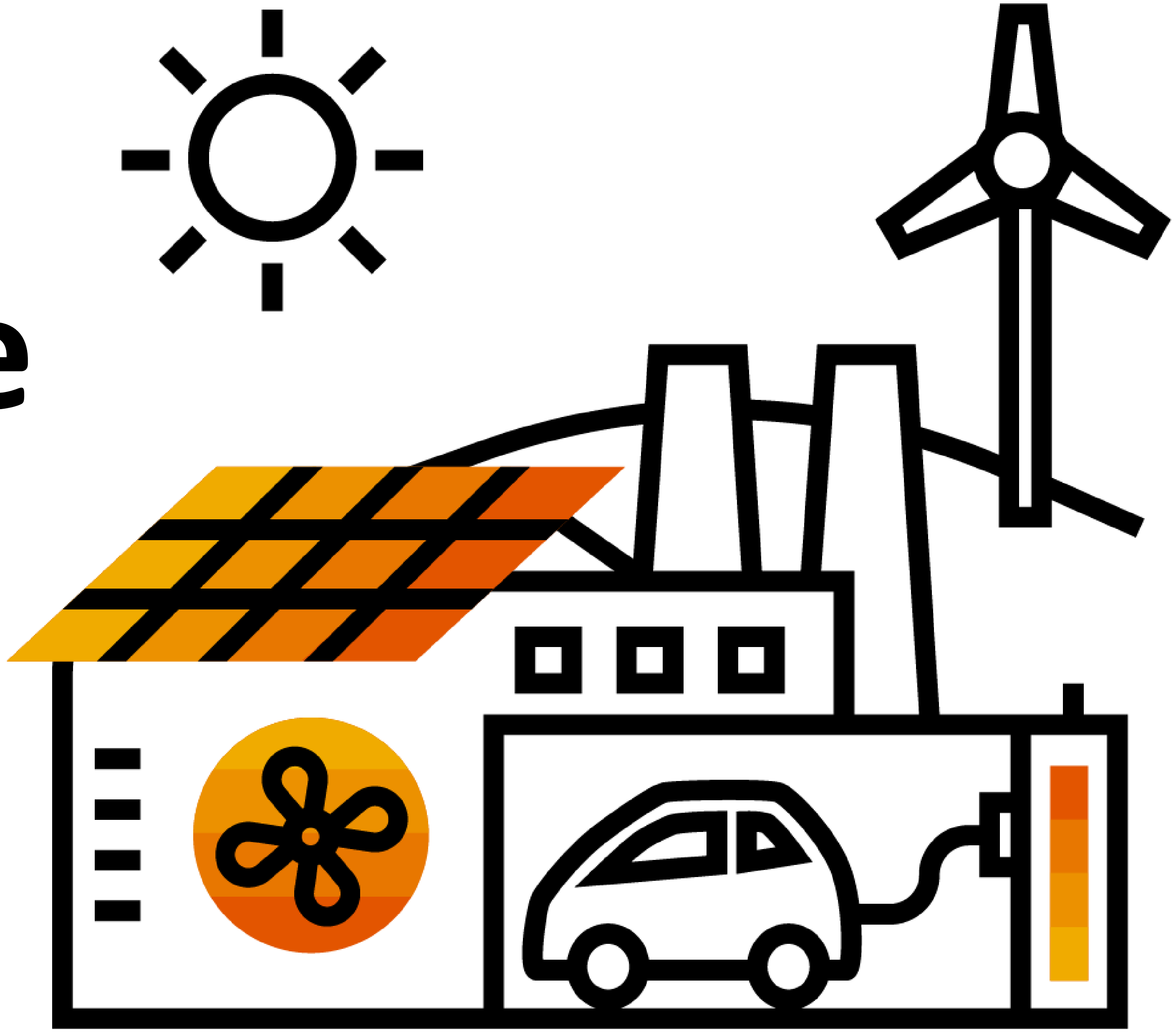
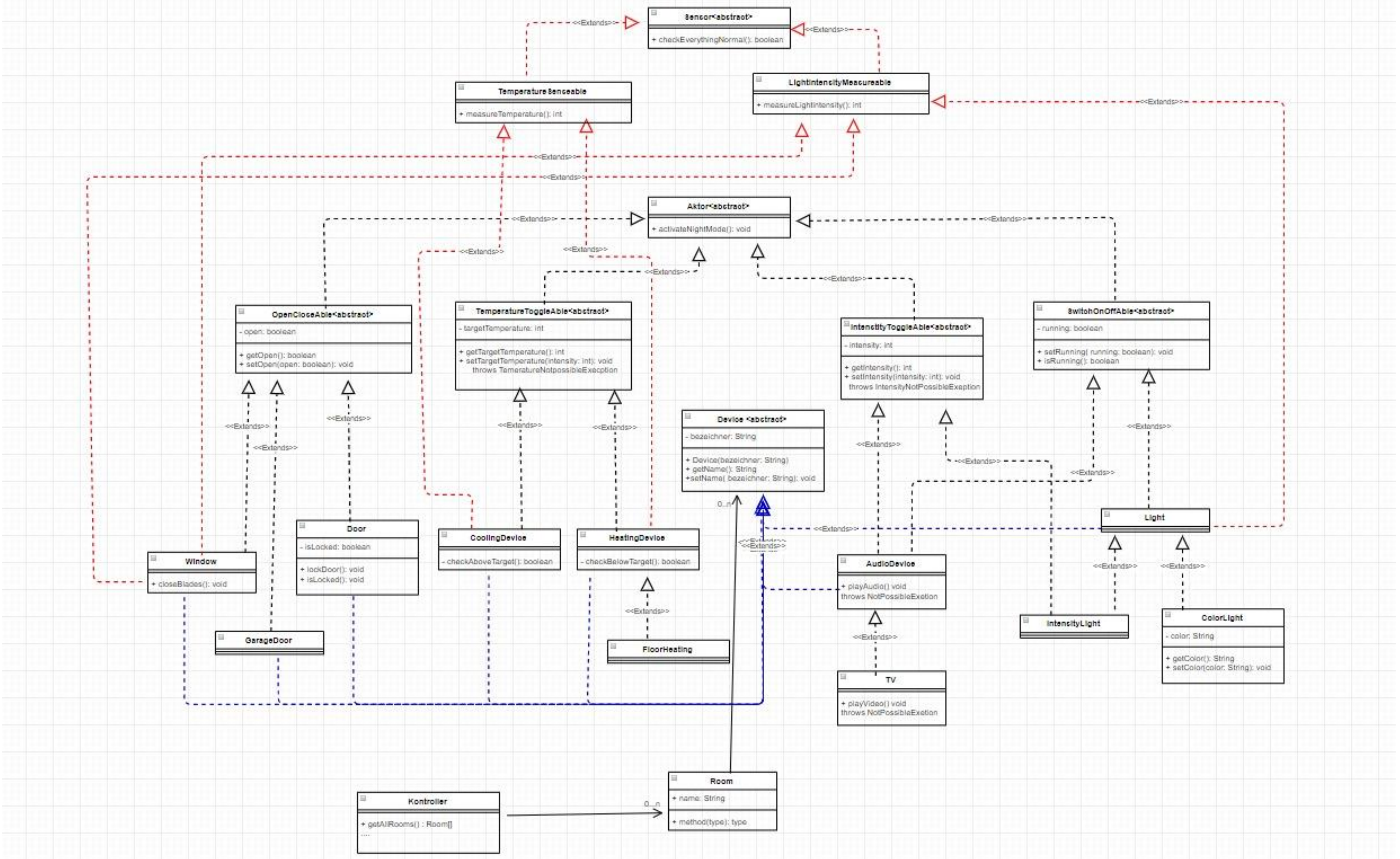
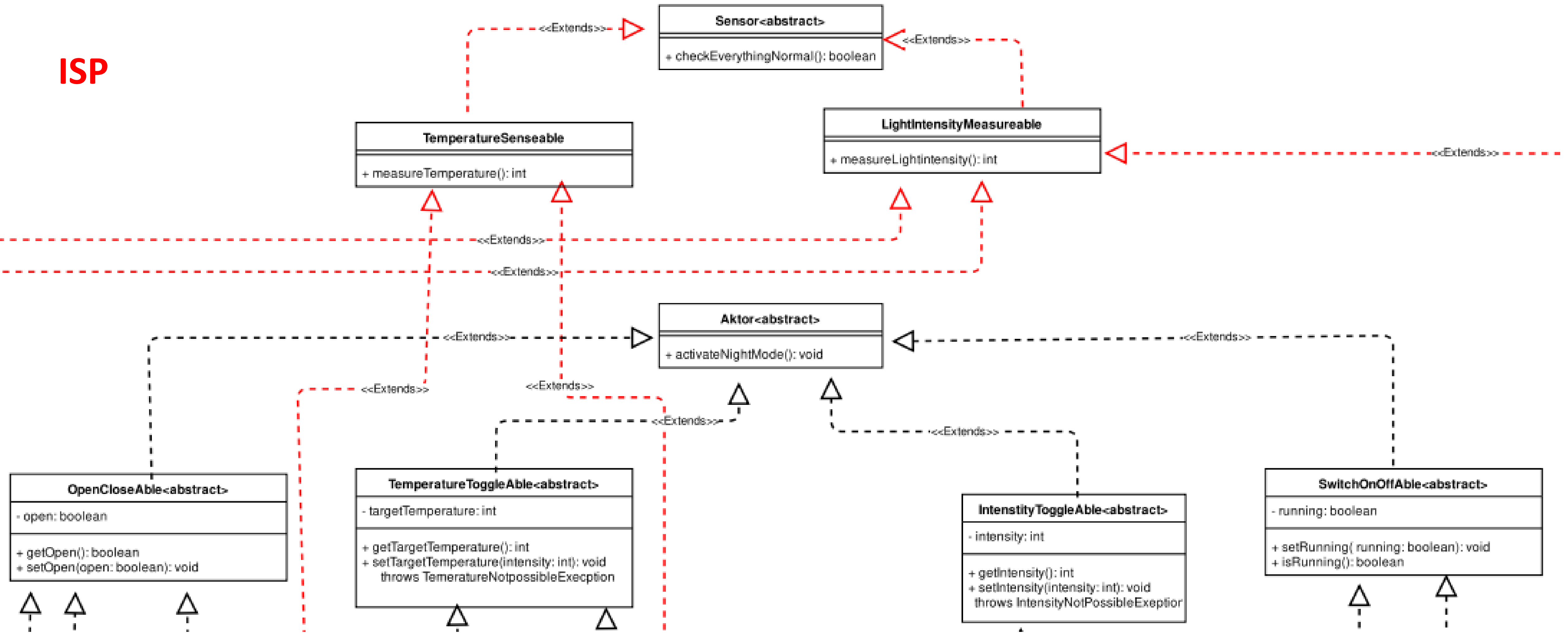
Smarthome

