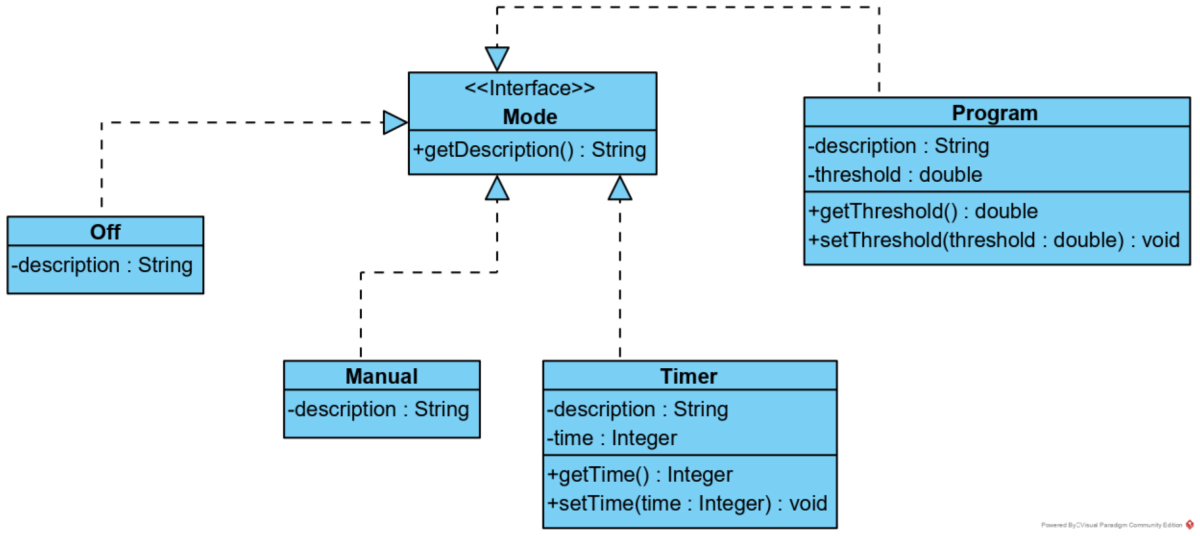
Design principles used:

- The Single Responsibility Principle (SRP) is used basically by the class Thermostat which deals only with changing the instances of one field for changing modes and let's the work to be done by the classes who inherit the interface Mode to manage the modes.
- The Open/Closed Principle and Liskov Substitution Principle are used by the interface Mode because if there is a need for more modes to be added for the thermostat only one class which implements Mode needs to be created.
- The Interface Segregation Principle regarding class Mode because it has only one method namely "getDescription" needed to do it's tasks and solve the modes management for the thermostat.
- The Composition Over Inheritance Principle was used for logging the events by adding the class History (which is basically a List of Strings where each string is the state of the thermostat or an action performed on the thermostat).

Design patterns used:

- Composite pattern
  I used the Composite pattern for the functional modes of the thermostat. I created an Interface "Mode" which takes a field inside the Thermostat class and this interface is implemented by 4 classes (Off, Manual, Timer and Program) each class acts like the interface Mode and by checking the instances we can make the code behave according to the specific behavior of the modes.



- Chain of responsibility pattern
  I used this pattern for logging the states of the thermostat. Basically I created the class History which is used to store the past information for the thermostat and it's modes. Each thermostat has a field for logging so every Thermostat instance will have a history regarding it's past functioning.