THE RETIREMEMT GAME

Zhaoning Yu

(MIDS-INFO-W18 project 1 03/16/2017)

The retirement game is a personal finance simulator: the player starts at age 20, making a series of financial decisions during the course of the game, the goal is to accumulate assets that will provide sufficient income to cover retirement expenses after age 65.

I. FEATURES AND CLASSES

The game incorporates following classes and features:

Person (Class): represents the player

- **name**: input by player
- **age:** increases by 1 each time when the grow() method is called
- **skill**: higher skill improves income
- **health**: (1) declines over time, (2) the decline is faster if the player spends too much time on work and study, (3) bad health increases retirement expenses (for healthcare)
- **work_time**: income/yr is proportional to the time spent on work
- **study_time**: study improves skill, skill improves income
- **spending**: personal spending for basic needs ($10,000/yr) plus fun and pleasure, more spending slows down the health decline
- **wage**: starting hourly wage of $12.5/hr corresponds to an income of ~$20,000/yr
- **income**: determined by work_time and skill

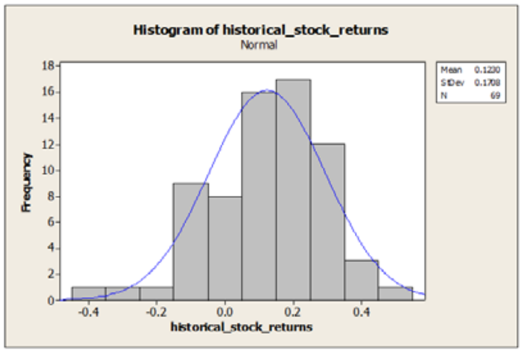
Account (Class): represents cash

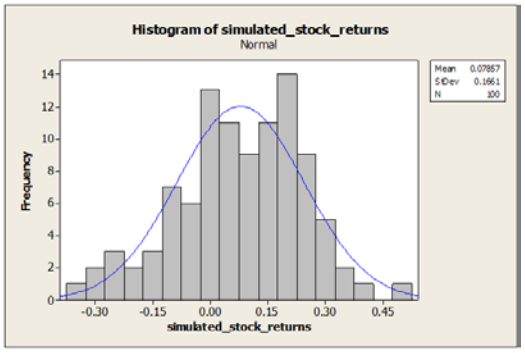
- **balance**: the amount of cash the player has
- **yield**: -2.0% for cash (to model inflation)

The parent class "Account" has two subclasses for bonds and stocks:

bondAccount (Class): yield = 2.5%, a safe investment with a steady return

stockAccount (Class): yield is a random normalvariate (μ=0.08, σ=0.17), mimicking real stock market historical returns:





House (Class):

- **name**: provided by player, serves as the key in the dictionary ‘houses’
- **age**: increases by 1 each time grow() method is called
- **type**: ‘shed’ for the base class
- **price**: $100,000
- **equity**: (1) accumulates over time, (2) is turned into cash when the house is sold

The player buys a house by paying off a 20-year mortgage, the payment per year is 10% of the house price, 50% of each payment goes into the equity (the rest is interest payment).

After 20 years, the house generates a cash income (2% of the house price) each year. In addition, the equity of the house also increases (by 1% of the house price) each year.

When a house is sold, the accumulated equity is deposited into the cash account.

The base class “House” has two subclasses, inheriting all the attributes and methods of the base class except ‘type’ and ‘price’. They represent more expensive houses:

cottageHouse (Class): type = ‘cottage’, price = $200,000

mansionHouse (Class): type = ‘mansion’, price = $500,000

Kid (Class):

- **name**: given by player, serves as the key in the dictionary ‘kids’
- **age**: increases by 1 each time grow() method is called
- **breed**: ‘FreeRange’ for the base class, representing a low-cost parenting style
- **expense**: $10,000/yr for ages 0-16, $30,000/yr for ages 17-20 (college years)

After age 20, the kid generates a cash income of $5000/yr for the player.

