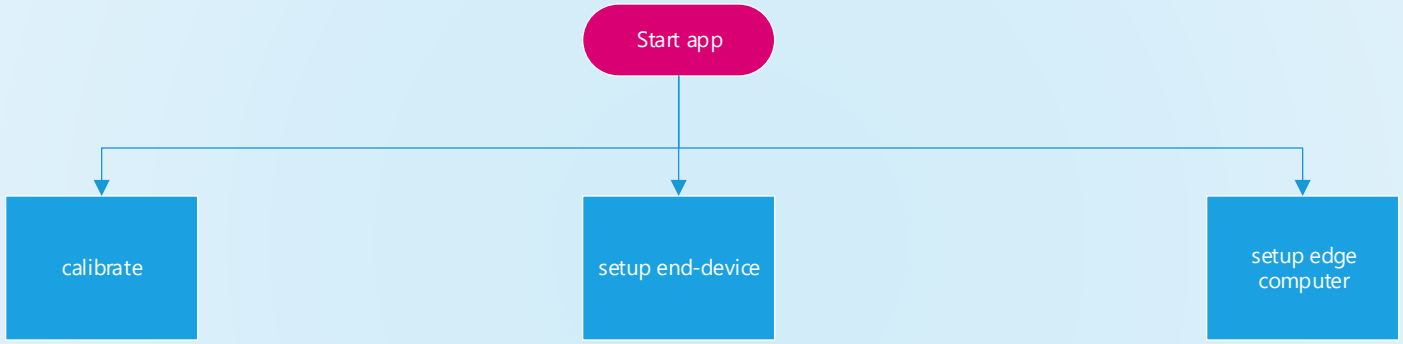
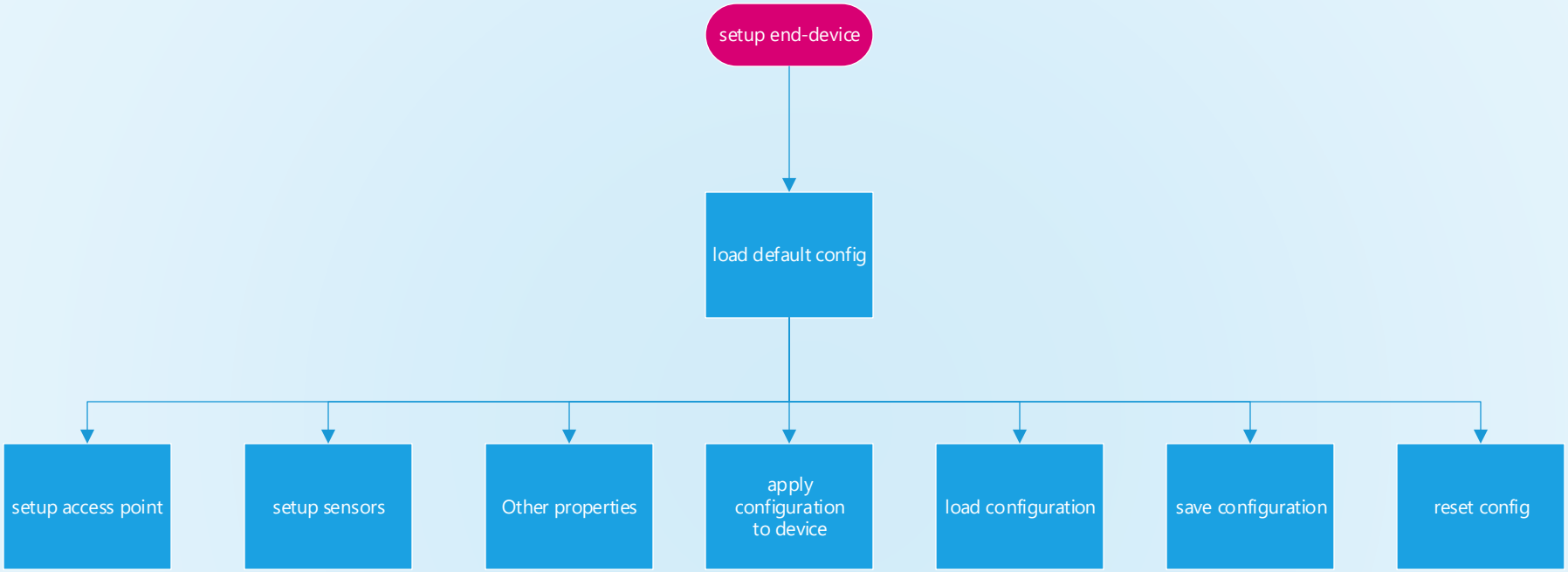
BCIT Research EMA