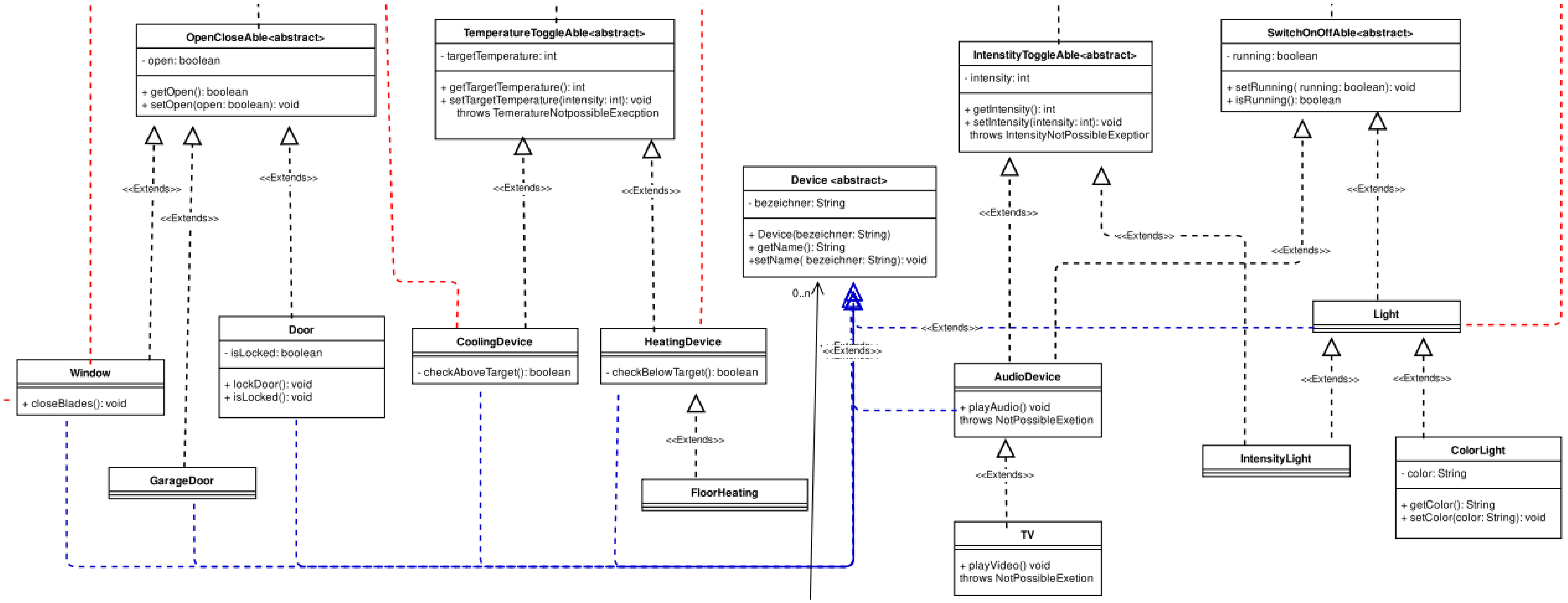
Gruppe3:
Paul Bahde, Maximilian Abrams, Jonas Braun, Nils Abels, Marcel Eitel, Fabian