The base class “Kid” has one subclass, representing higher-cost parenting style:

fancyKid (Class): (1) breed = ‘Attentive’, (2) expense is $20,000/yr (ages 0-16) and $60,000/yr (ages 17-20), (3) generates a cash income of $10,000/yr for the player after age 20.

Node (Class):

A “Node” object is a holder of the following data objects:

- **person**: a ‘Person’ object
- **cash**: an ‘Account’ object
- **bonds**: a ‘bondAccount’ object
- **stocks**: a ‘stockAccount’ object
- **houses**: {} a dictionary containing ‘House’, ‘cottageHouse’, ‘mansionHouse’ objects
- **kids**: {} a dictionary containing ‘Kid’, ‘fancyKid’ objects

A node represents one year of the player’s life, it prompts the player for inputs and acts on the data objects in the node by performing following functions:

- **getNodeData()**: outputs data for the next node to copy
- **getStepSize()**: the number of years the game runs each time without asking for user action
- **setStepSize()**: default step size = 1, can be reset by user
- **getExpense()**: personal spending + expense on houses + expense on kids
- **getCash()**: cash balance + income
- **getAssets()**: cash + bonds + stocks + house equity
- **buyBonds()**: turns cash into bonds
- **sellBonds()**: turns bonds into cash
- **buyStocks()**: turns cash into stocks
- **sellStocks()**: turns stocks into cash
- **buyHouse()**: creates and adds a new ‘House’ object to the dictionary ‘houses’
- **sellHouse()**: removes a specified ‘House’ from the dictionary ‘houses’, deposits the equity to the cash
- **raiseKid()**: creates a ‘Kid’ object and adds it to the dictionary ‘kids’
- **setWorkTime()**: total work and study time cannot exceed 12 hrs/day
- **setStudyTime()**: study improves skill
- **setSpending()**: (1) personal spending on fun and pleasure, (2) higher spending slows down health decline, (3) must be >$10,000/yr to cover basic needs
- **buyLottery()**: a 1 in 10 chance of winning (10x the original amount)
- **checkBankrupcy()**: ‘true’ if expenses > total assets
- **checkCashFlow()**: ‘true’ if income + cash > expenses
- **printMainMenu()**: displays the main menu
- **getAction()**: prompts an action from the user
- **executeAction()**: interprets and executes the action
- **printNode()**: prints out data stored in the node
- **grow()**: (1) increases objects age by 1, (2) adds interest to cash, bonds, and stocks, (3) adds equity to houses

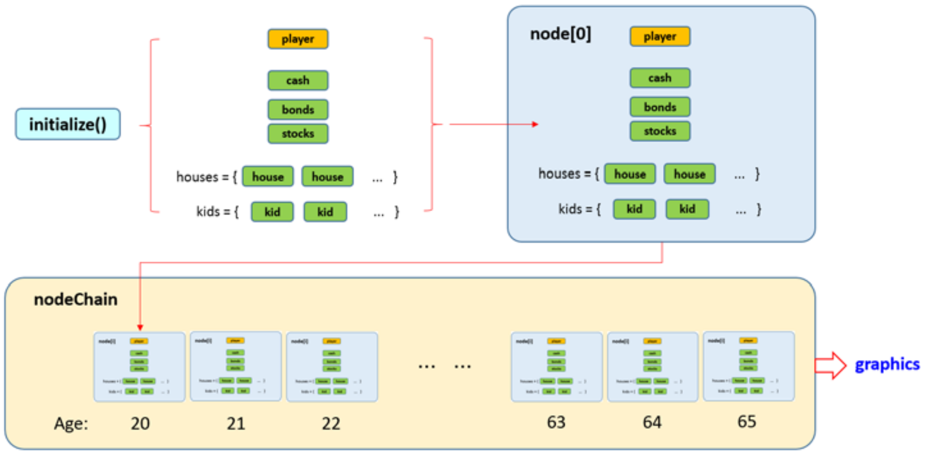
NodeChain (Class):

A “NodeChain” object called “history” holds a list containing all the Nodes, thus this object represents the life span of the player for the course of the game (from age 20 to 65 if the player finishes the game, but the chain may be truncated if the player is bankrupt before the age of 65).