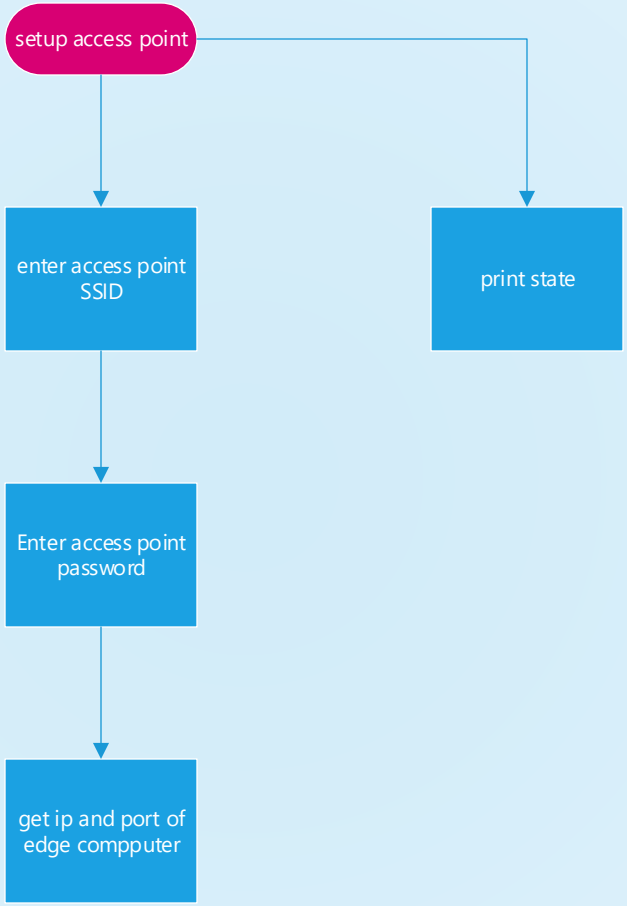
Desktop Application Design

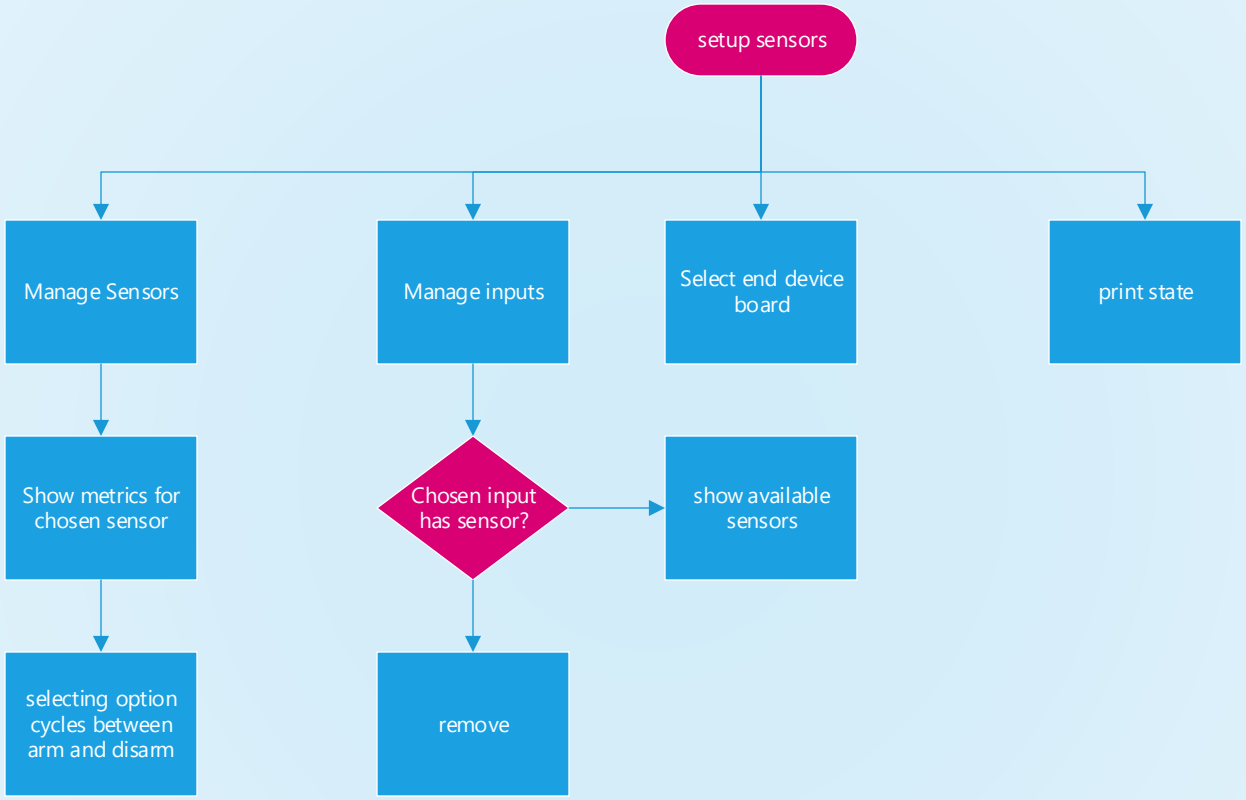
Created: May 22, 2021 – Karel Chanivecky
Updates:

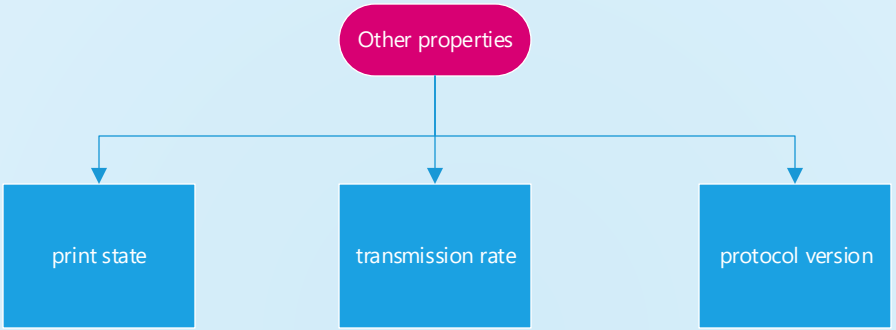

Create app dir
clone driver repo
Install aws cli
Install access key and enter it
clone desktop app repo
add script to start app to path
create configurations folder

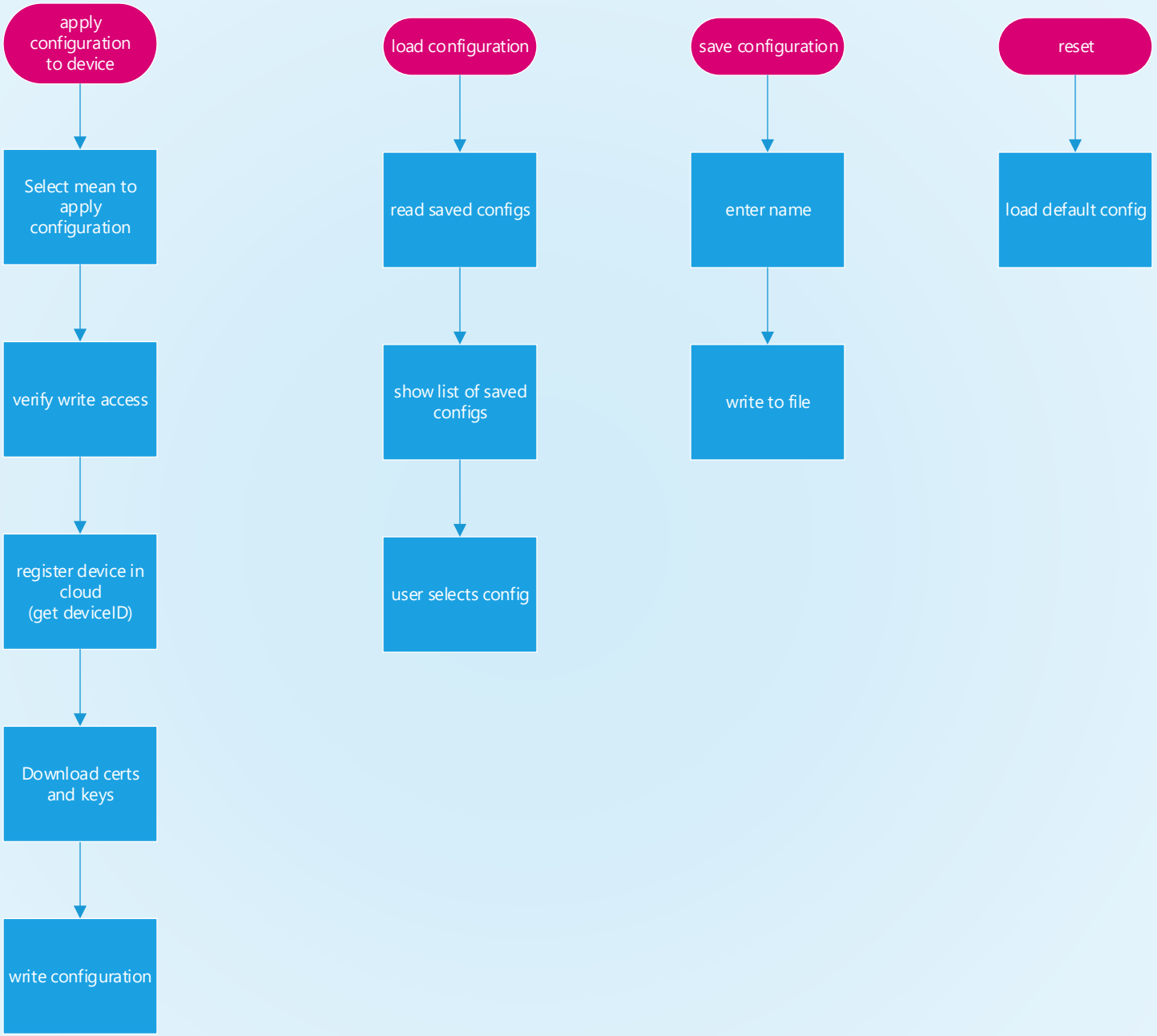

Start app
calibrate
setup end-device
setup edge computer
TBD
TBD


setup end-device
load default config
setup access point
setup sensors
Other properties
apply configuration to device
load configuration
save configuration
reset config

