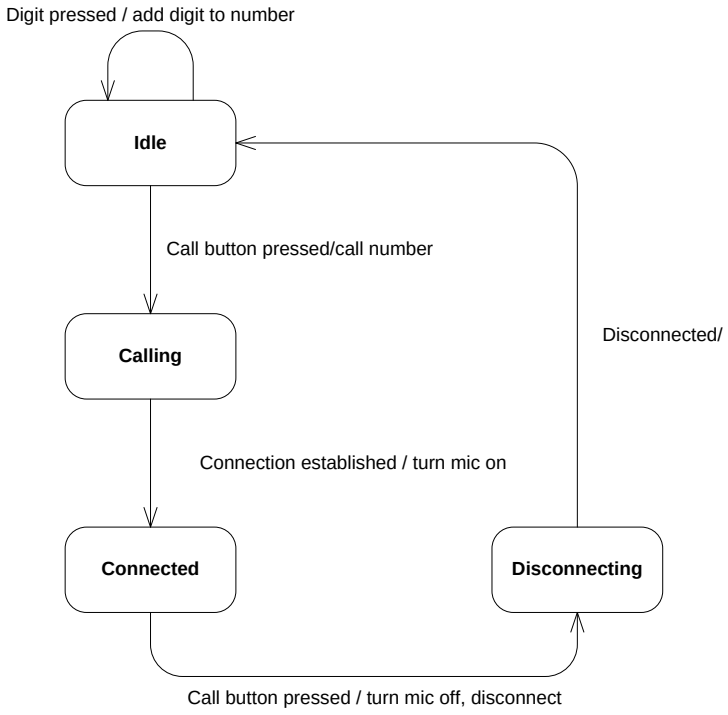
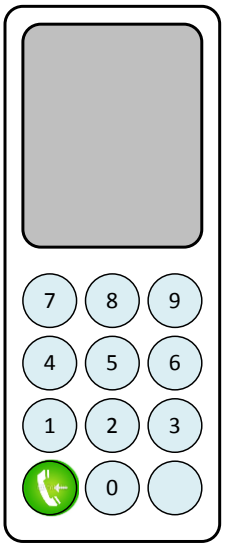
Exercise: The GoF State Pattern

In this exercise you will design the business logic of a simple telephone by means of the GoF State Pattern. You will incrementally add functionality to the system, thereby gaining insight into the implementation of state machines.

The situation: You are to design and implement a simple telephone along the lines of the below state chart, supplied by a sales representative from a meeting with a customer:





Exercise 1:
As usual, the sales representative left a number of questions unanswered, such as

- What happens if the line is busy? If the dialed number is invalid?
- How should we handle if the other party disconnects during the call?
- …

Amend the state chart above to answer these questions and any other questions you may have – it's your decision!

Exercise 2:
Identify the source of the events in the diagram. Some are from the phone's GUI, some are from the telephone system.

Exercise 3:
Implement and test your solution. You may need to create stubs for the microphone and GUI, but thankfully you know how to do that ☺.

Since you are using the GoF State Pattern, you can unit test each state to ensure that the transitions made and the actions taken are correct. Do that.

Exercise 4:
The sales representative has had another meeting with the customer. He has two further requests for the no-longer-so-simple-telephone: When a call is connected, it shall be possible to…

1. Toggle between the use of regular speaker or loudspeaker (speakerphone)
2. Mute the microphone during a call

Again, amend these changes (hint: think orthogonal substates) to your design, then –and *only* then – implement your design.