THE RETIREMENT GAME

I. Description:
The player starts at age 20 with a starting amount of cash, the goal is to accumulate enough assets that can generate required cash flow at age 65 to cover retirement.

The player can choose step size (1, 2, 5, 10 yr), at each step and set the course for that period. The player can stay the course or choose from a set of actions (work_more, work_less, study, buy or sell investments, buy lottery, buy house, raise kids) that take effect for the next time period.

II. Features:
- Buying house and having kids increases current spending but reduces the level of cash flow needed for retirement.

- Current "spending" uses cash, but makes player happier and healthier --> less cash flow needed for retirement at 65

- Total "time" available to the player per day is a constant (thus "work_more" means less time for "rest" --> too much work leads to increases in cash needs for retirement (for healthcare)

- Study uses "time" and "cash", but it increases player's education level --> more earning power later

- At 65, income from work becomes 0, the player will rely on cash flow generated by investments, if the cash flow does not meet the required level, the player fails the game.

- At 65, the case flow from work becomes 0, if the cash flow from bonds/stocks does not meet the requirement for retirement, the player loses the game.

-Difficulty of the game can be increased by giving the player less cash at the beginning of the game, or by reducing the salary for work, etc.

- Player can use "q" to quit and "h" for help at any point of the game

The game may play like this:
...

Now you are at age 40.
You have:
Investments: 5000
House equity: 2000
Cash: 1000
Kids: 2
Education level: 10
Work time per day: 8
Study time per day: 0
Income per year: 2500
Spending per year: 2000
Required cash flow per year for retirement: 1200

What would you like to do? s
What is the next step size (yrs)? 5
What else would you like to do? b
What would you like to buy? i
How much investments would you like to buy? 500
...

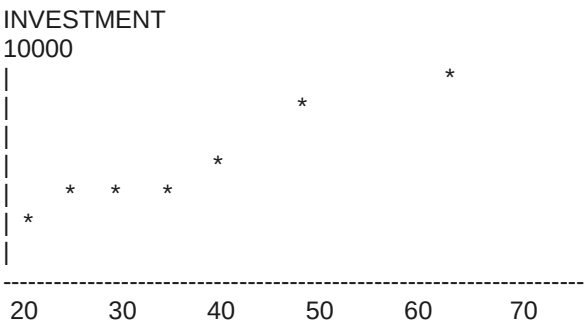
Now you are at age 65.
You have:

Investments: 10000
House equity: 2000
Cash: 2000
Kids: 2
Education level: 20
Work time per day: 0
Study time per day: 0
Income per year: 1600 (# generated by investments instead of work)
Required cash flow per year for retirement: 1500

Congratulations! You won! You can start to enjoy your life now.

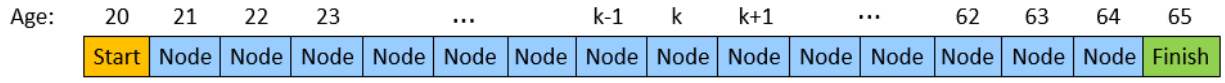
May include an option for simple graphics:

For example:
What would you like to do? d
What would you like to display? i



III. Design

Update 03/01/17



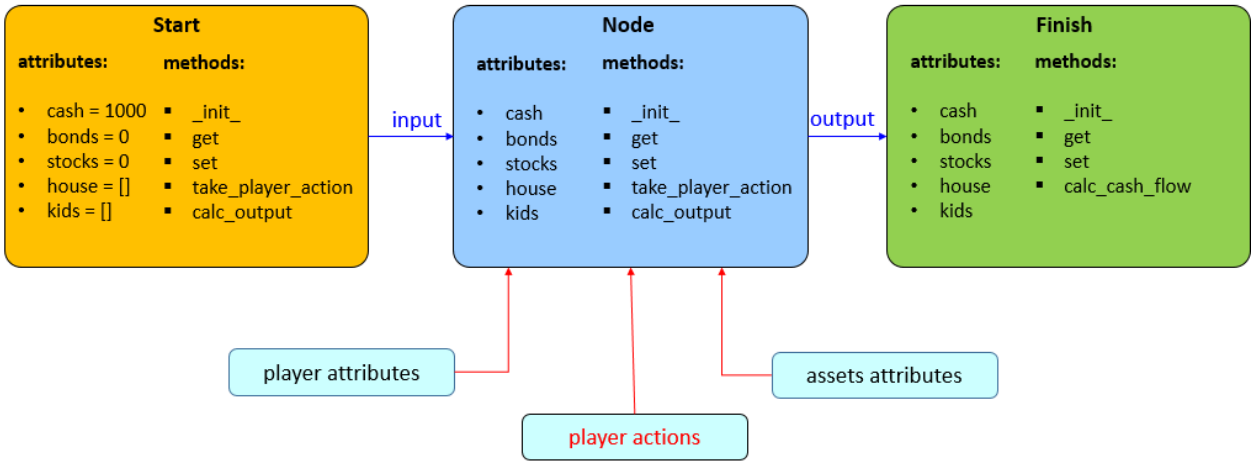
The game is played as if the player moving through a series of Nodes connected together, taking actions along the way, causing changes in the asset accounts.