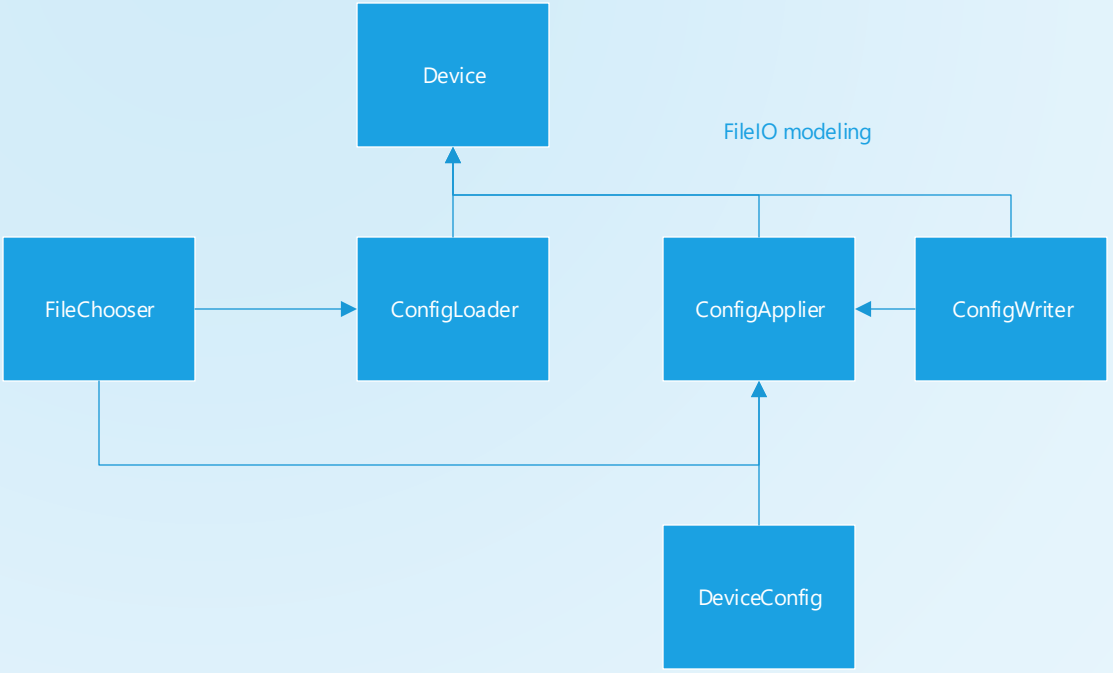

setup access point
enter access point SSID
print state
Enter access point password
get ip and port of edge compputer


setup sensors
Manage Sensors
Manage inputs
Select end device board
print state
Show metrics for chosen sensor
Chosen input has sensor?
show available sensors
selecting option cycles between arm and disarm
remove


Other properties
print state
transmission rate
protocol version


apply configuration to device
Select mean to apply configuration
verify write access
register device in cloud (get deviceID)
Download certs and keys
write configuration
load configuration
read saved configs
show list of saved configs
user selects config
save configuration
enter name
write to file
reset
load default config

Ignore for now



MenuUI: Provides a graphical representation of menu. For example, could be CLI (print to screen) or could be GUI (JavaFX)

MenuSerializer: Presents several MenuProviders as one(Current implementation returns Option[] and not a new MenuProvider)

MenuProvider: An interface that defines methods to get options

BaseMenuProvider: An abstract class that defines basic variables that all MenuProviders should have. Can be directly inherited to create new MenuProviders