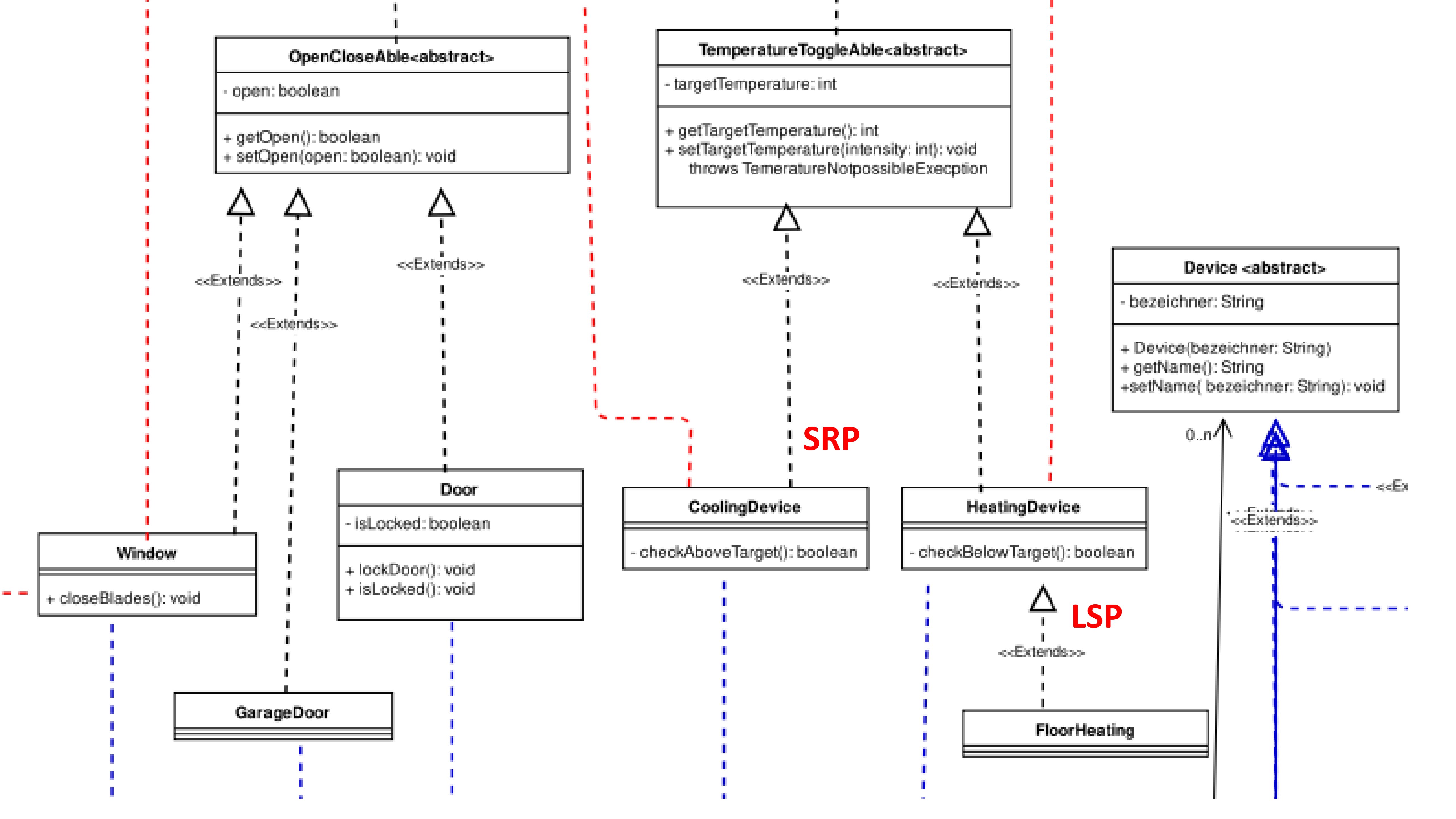


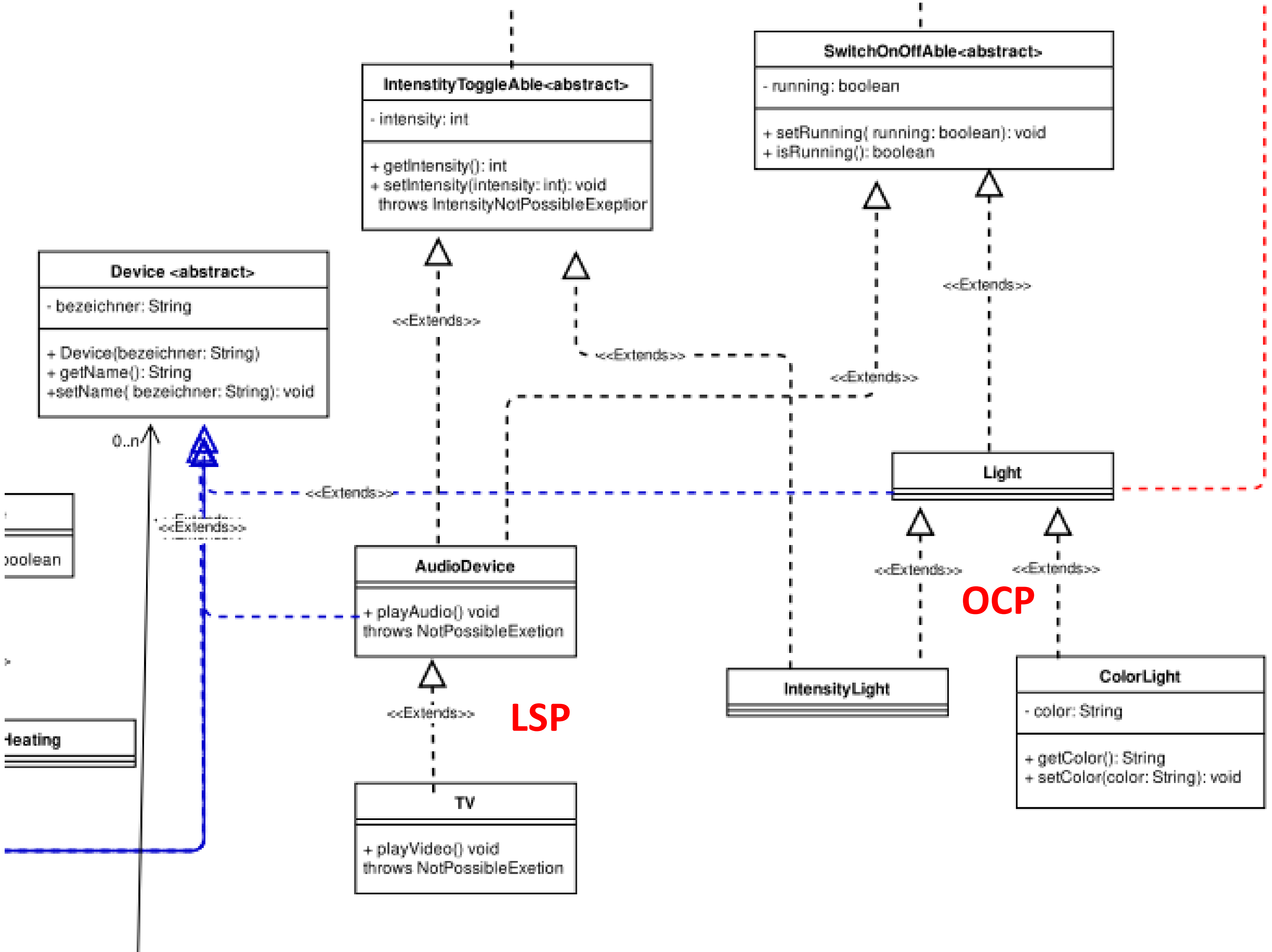

Sensor<abstract>
+ checkEverythingNormal(): boolean
TemperatureSenseable
+ measureTemperature(): int
LightIntensityMeasureable
+ measureLightIntensity(): int
Aktor<abstract>
+ activateNightMode(): void
OpenCloseAble<abstract>
- open: boolean
+ getOpen(): boolean
+ setOpen(open: boolean): void
TemperatureToggleAble<abstract>
- targetTemperature: int
+ getTargetTemperature(): int
+ setTargetTemperature(intensity: int): void
throws TemeratureNotpossibleExecption
IntensityToggleAble<abstract>
- intensity: int
+ getIntensity(): int
+ setIntensity(intensity: int): void
throws IntensityNotPossibleExeption
SwitchOnOffAble<abstract>
- running: boolean
+ setRunning(running: boolean): void
+ isRunning(): boolean
Device <abstract>
- bezeichner: String
+ Device(bezeichner: String)
+ getName(): String
+setName(bezeichner: String): void
Window
+ closeBlades(): void
Door
- isLocked: boolean
+ lockDoor(): void
+ isLocked(): void
GarageDoor
CoolingDevice
- checkAboveTarget(): boolean
HeatingDevice
- checkBelowTarget(): boolean
FloorHeating
AudioDevice
+ playAudio() void
throws NotPossibleExetion
TV
+ playVideo() void
throws NotPossibleExetion
Light
IntensityLight
ColorLight
- color: String
+ getColor(): String
+ setColor(color: String): void
Room
+ name: String
+ method(type): type
Kontroller
+ getAllRooms() : Room[]
....
<<Extends>>
0..n
0..n

ISP


Sensor<abstract>
+ checkEverythingNormal(): boolean
TemperatureSenseable
+ measureTemperature(): int
LightIntensityMeasureable
+ measureLightIntensity(): int
Aktor<abstract>
+ activateNightMode(): void
OpenCloseAble<abstract>
- open: boolean
+ getOpen(): boolean
+ setOpen(open: boolean): void
TemperatureToggleAble<abstract>
- targetTemperature: int
+ getTargetTemperature(): int
+ setTargetTemperature(intensity: int): void
throws TemeratureNotpossibleExecption
IntenstityToggleAble<abstract>
- intensity: int
+ getIntensity(): int
+ setIntensity(intensity: int): void
throws IntensityNotPossibleExeptior
SwitchOnOffAble<abstract>
- running: boolean
+ setRunning(running: boolean): void
+ isRunning(): boolean
<<Extends>>


OpenCloseAble<abstract>
- open: boolean
+ getOpen(): boolean
+ setOpen(open: boolean): void
TemperatureToggleAble<abstract>
- targetTemperature: int
+ getTargetTemperature(): int
+ setTargetTemperature(intensity: int): void
throws TemeratureNotpossibleExecption
IntenstityToggleAble<abstract>
- intensity: int
+ getIntensity(): int
+ setIntensity(intensity: int): void
throws IntensityNotPossibleExeption
SwitchOnOffAble<abstract>
- running: boolean
+ setRunning(running: boolean): void
+ isRunning(): boolean
Device <abstract>
- bezeichner: String
+ Device(bezeichner: String)
+ getName(): String
+setName(bezeichner: String): void
0..n
Window
+ closeBlades(): void
GarageDoor
Door
- isLocked: boolean
+ lockDoor(): void
+ isLocked(): void
CoolingDevice
- checkAboveTarget(): boolean
HeatingDevice
- checkBelowTarget(): boolean
FloorHeating
AudioDevice
+ playAudio() void
throws NotPossibleExetion
TV
+ playVideo() void
throws NotPossibleExetion
Light
IntensityLight
ColorLight
- color: String
+ getColor(): String
+ setColor(color: String): void
<<Extends>>