The “NodeChain” performs following functions:

- **getChain()**: outputs the list of nodes stored in this NodeChain object
- **addNode()**: adds a new node to the chain
- **evaluate()**: evaluates the player performance at age 65, outputs a game summary
- **displayMenu()**: shows menu for graphics
- **drawGraph()**: takes a list of numbers then draws a graph
- **getData()**: retrieves specified data from the chain of nodes,
- **displayData()**: asks for user input, calls getData() and drawGraph() to plot data

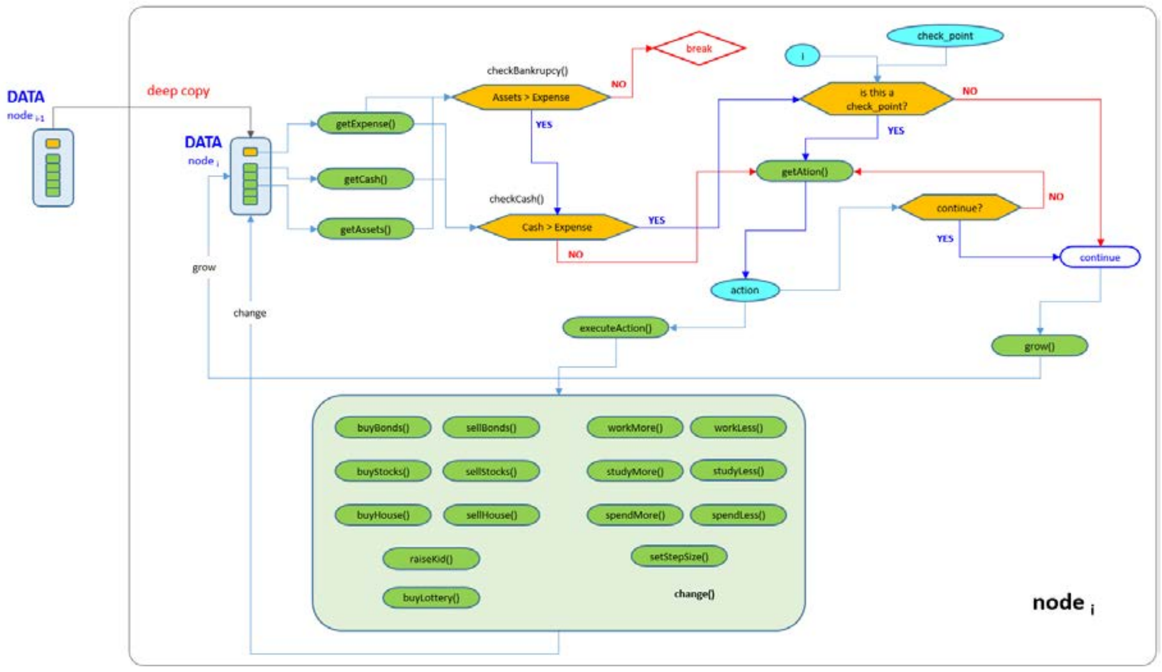
II. DESIGN OF THE GAME



Objects representing the player and the player's assets are saved in a series of 'Node' objects, each node represents one year of the player's life. Data objects in each note describe the state of the player and the player's assets at the end of that year.

All these node objects are packaged inside a list in a 'nodeChain' object called "history", which analyzes the data and outputs the graphics.