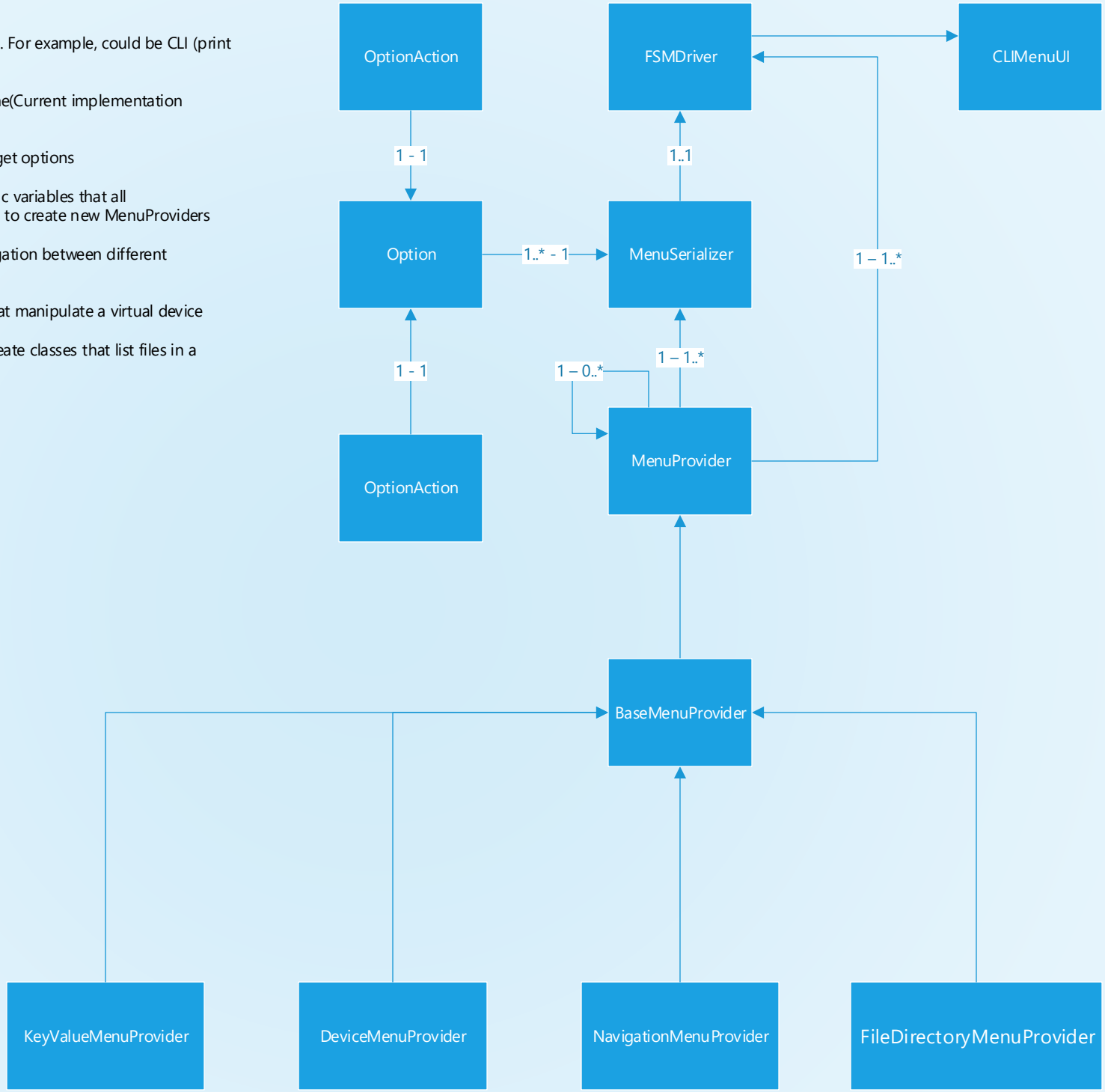
NavigationMenu: A concrete class that provides navigation between different MenuProviders as part of the output options.