Each Node takes the inputs from the previous node and decides that outputs, which will be used as inputs by the next node.

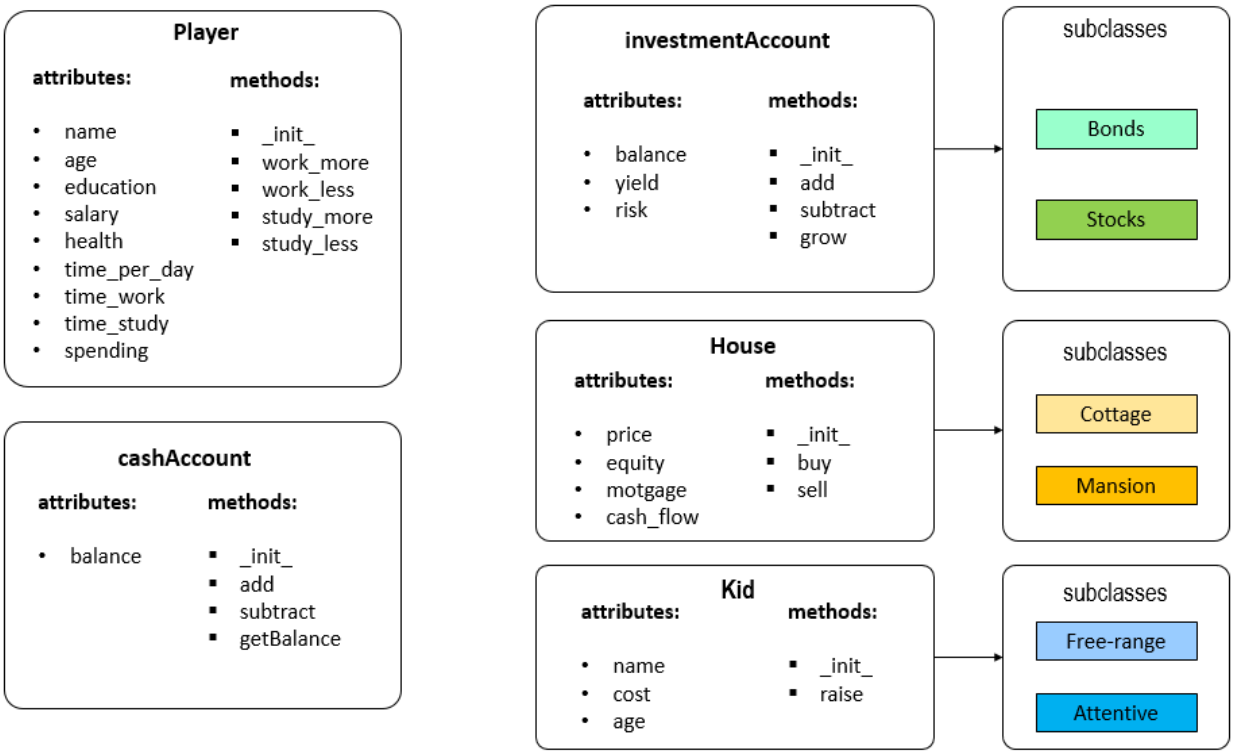
The Node also takes the actions from the user. These actions, together with the attributes of the player and assets classes, determine the outputs of that node.