III. FLOW CONTROL

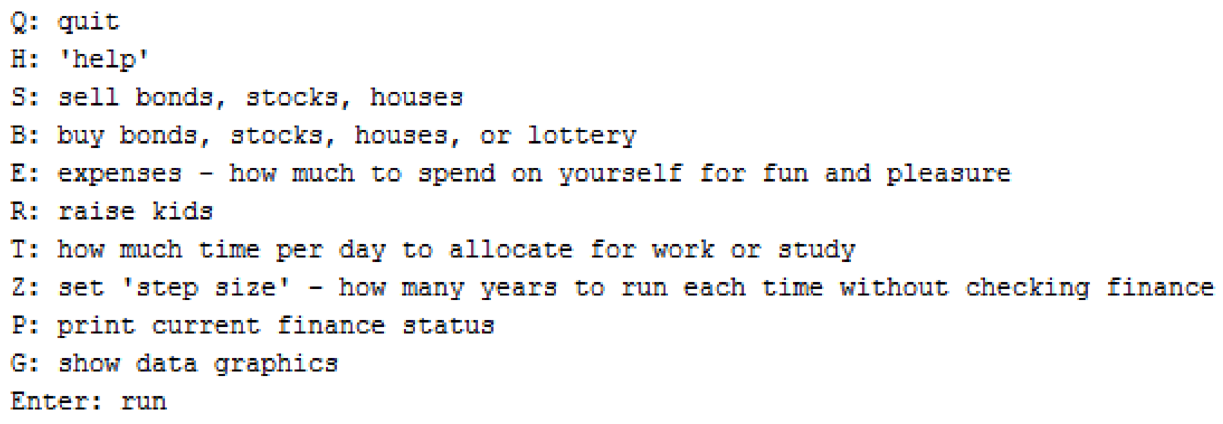


A node first makes a 'deep copy' of the data stored in the previous node, then it will check if the player has enough cash for that year, if the total assets are less than the expenses, the player loses and the game terminates.

Otherwise, the node will ask inputs from the player and acts on the data objects accordingly.

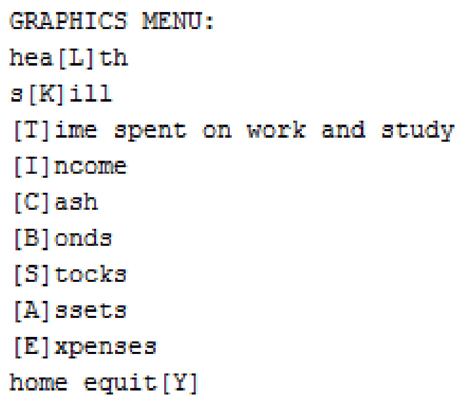
IV. USER MENU

The Node.getAction() method has following menu options:



```
Q: quit
H: 'help'
S: sell bonds, stocks, houses
B: buy bonds, stocks, houses, or lottery
E: expenses - how much to spend on yourself for fun and pleasure
R: raise kids
T: how much time per day to allocate for work or study
Z: set 'step size' - how many years to run each time without checking finance
P: print current finance status
G: show data graphics
Enter: run
```

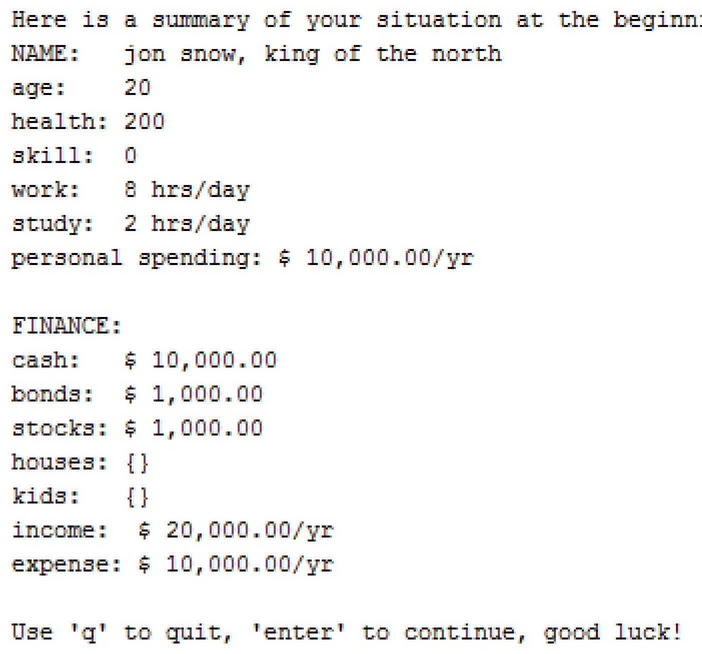
The NodeChain.displayData() method has the following menu options for graphics:



```
GRAPHICS MENU:
hea[L]th
s[K]ill
[T]ime spent on work and study
[I]ncome
[C]ash
[B]onds
[S]tocks
[A]ssets
[E]xpenses
home equit[Y]
```