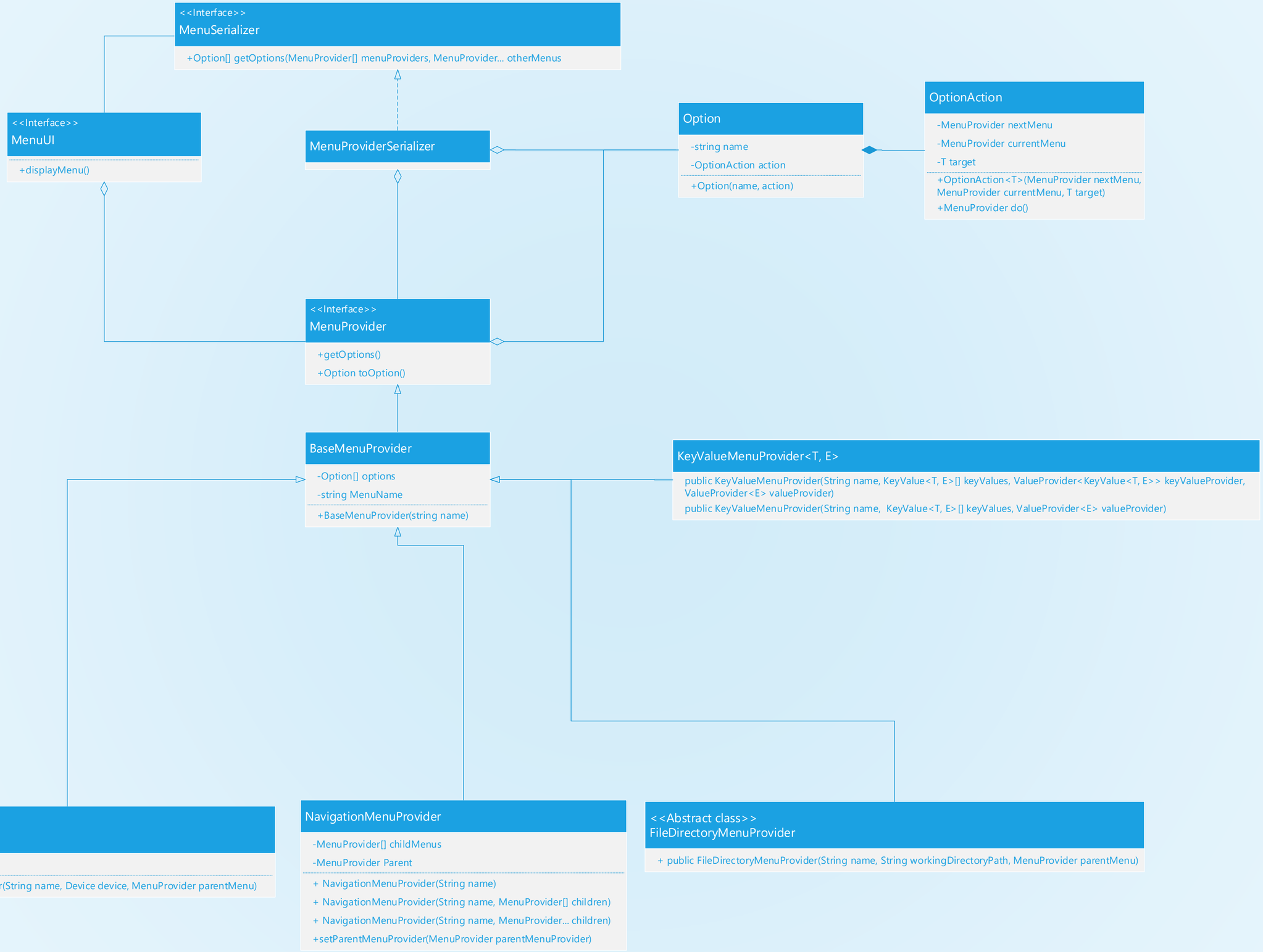
DeviceMenu: Abstract class. Used to create classes that manipulate a virtual device

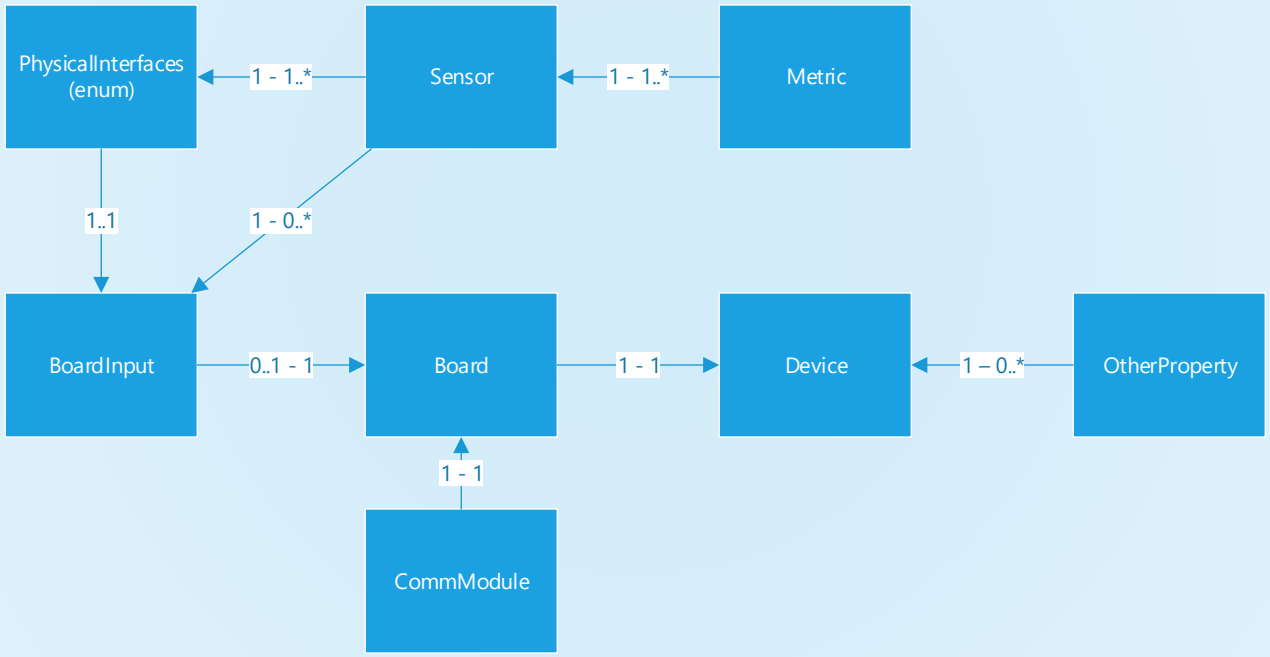
FileDirectoryMenuProvider: Abstract dass. Used to create classes that list files in a directory

