308-304B – Object-Oriented Design
Assignment 2

Due date: Saturday February 23, 2002 before 23:55

Practical information

- 1 <= team size <= 2
- Each team submits only one full solution. Include name(s) and ID(s) in *every* submitted file. Use the `index.html` template provided on the assignments page. The other team member (if any) *must* submit a single `index.html` file containing coordinates of both team members. This will allow us to put in grades for both team members in WebCT.
- Your submission must be in the form of a simple HTML file (`index.html`) with explicit references to all submitted files as well as inline inclusion of images.
- The submission medium is WebCT.
- Use a drawing tool such as `xfig` (for drawing state diagrams) and `xpns` (for modelling and simulating Petri Nets). Both are installed on all SOCS FreeBSD machines. Java as well as C sources of the PNS Petri Net simulator can be obtained from the PNS modeller/simulator website. Don't forget to set "Sequential Manual" simulation mode in xpns to get insight into the dynamics.

FSA model

Draw a FSA design model (include the image inline) and give some comments for the following requirements:

In our spreadsheet application to-be-built, we want tight control over what the user types into a spreadsheet cell. In our Python/Tkinter implementation, each user keystroke will result in a KeyPressed event. The event notice will contain information about which key was pressed. In particular, we will assume the following classes of input:

- `<DigitChar>` in the range `0 - 9`
- `<AlphaChar>` in the range `a - z, A - Z`
- `<NonAlphaChar>` in { `~!@#$%^&*()-_+;:'",<>?[]` }.
- `<period>` = '.'
- `<equals>` = '='
- `<IllegalChar>` any keyboard character (but not any keypress) not in the above. Note how we have chosen not to accept `|, {, }`
- `<EnterCell>` any means by which a cell is entered for editing.
- `<EndCell>` any means by which a cell is exited. Both `<EnterCell>` and `<EndCell>` may consist of a `MouseMove` followed by a `MouseClick` in a cell or of a keypress in {`ENTER`,`RETURN`,←,→,↑,↓}.

Use the notation `d:<DigitChar>` to denote access to the actual digit `d` (from the event notice) in your FSA model. Use only the above abbreviated notations in your model (for example, do not write `a - z, A - Z` explicitly, but write `c:<AlphaChar>`).

Your design must accept the following input sequences, given in Regular Expression form:

- INTEGER: `<EnterCell><DigitChar>+<EndCell>`
- REAL: `<EnterCell><DigitChar>*<period><DigitChar>*<EndCell>`
- FORMULA: `<EnterCell><equals>(!<IllegalChar>)+<EndCell>`
- STRING: `<EnterCell>(!<equals>)(!<IllegalChar>)*<EndCell>`
  Also, it should not be an INTEGER or a REAL.

In the above, the following meta-characters are used:

- `X+` means a sequence of one or more `X`
- `X*` means a sequence of zero or more `X`
- `!X` means anything but X
- `(XY)` groups X followed by Y
- `(X|Y)` means X or Y

We assume that if an entry was already present in the selected cell, it will be overwritten (no editing of the existing entry). Note how this assumption has some repercussions on initialization.

`<IllegalChar>` keystrokes are not used in constructing the final cell entry. A warning message is however sent to the object `messageArea`.

It is obvious that after a number of keystrokes, we may recognize an INTEGER, only to later find it is really a REAL, and after some more keystrokes, that really, a STRING is entered. This should be reflected in your FSA model.

Some objects are available to the FSA:

- `stringOrFormulaValue` (initially `""`)
- `integerValue` (initially 0)
- `realValue` (initially 0.0)
- `type` (initially UNKNOWN, can take on values UNKNOWN, INTEGER, REAL, STRING, or FORMULA)

At the end of processing a cell, the above values must be set appropriately by the automaton.

Multi-user spreadsheet

1. Build a Petri Net model of the behaviour of a distributed, multi-user spreadsheet (in particular, assume 3 users). You should build the model using the `xpns` tool. Include (inline) a few representative screen grabs taken during a simulation. Provide comments of what is happening. Also include a link to the `xpns` model file.

   We assume that

   - There is a single software object which "has" the actual spreadsheet data. All users view this same data. In a later assignment, we will see how the Observer Pattern will allow updating all views in case of changes.
   - There is mutual exclusion in write-access to the spreadsheet data.
   - Initially, no user has write-access. The user has to request a "token" (or "chalk" if you see the central data as a blackboard). If/when a user gets the token, he/she will have write access. When no more write access is needed, the user needs to return the token.
   - The life-cycle of a spreadsheet user consists of alternating periods of viewing and writing.

2. Prove from the model that it will never be possible for more than one user to be modifying the shared spreadsheet data.