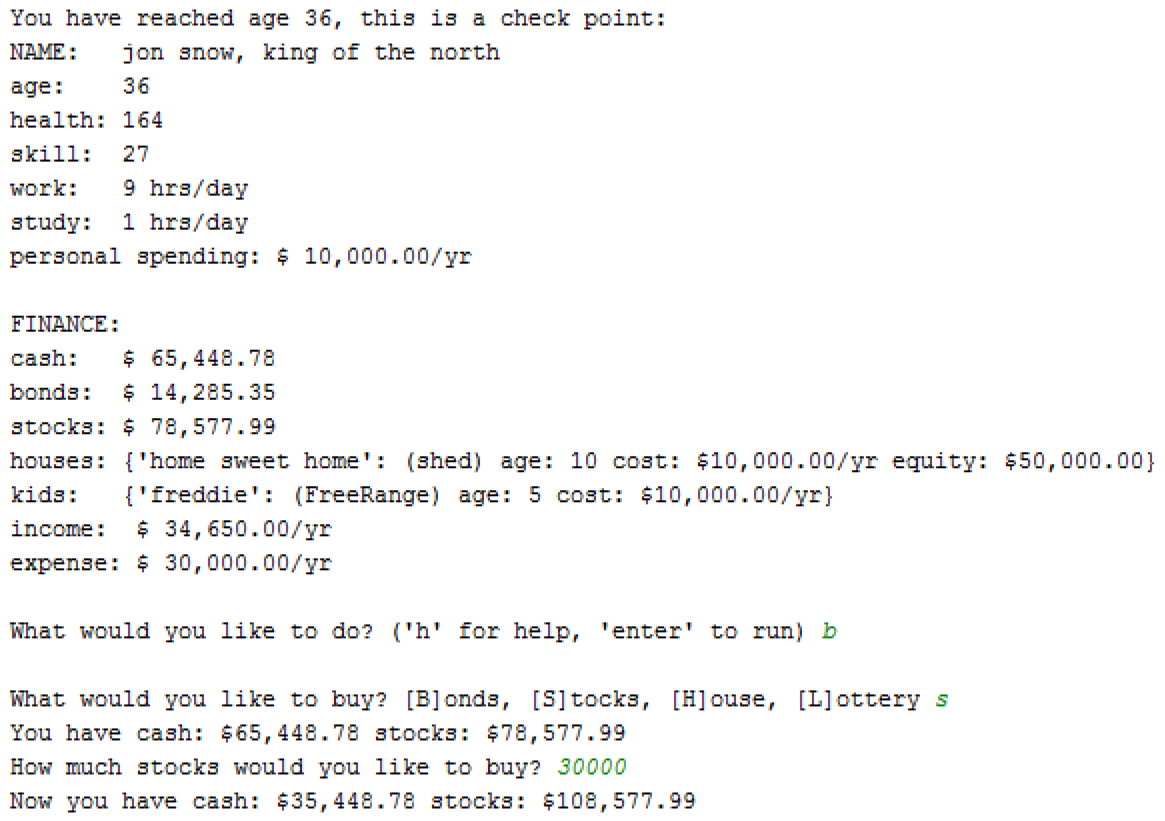
V. EXAMPLE GAME PLAY

1. Initial setup of the game:



```
Here is a summary of your situation at the beginning of your adult life:
NAME:   jon snow, king of the north
age:    20
health: 200
skill:  0
work:   8 hrs/day
study:  2 hrs/day
personal spending: $ 10,000.00/yr

FINANCE:
cash:   $ 10,000.00
bonds:  $ 1,000.00
stocks: $ 1,000.00
houses: {}
kids:   {}
income:  $ 20,000.00/yr
expense: $ 10,000.00/yr

Use 'q' to quit, 'enter' to continue, good luck!
```