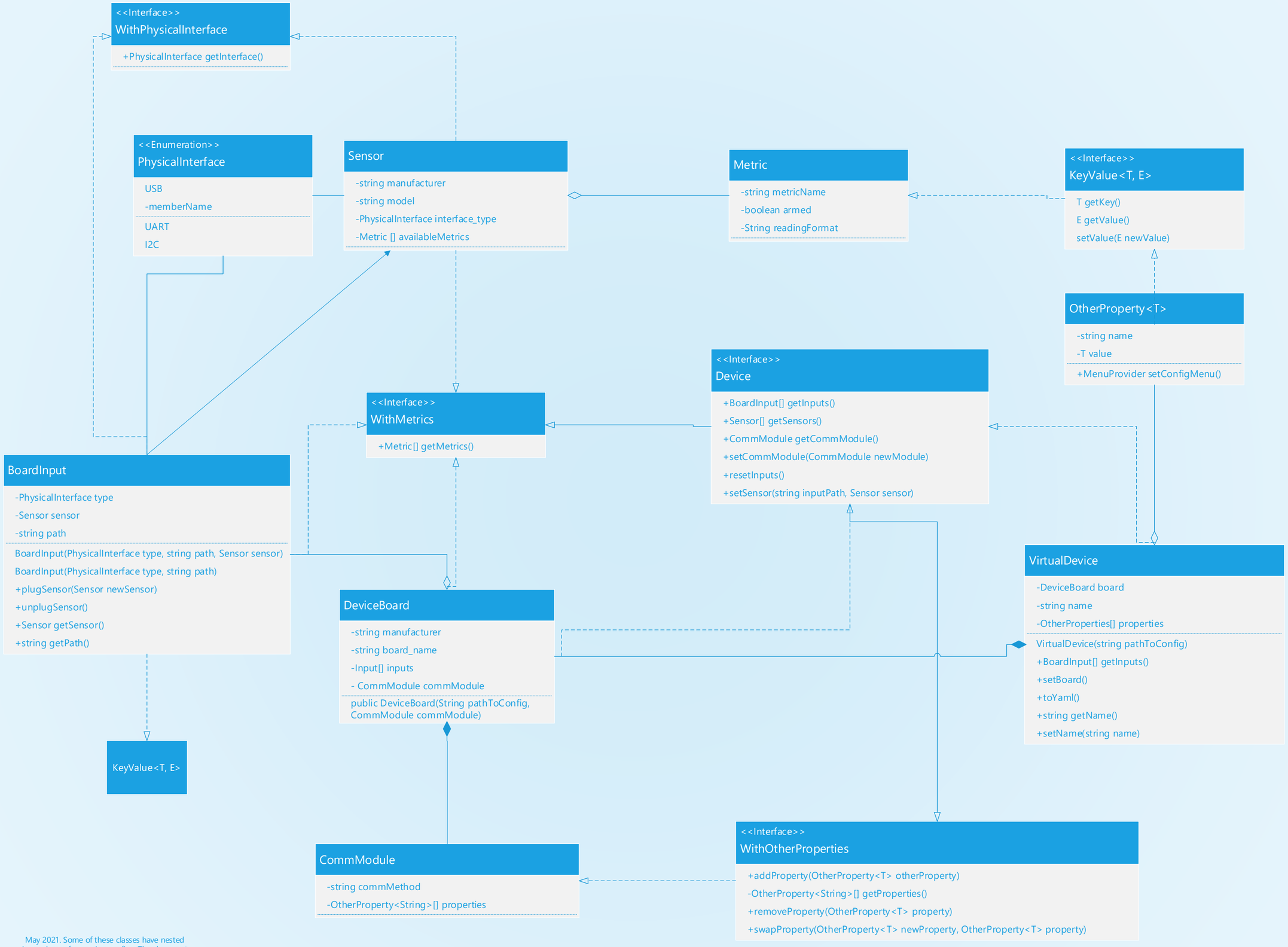
Option: Models a menu option

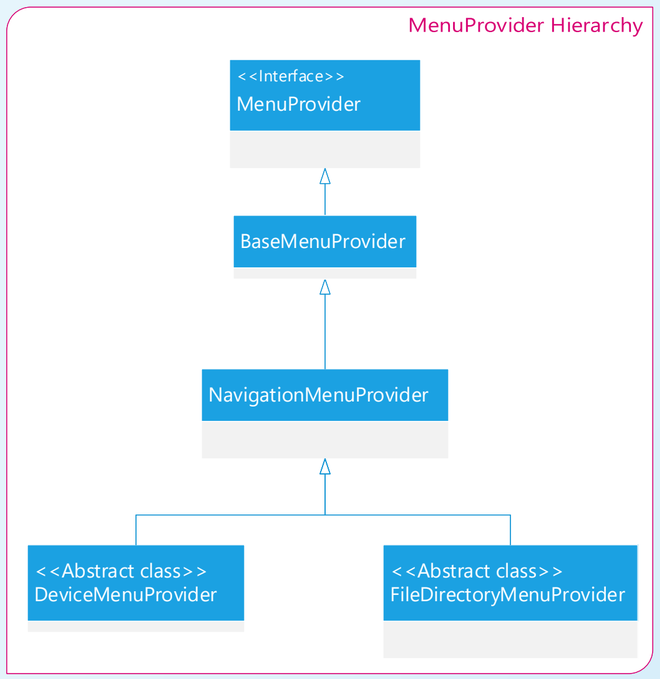
OptionAction: the action represented by the option




