Assumptions

Task 1: Modelling (MOD)

This document includes assumptions and description of my implementation.

The implementation of Streaming Library and Subscription management had to follow the MVC pattern. When swing components trigger some events, controller synchronizes the model and the view correspondingly:

```
((DefaultTableModel) tableAC.getModel()).addTableModelListener(new TableModelListener() {
      @Override
      public void tableChanged(TableModelEvent e) {
          if(e.getType()==0)
            try {
                controller.updateAC(e.getFirstRow(),e.getColumn());
            } catch (ParseException e1) {
                throw new IllegalArgumentException();
            }
      }
});
```

Figure 1: Update example

Other minor notes:
There is no button for data-display, because it is always displayed automatically up-to-date;
There is no update-button, to update an instance variable user just has to double click on the table cell, enter text and press enter;
To delete an entry, user has to select it in the table and press the delete-button;
While updating/creating data, user can't proceed if entering some senseless data (duplicate id, wrong formats for booleans or dates);
In the Streaming Library table the user can't change the instance type;
In the Accounts table user can't change the amount of profiles, without creating a new profile;
In the Profiles table user can't change the id of the according account (it can be changed in the Accounts table though);
Accounts and Profiles tables are synchronized;
All views and models are synchronized;