## Tutorial 3

1) Evaluate the schema names below:
   - (a) add Numbers
   - (b) addNumbers
   - (c) Add Numbers
   - (d) AddNumbers - valid
   - (e) Add_Numbers
   - (f) AddNumbers5
   - (g) Addnumbers

2) In this question, we are going to formalize some of the ideas in the management of bank accounts.
   - (a) Introduce *ACCNO* as a given set (basic type) for account numbers.

     [ACCNO] - the set of all account number in the bank

   - (b) Introduce the following sets in a single declaration
     - (i) *active* - the set of account numbers that are in use

       active : $\mathbb{P}$ ACCNO

     - (ii) *overdrawn* - the set of account numbers of the accounts that are overdrawn

       overdrawn : $\mathbb{P}$ ACCNO

     - (iii) *depositor* - the set of account numbers of deposit accounts (accounts that receive interest)

       depositor : $\mathbb{P}$ ACCNO

     - (iv) *current*- the set of account numbers of current accounts (accounts that receive no interest)

       current : $\mathbb{P}$ ACCNO

       OR

       active,overdrawn,depositor,current : $\mathbb{P}$ ACCNO

```
__BankAccount_______________________________________
active,overdrawn,depositor,current : ℙ ACCNO
____________________________________________

____________________________________________________
```

   - (c) Write predicates to formalize the following statements:
     - (i) Only active accounts can be overdrawn

       overdrawn $\subseteq$ active

     - (ii) Only current accounts can be overdrawn

$\text{overdrawn} \subseteq \text{current}$

(iii) The active accounts are the deposit accounts and the current accounts taken together
$\text{depositor} \cup \text{current} = \text{active}$

(iv) No account can be both a deposit account and a current account
$\text{depositor} \cap \text{current} = \varnothing$

(v) No deposit account can be overdrawn
$\text{depositor} \cap \text{overdrawn} = \varnothing$

__*BankAccount*______________________________________

$\mathit{active, overdrawn, depositor, current} : \mathbb{P}\ \text{ACCNO}$

---

$\text{overdrawn} \subseteq \text{active}$
$\text{overdrawn} \subseteq \text{current}$
$\text{depositor} \cup \text{current} = \text{active}$
$\text{depositor} \cap \text{current} = \varnothing$
$\text{depositor} \cap \text{overdrawn} = \varnothing$

3) You have been asked by your project manager to model a system for a tennis club in a university. The club has the sole use of the university's tennis court. To use the tennis court, the student has to be a member of the club. To ensure that everyone gets enough games, there is a limit of 20 students that is allowed to be in the court area at any one time. Students at the court area are either playing a game at the tennis court, or waiting to play. Those who are waiting cannot be playing at the same time.

There is only one basic type involved:

[STUDENT] - the set of all students who can be members of the club

The court area has a limit on the number of students allowed at any one time:

$\mathit{maxStudents} : \mathbb{N}$

---

$\mathit{maxStudents} \leq 20$

Construct a state space schema called *TennisClub* that will specify the following properties of the system:

- *members* which describes the set of all members of the club
- *courtArea* which describes the set of all members who are in the court area
- *onCourt* which describes the set of those who are playing the game
- *wait* which describes the set of all members who are waiting to play the game

You must show all the conditions of your properties in the predicate part of your state space schema.

*TennisClub*

$members : \mathbb{P}\ STUDENT$
$courtArea : \mathbb{P}\ STUDENT$
$onCourt : \mathbb{P}\ STUDENT$
$wait : \mathbb{P}\ STUDENT$

---

$courtArea \subseteq members$
$\#courtArea \leq maxStudents$
$onCourt \cup wait = courtArea$
$onCourt \cap wait = \varnothing$