<<Interface>>
MenuSerializer
+Option[] getOptions(MenuProvider[] menuProviders, MenuProvider... otherMenus
<<Interface>>
MenuUI
+displayMenu()
MenuProviderSerializer
Option
-string name
-OptionAction action
+Option(name, action)
OptionAction
-MenuProvider nextMenu
-MenuProvider currentMenu
-T target
+OptionAction<T>(MenuProvider nextMenu, MenuProvider currentMenu, T target)
+MenuProvider do()
<<Interface>>
MenuProvider
+getOptions()
+Option toOption()
BaseMenuProvider
-Option[] options
-string MenuName
+BaseMenuProvider(string name)
KeyValueMenuProvider<T, E>
public KeyValueMenuProvider(String name, KeyValue<T, E>[] keyValues, ValueProvider<KeyValue<T, E>> keyValueProvider, ValueProvider<E> valueProvider)
public KeyValueMenuProvider(String name, KeyValue<T, E>[] keyValues, ValueProvider<E> valueProvider)
<<Abstract class>>
DeviceMenuProvider
-Device device
+public DeviceMenuProvider(String name, Device device, MenuProvider parentMenu)
NavigationMenuProvider
-MenuProvider[] childMenus
-MenuProvider Parent
+ NavigationMenuProvider(String name)
+ NavigationMenuProvider(String name, MenuProvider[] children)
+ NavigationMenuProvider(String name, MenuProvider... children)
+setParentMenuProvider(MenuProvider parentMenuProvider)
<<Abstract class>>
FileDirectoryMenuProvider
+ public FileDirectoryMenuProvider(String name, String workingDirectoryPath, MenuProvider parentMenu)


PhysicalInterfaces (enum)
1 - 1..*
Sensor
1 - 1..*
Metric
1..1
1 - 0..*
BoardInput
0..1 - 1
Board
1 - 1
Device
1 – 0..*
OtherProperty
1 - 1
CommModule


<<Interface>>
WithPhysicalInterface
+PhysicalInterface getInterface()
<<Enumeration>>
PhysicalInterface
USB
-memberName
UART
I2C
Sensor
-string manufacturer
-string model
-PhysicalInterface interface_type
-Metric [] availableMetrics
Metric
-string metricName
-boolean armed
-String readingFormat
<<Interface>>
KeyValue<T, E>
T getKey()
E getValue()
setValue(E newValue)
OtherProperty<T>
-string name
-T value
+MenuProvider setConfigMenu()
<<Interface>>
WithMetrics
+Metric[] getMetrics()
<<Interface>>
Device
+BoardInput[] getInputs()
+Sensor[] getSensors()
+CommModule getCommModule()
+setCommModule(CommModule newModule)
+resetInputs()
+setSensor(string inputPath, Sensor sensor)
BoardInput
-PhysicalInterface type
-Sensor sensor
-string path
BoardInput(PhysicalInterface type, string path, Sensor sensor)
BoardInput(PhysicalInterface type, string path)
+plugSensor(Sensor newSensor)
+unplugSensor()
+Sensor getSensor()
+string getPath()
DeviceBoard
-string manufacturer
-string board_name
-Input[] inputs
- CommModule commModule
public DeviceBoard(String pathToConfig, CommModule commModule)
VirtualDevice
-DeviceBoard board
-string name
-OtherProperties[] properties
VirtualDevice(string pathToConfig)
+BoardInput[] getInputs()
+setBoard()
+toYaml()
+string getName()
+setName(string name)
KeyValue<T, E>
CommModule
-string commMethod
-OtherProperty<String>[] properties
<<Interface>>
WithOtherProperties
+addProperty(OtherProperty<T> otherProperty)
-OtherProperty<String>[] getProperties()
+removeProperty(OtherProperty<T> property)
+swapProperty(OtherProperty<T> newProperty, OtherProperty<T> property)
May 2021. Some of these classes have nested inner classes for parsing configs. They have not been implemented. Look for the TODOs in code



These may not necessarily have been completed

DeviceMenuProvider

SensorChooser

InputChooser

MetricChooser

Given a device, ask user to choose an input. Perform an action.

MetricChooser

SensorChooser

ConfigApplierChooser

InputChooser

BoardChooser

UI modeling

BoardChooser