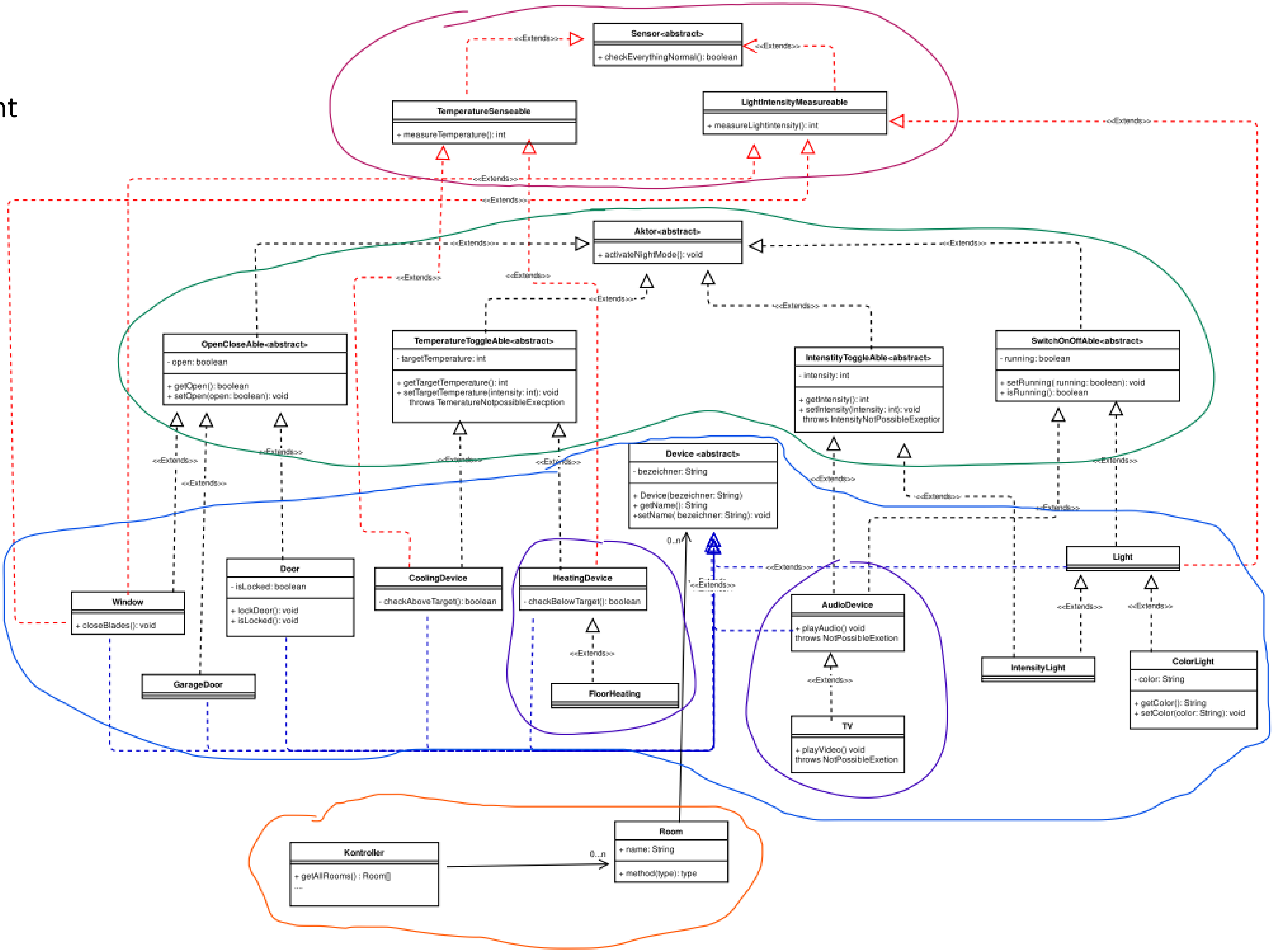

OpenCloseAble<abstract>
- open: boolean
+ getOpen(): boolean
+ setOpen(open: boolean): void
TemperatureToggleAble<abstract>
- targetTemperature: int
+ getTargetTemperature(): int
+ setTargetTemperature(intensity: int): void
throws TemeratureNotpossibleExecption
Device <abstract>
- bezeichner: String
+ Device(bezeichner: String)
+ getName(): String
+setName(bezeichner: String): void
<<Extends>>
<<Extends>>
<<Extends>>
<<Extends>>
<<Extends>>
<<Extends>>
<<Extends>>
0..n
SRP
LSP
Window
+ closeBlades(): void
Door
- isLocked: boolean
+ lockDoor(): void
+ isLocked(): void
CoolingDevice
- checkAboveTarget(): boolean
HeatingDevice
- checkBelowTarget(): boolean
GarageDoor
FloorHeating


IntenstityToggleAble<abstract>
- intensity: int
+ getIntensity(): int
+ setIntensity(intensity: int): void throws IntensityNotPossibleExeptior
SwitchOnOffAble<abstract>
- running: boolean
+ setRunning(running: boolean): void
+ isRunning(): boolean
Device <abstract>
- bezeichner: String
+ Device(bezeichner: String)
+ getName(): String
+setName(bezeichner: String): void
0..n
<<Extends>>
oolean
eating
AudioDevice
+ playAudio() void throws NotPossibleExetion
TV
+ playVideo() void throws NotPossibleExetion
Light
IntensityLight
ColorLight
- color: String
+ getColor(): String
+ setColor(color: String): void
LSP
OCP

ADP erfüllt
(wichtig dass Device nicht in orangenem Paket)

SDP tendentiell erfüllt (Stabilität wächst)

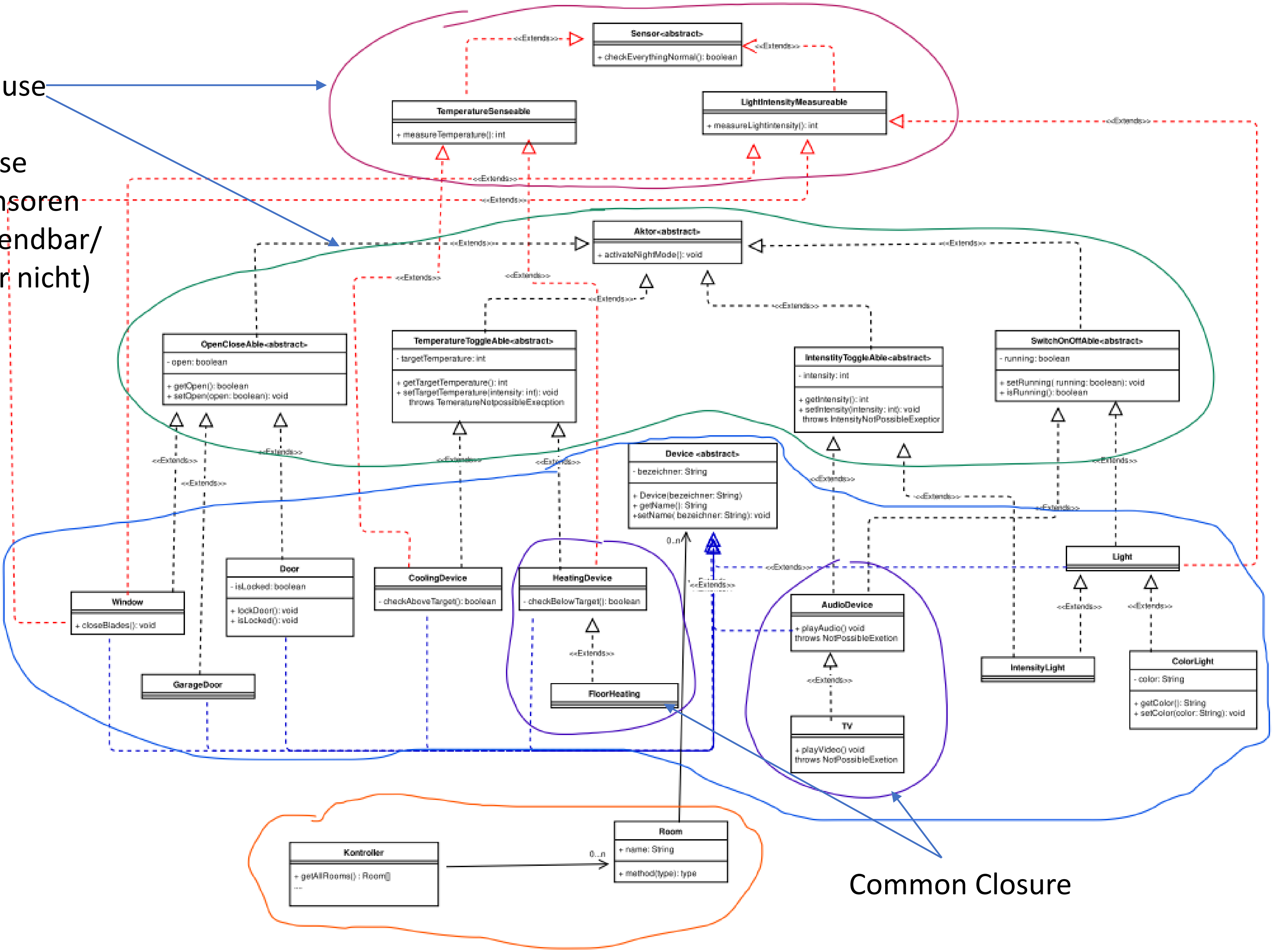
SAP
Stabiltät wächst in selbe Richtung, wie Abstraktheit