2. Player taking actions in the middle of the game:

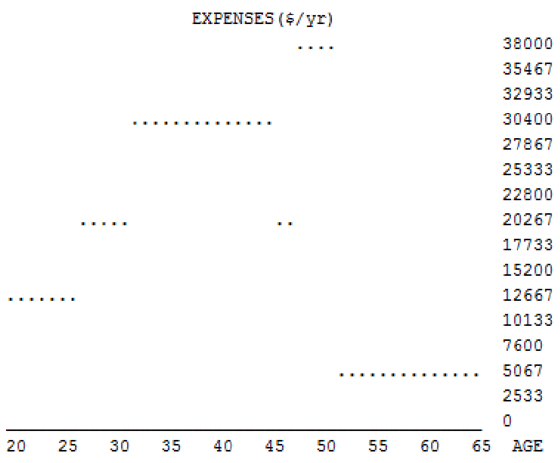


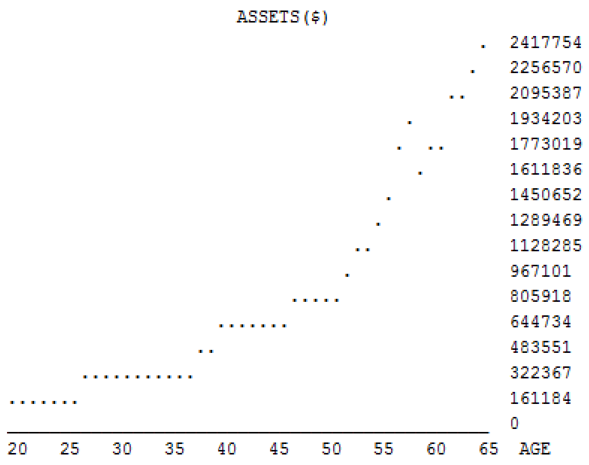
```
You have reached age 36, this is a check point:
NAME:   jon snow, king of the north
age:    36
health: 164
skill:  27
work:   9 hrs/day
study:  1 hrs/day
personal spending: $ 10,000.00/yr

FINANCE:
cash:   $ 65,448.78
bonds:  $ 14,285.35
stocks: $ 78,577.99
houses: {'home sweet home': (shed) age: 10 cost: $10,000.00/yr equity: $50,000.00}
kids:   {'freddie': (FreeRange) age: 5 cost: $10,000.00/yr}
income:  $ 34,650.00/yr
expense: $ 30,000.00/yr

What would you like to do? ('h' for help, 'enter' to run) b

What would you like to buy? [B]onds, [S]tocks, [H]ouse, [L]ottery s
You have cash: $65,448.78 stocks: $78,577.99
How much stocks would you like to buy? 30000
Now you have cash: $35,448.78 stocks: $108,577.99
```

3. The performance summary at the end of the game:

```
What would you like to do? ('h' for help, 'enter' to run)
You have reached age 65, you have accumulated $2,417,753.73 in total assets:
NAME:   jon snow, king of the north
age:    65
health: 99
skill:  56
work:   9 hrs/day
study:  1 hrs/day
personal spending: $ 10,000.00/yr

FINANCE:
cash:   $ 191,169.92
bonds:  $ 175,422.01
stocks: $ 1,932,161.80
houses: {'home sweet home': (shed) age: 39 cost: $-2,000.00/yr equity: $119,000.00}
kids:   {'freddie': (FreeRange) age: 34 cost: $-5,000.00/yr}
income:  $ 47,700.00/yr
expense: $ 3,000.00/yr
Your estimated income of $60,443.84/yr for retirement is much higher than expenses: $23,250.00/yr.

EXCELLENT JOB! JON SNOW, KING OF THE NORTH! You can start to enjoy your life in retirement now.

You can choose to view your performance data before exiting the game.
```

4. Sample graphs showing the player's expenses and the value of assets over time:


EXPENSES($/yr)
20 25 30 35 40 45 50 55 60 65 AGE


ASSETS($)
20 25 30 35 40 45 50 55 60 65 AGE