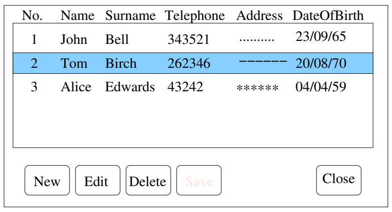
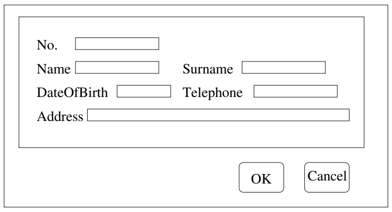
Example

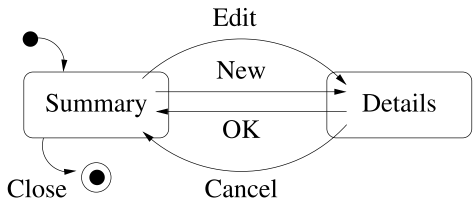
- Design the statechart for an application for updating a database of people
- the application will have two screen:
  1. a Summary screen, listing all the record in the database (in a concise form)
  2. a Detail screen, in which a record can be edited





The High Level Statechart

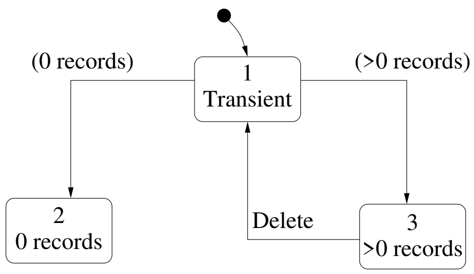
- the starting point is to identify the top level states
- in most cases these corresponds to the main screen of the application
- the events that cause navigation between the screen should be identified (even if not with precise details at this stage)



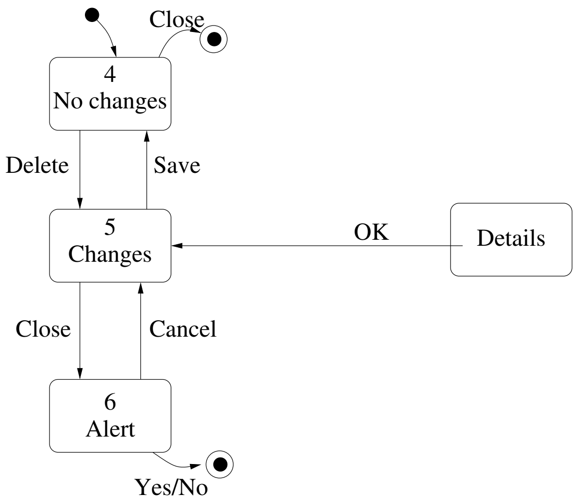
The Summary Screen

- Screen Rules:
  - objects with constant behaviour:
    - scrolling list: the user clicks a record, that will be highlighted. Only one record can be selected at a time. The default behaviour of a scrolling list can be used for this application
    - New button: causes the Details screen to appear, and its enabled all the time
  - objects with varying behaviour:
    - Edit/Delete buttons: enabled only if there are items in the list
    - Save button: enabled only if changes were made
    - Close button: if changes were made and not saved, an alert message is displayed, otherwise the application is closed
  - entry events: when the application is started, when OK or Cancel are clicked in the Details screen
  - exit events: when Close, New or Edit button are clicked
- Screen modes: none

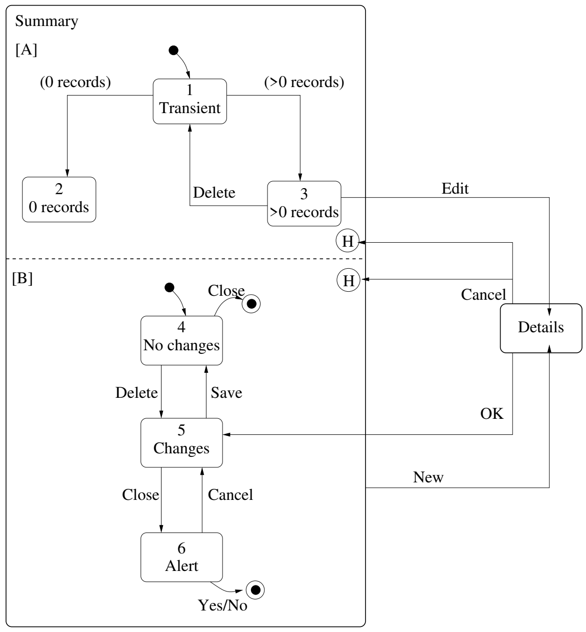
- Statechart design:
  - as there are no screen modes, consider the objects with varying behaviour
  - the behaviour of Delete and Edit is similar (both enabled or both disabled)
  - it depends on how many records there are in the list (i.e. on a response from the database management system)
  - create a state ("0 records") in which they are both disabled and a state (">0 records") in which they are both enabled (consolidating related behaviour)
  - from the ">0 records" state, a Delete makes the DBMS re-elaborate the list



– the other two varying behaviour objects (Save and Close) identify another pair of states in which the Summary screen can be: “changes have been made” and “changes have not been made”
– the former is influenced by clicking on the Delete button, or by the behaviour of the Details screen
– saving changes will bring the system again in the state “no changes”



- the two groups of states identified have a concurrent behaviour:



The Details Screen

- Screen Rules:
  – objects with constant behaviour:
    * Cancel button: will always get back to the Summary screen
    * Text fields: they are “passive” fields (they receive values from the user but do not perform any action when the user interacts with them)
  – objects with varying behaviour:
    * The OK button: if some mandatory field in not filled in, an Alert message will appear
  – entry events: when New or Edit are clicked in the Summary screen
  – exit events: when Cancel is clicked and when OK is clicked and changes are accepted
- Screen modes: none
- Statechart design:
  – the screen behaves in the same way regardless of whether the user is creating a new record or editing an existing one
  – the only difference is when the screen is first entered and when it is exited by clicking on the OK button
  – it is then necessary for the application to keep track of these two states
  – the check after clicking the OK button may be achieved by using prioritized events