Sensor<abstract>
+ checkEverythingNormal(): boolean
TemperatureSenseable
+ measureTemperature(): int
LightIntensityMeasureable
+ measureLightintensity(): int
Aktor<abstract>
+ activateNightMode(): void
OpenCloseAble<abstract>
- open: boolean
+ getOpen(): boolean
+ setOpen(open: boolean): void
TemperatureToggleAble<abstract>
- targetTemperature: int
+ getTargetTemperature(): int
+ setTargetTemperature(intensity: int): void throws TemeratureNotpossibleExecption
IntensityToggleAble<abstract>
- intensity: int
+ getIntensity(): int
+ setIntensity(intensity: int): void throws IntensityNotPossibleExeptior
SwitchOnOffAble<abstract>
- running: boolean
+ setRunning(running: boolean): void
+ isRunning(): boolean
Device <abstract>
- bezeichner: String
+ Device(bezeichner: String)
+ getName(): String
+setName(bezeichner: String): void
0..n
Window
+ closeBlades(): void
Door
- isLocked: boolean
+ lockDoor(): void
+ isLocked(): void
GarageDoor
CoolingDevice
- checkAboveTarget(): boolean
HeatingDevice
- checkBelowTarget(): boolean
FloorHeating
AudioDevice
+ playAudio() void throws NotPossibleExetion
TV
+ playVideo() void throws NotPossibleExetion
Light
IntensityLight
ColorLight
- color: String
+ getColor(): String
+ setColor(color: String): void
Kontroller
+ getAllRooms() : Room[]
...
Room
+ name: String
+ method(type): type
0...n
<<Extends>>

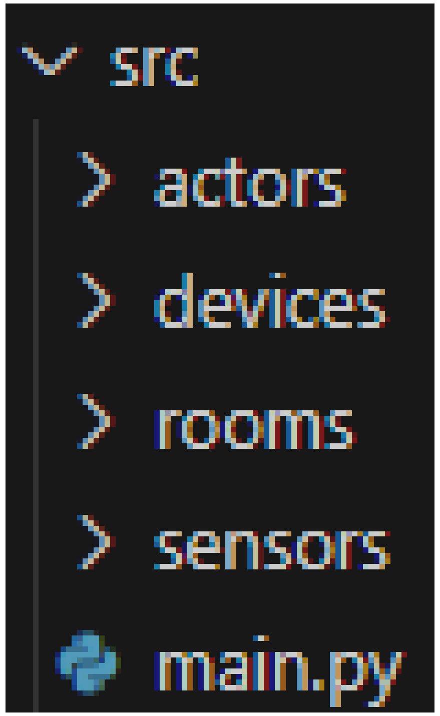
Common Reuse

Reuse Release
(Aktor & Sensoren
wiederverwendbar/
Devices eher nicht)

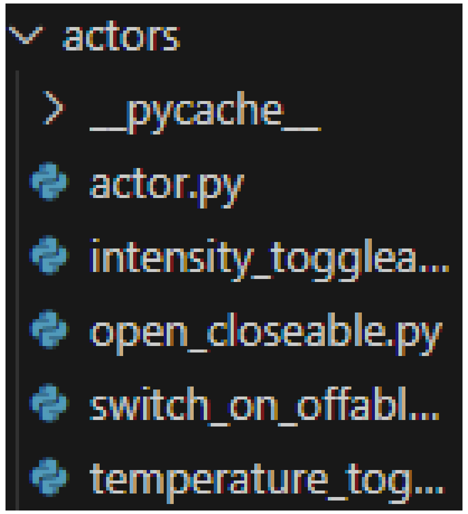

Sensor<abstract>
+ checkEverythingNormal(): boolean
TemperatureSenseable
+ measureTemperature(): int
LightIntensityMeasureable
+ measureLightIntensity(): int
Aktor<abstract>
+ activateNightMode(): void
OpenCloseAble<abstract>
- open: boolean
+ getOpen(): boolean
+ setOpen(open: boolean): void
TemperatureToggleAble<abstract>
- targetTemperature: int
+ getTargetTemperature(): int
+ setTargetTemperature(intensity: int): void throws TemeratureNotpossibleExecption
IntensityToggleAble<abstract>
- intensity: int
+ getIntensity(): int
+ setIntensity(intensity: int): void throws IntensityNotPossibleExeption
SwitchOnOffAble<abstract>
- running: boolean
+ setRunning(running: boolean): void
+ isRunning(): boolean
Device <abstract>
- bezeichner: String
+ Device(bezeichner: String)
+ getName(): String
+setName(bezeichner: String): void
Window
+ closeBlades(): void
GarageDoor
Door
- isLocked: boolean
+ lockDoor(): void
+ isLocked(): void
CoolingDevice
- checkAboveTarget(): boolean
HeatingDevice
- checkBelowTarget(): boolean
FloorHeating
AudioDevice
+ playAudio() void throws NotPossibleExetion
TV
+ playVideo() void throws NotPossibleExetion
Light
IntensityLight
ColorLight
- color: String
+ getColor(): String
+ setColor(color: String): void
Kontroller
+ getAllRooms() : Room[]
...
Room
+ name: String
+ method(type): type
<<Extends>>
0..n

Common Closure

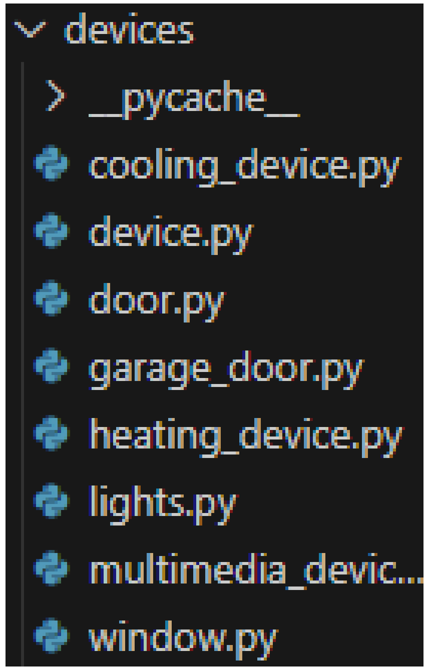
Package-Struktur im Code



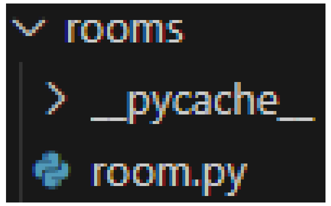
```
v src
  > actors
  > devices
  > rooms
  > sensors
  main.py
```



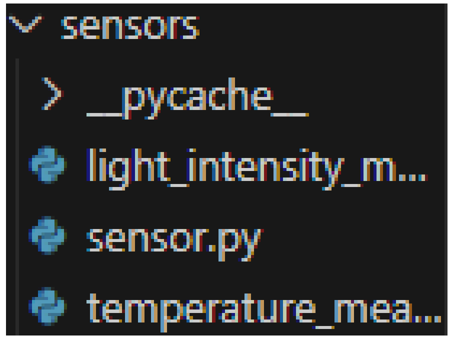
```
v actors
  > __pycache__
  actor.py
  intensity_togglea...
  open_closeable.py
  switch_on_offabl...
  temperature_tog...
```



```
v devices
  > __pycache__
  cooling_device.py
  device.py
  door.py
  garage_door.py
  heating_device.py
  lights.py
  multimedia_devic...
  window.py
```