“Start” and “Finish” can be implemented as subclasses of the class Node.

The “Finish” node will have a method to calculate cash flow based on the accumulated assets at that time, thus determines the outcome of the game.



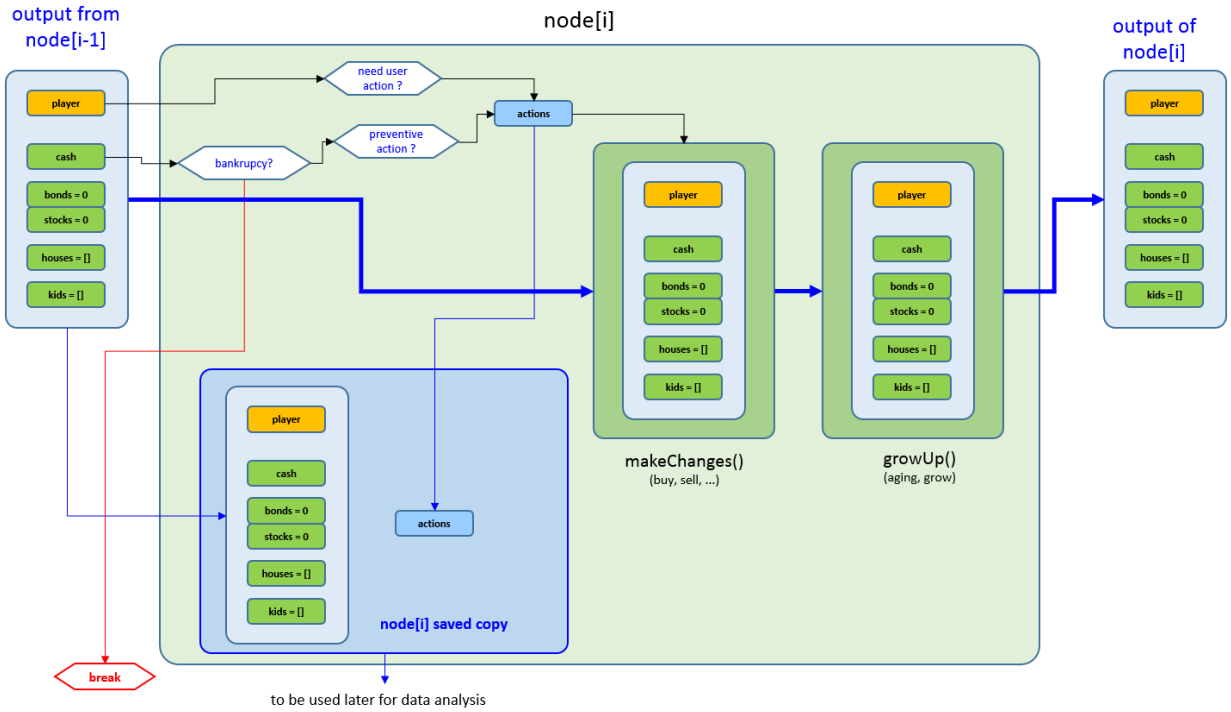
In addition to “Node”, I would also like to implement following classes for the player and various assets:



I may reduce the complexity of the problem later to keep the length of the program under control.

Update 03/02/17

Functions of a node:



The objects “player” and the player’s various assets will be passed to each node.

The node first save a copy of the inputs for itself, these copies are the data stored in the object node[i] and can be accessed later for data analysis.

Meantime, when objects “player” and assets move through the node[i], the node will first decide if the player have enough cash to cover the cash flow for this node (representing a year of the player’s life), it will prompt the user to sell assets if cash is insufficient. If the player does not have enough assets to cover the cash flow for that year, the program will break and the player loses the game.

Then the node will decide if it need to take user actions.

The user actions will become a vector that will be used to control how to change the “player” and assets objects.

The actions are performed by in two steps: first is the actions such as buy and sell that come from user intervention, second is actions such as aging or grow which do not come from user intervention.

Thus, the objects “player” and assets themselves are changed by node[i], and they will be passed to the next node[i+1] as inputs.

For the last node, instead of being passed on to another node, the “player” and assets objects will be passed to a evaluate() function, which will determine if the player has enough assets to cover the cash flow for retirement, thus determine the outcome of the game (win or lose)