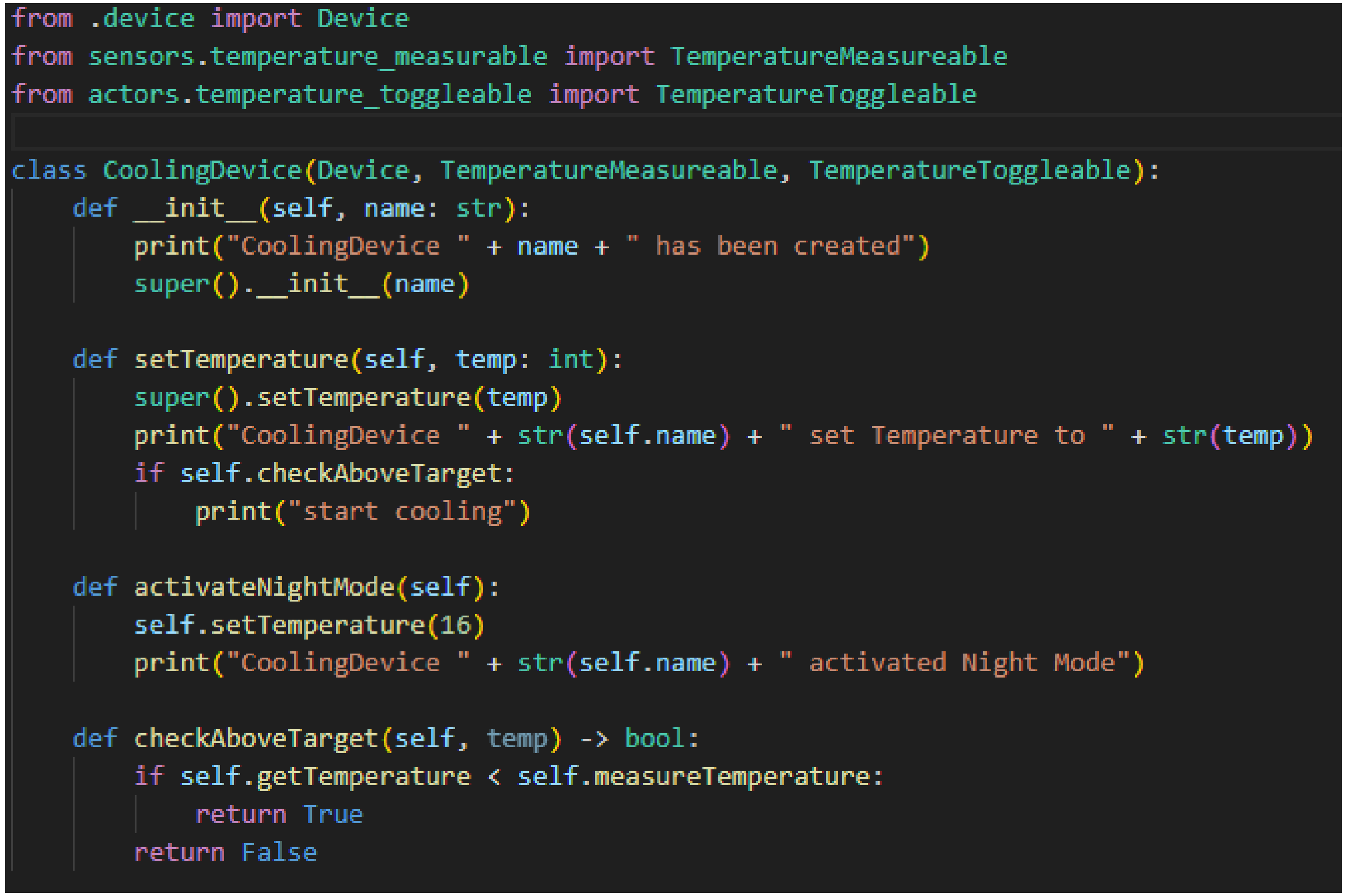


```
v rooms
  > __pycache__
  room.py
```



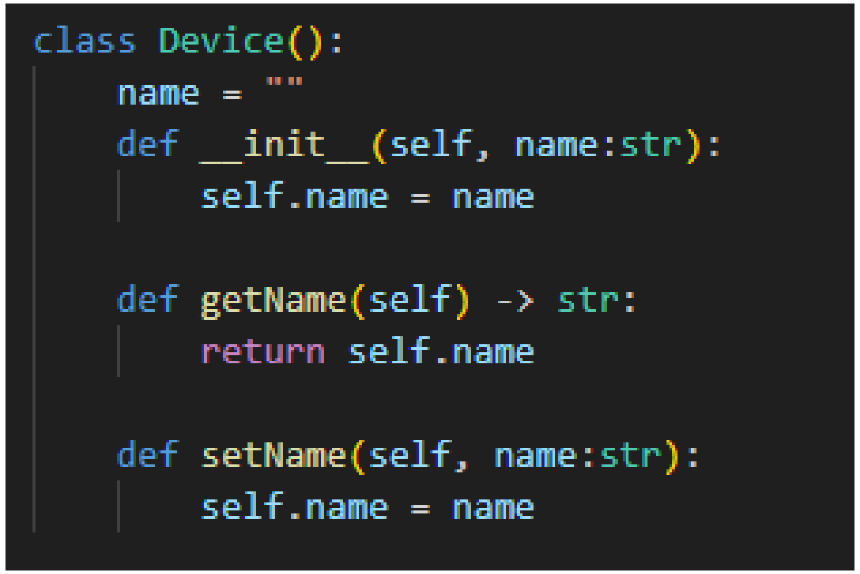
```
v sensors
  > __pycache__
  light_intensity_m...
  sensor.py
  temperature_mea...
```

SRP

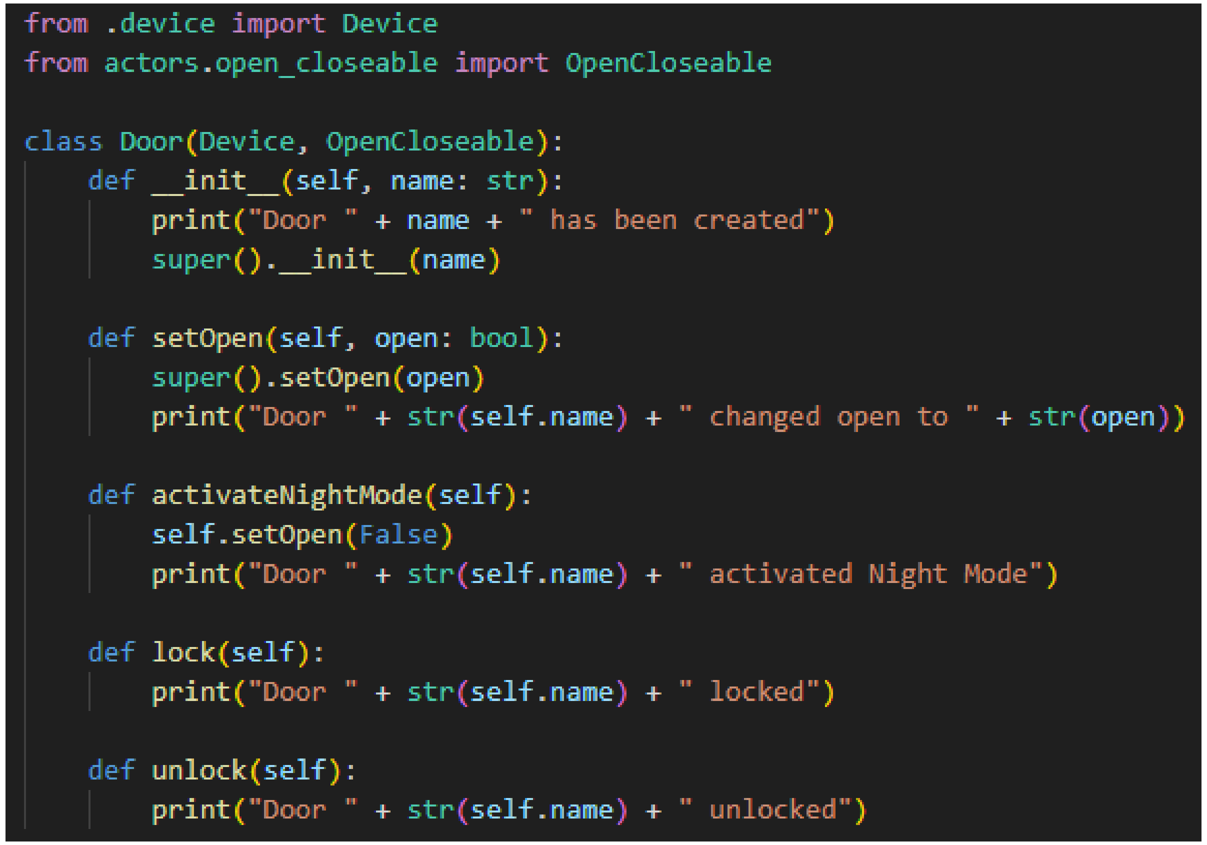


```python
from .device import Device
from sensors.temperature_measurable import TemperatureMeasureable
from actors.temperature_toggleable import TemperatureToggleable

class CoolingDevice(Device, TemperatureMeasureable, TemperatureToggleable):
    def __init__(self, name: str):
        print("CoolingDevice " + name + " has been created")
        super().__init__(name)

    def setTemperature(self, temp: int):
        super().setTemperature(temp)
        print("CoolingDevice " + str(self.name) + " set Temperature to " + str(temp))
        if self.checkAboveTarget:
            print("start cooling")

    def activateNightMode(self):
        self.setTemperature(16)
        print("CoolingDevice " + str(self.name) + " activated Night Mode")

    def checkAboveTarget(self, temp) -> bool:
        if self.getTemperature < self.measureTemperature:
            return True
        return False
```

OCP

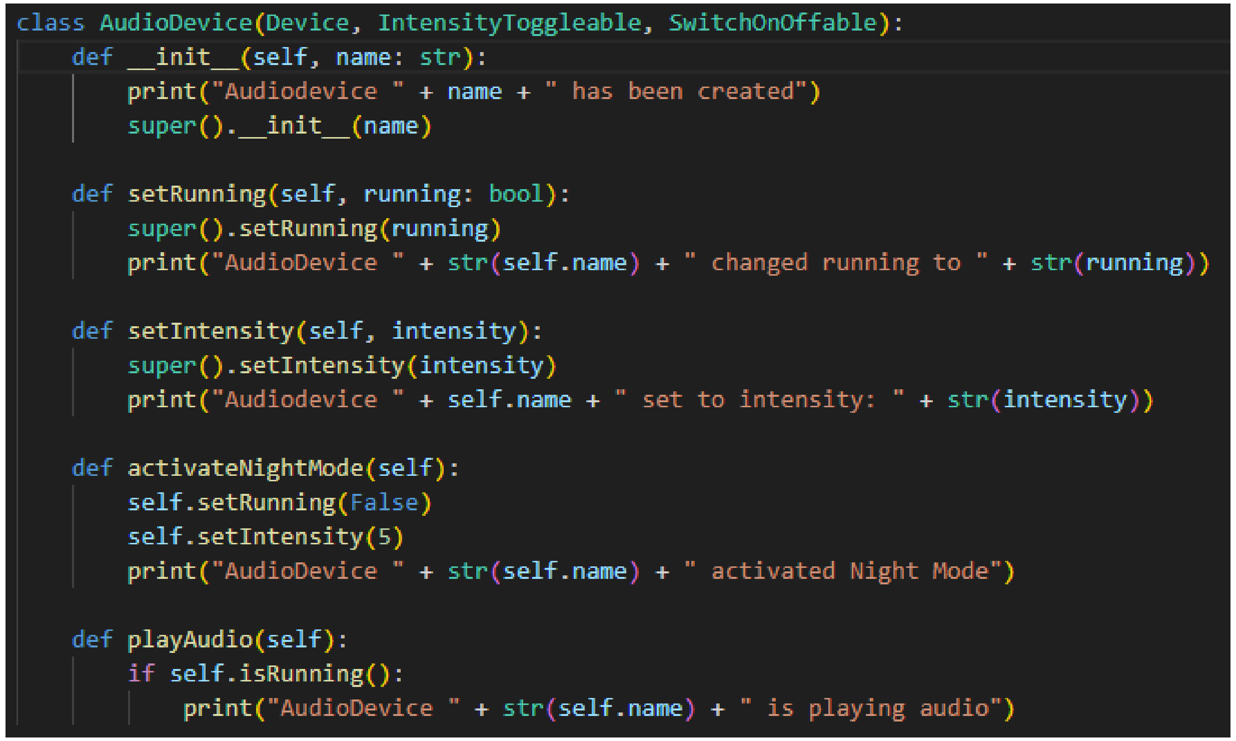


```python
class Device():
    name = ""
    def __init__(self, name:str):
        self.name = name

    def getName(self) -> str:
        return self.name

    def setName(self, name:str):
        self.name = name
```

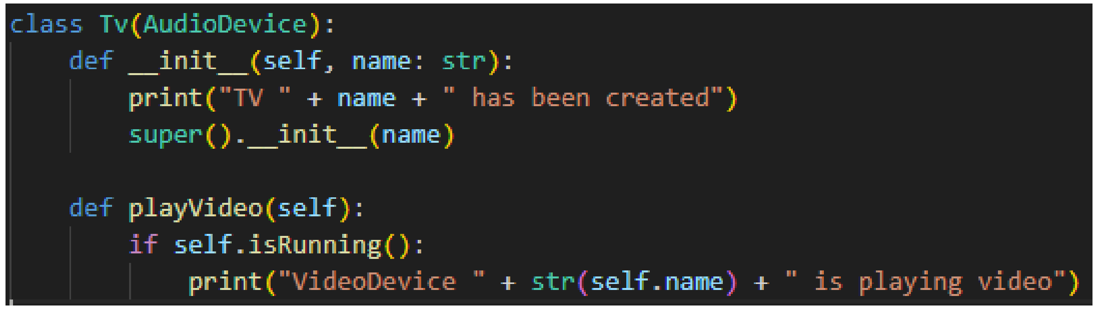


```python
from .device import Device
from actors.open_closeable import OpenCloseable

class Door(Device, OpenCloseable):
    def __init__(self, name: str):
        print("Door " + name + " has been created")
        super().__init__(name)

    def setOpen(self, open: bool):
        super().setOpen(open)
        print("Door " + str(self.name) + " changed open to " + str(open))

    def activateNightMode(self):
        self.setOpen(False)
        print("Door " + str(self.name) + " activated Night Mode")

    def lock(self):
        print("Door " + str(self.name) + " locked")

    def unlock(self):
        print("Door " + str(self.name) + " unlocked")
```

LSP

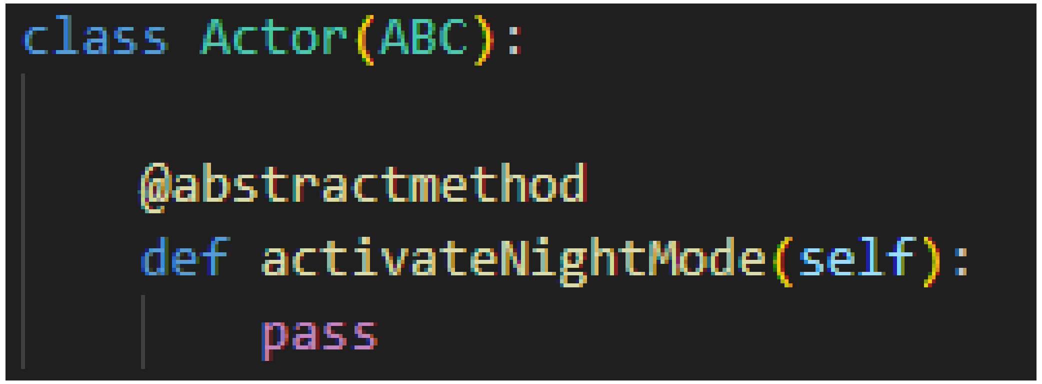


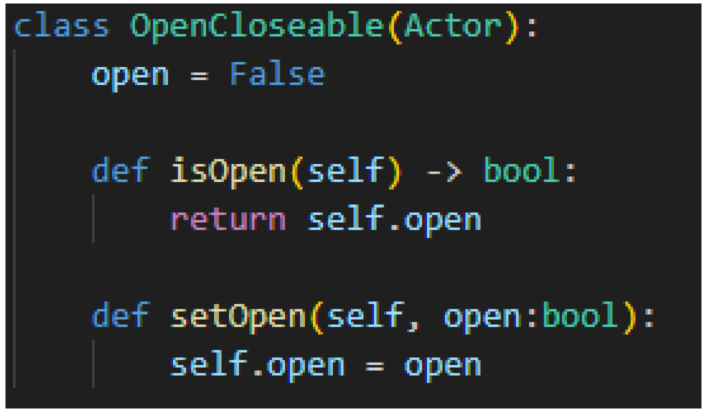
```python
class AudioDevice(Device, IntensityToggleable, SwitchOnOffable):
    def __init__(self, name: str):
        print("Audiodevice " + name + " has been created")
        super().__init__(name)

    def setRunning(self, running: bool):
        super().setRunning(running)
        print("AudioDevice " + str(self.name) + " changed running to " + str(running))

    def setIntensity(self, intensity):
        super().setIntensity(intensity)
        print("Audiodevice " + self.name + " set to intensity: " + str(intensity))

    def activateNightMode(self):
        self.setRunning(False)
        self.setIntensity(5)
        print("AudioDevice " + str(self.name) + " activated Night Mode")

    def playAudio(self):
        if self.isRunning():
            print("AudioDevice " + str(self.name) + " is playing audio")
```

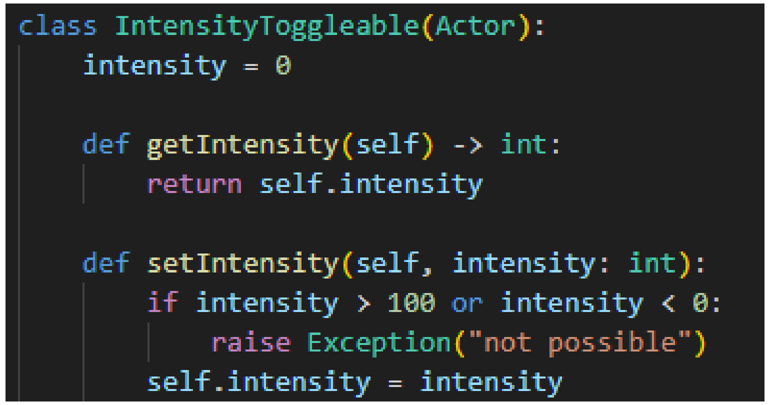


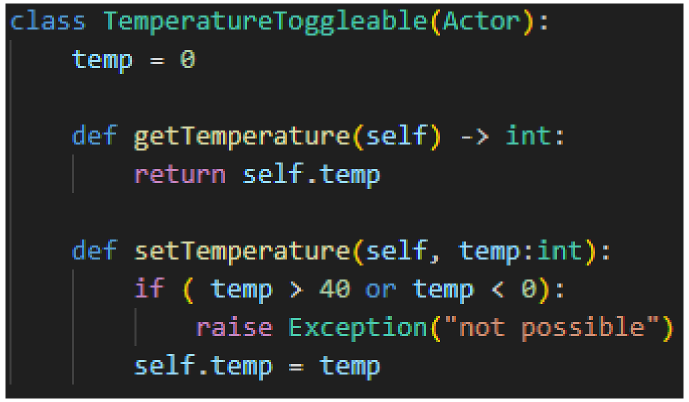
```python
class Tv(AudioDevice):
    def __init__(self, name: str):
        print("TV " + name + " has been created")
        super().__init__(name)

    def playVideo(self):
        if self.isRunning():
            print("VideoDevice " + str(self.name) + " is playing video")
```

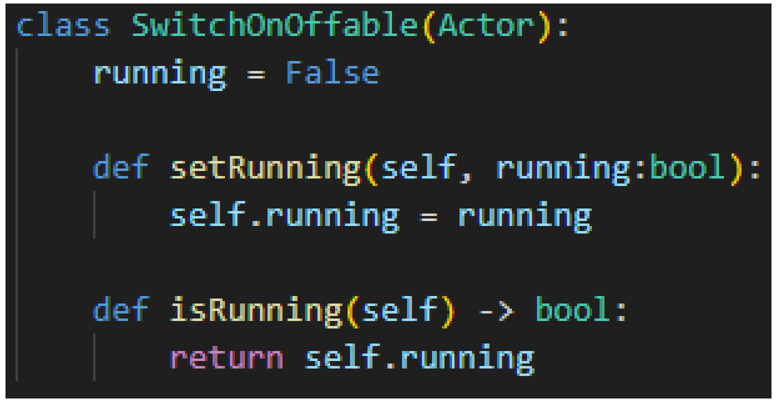
ISP


```python
class Actor(ABC):

    @abstractmethod
    def activateNightMode(self):
        pass
```

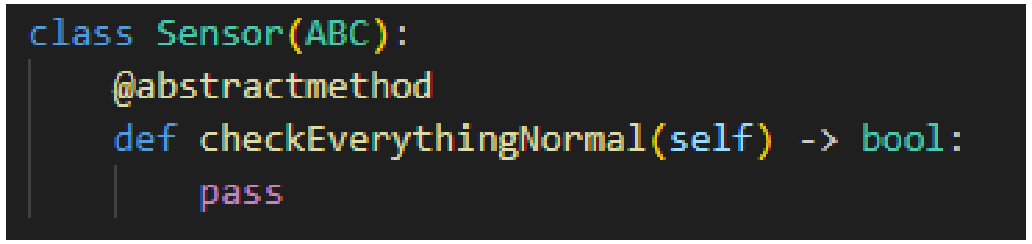

```python
class OpenCloseable(Actor):
    open = False

    def isOpen(self) -> bool:
        return self.open

    def setOpen(self, open:bool):
        self.open = open
```


```python
class IntensityToggleable(Actor):
    intensity = 0

    def getIntensity(self) -> int:
        return self.intensity

    def setIntensity(self, intensity: int):
        if intensity > 100 or intensity < 0:
            raise Exception("not possible")
        self.intensity = intensity
```

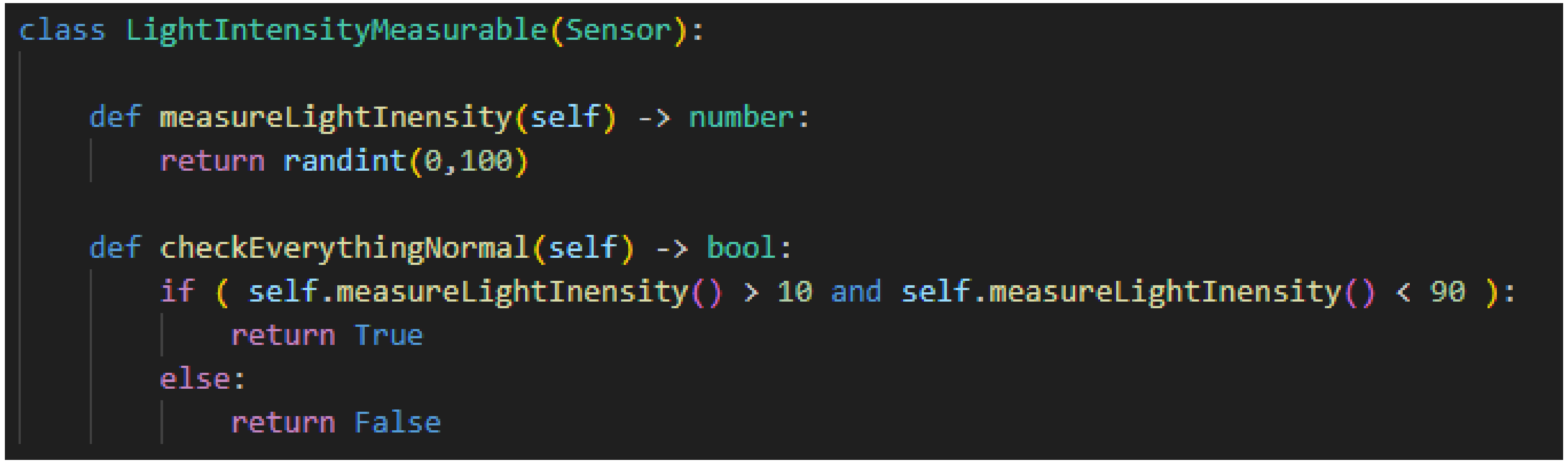

```python
class TemperatureToggleable(Actor):
    temp = 0

    def getTemperature(self) -> int:
        return self.temp

    def setTemperature(self, temp:int):
        if ( temp > 40 or temp < 0):
            raise Exception("not possible")
        self.temp = temp
```


```python
class SwitchOnOffable(Actor):
    running = False

    def setRunning(self, running:bool):
        self.running = running

    def isRunning(self) -> bool:
        return self.running
```

DIP



```python
class Sensor(ABC):
    @abstractmethod
    def checkEverythingNormal(self) -> bool:
        pass
```



```python
class LightIntensityMeasurable(Sensor):

    def measureLightInensity(self) -> number:
        return randint(0,100)

    def checkEverythingNormal(self) -> bool:
        if ( self.measureLightInensity() > 10 and self.measureLightInensity() < 90 ):
            return True
        else:
            return False
```