Design Description

Program name : Project5_Design_Description_Schaefer_Kristin.pdf
Author : Kristin Schaefer
Date : 06-11-2019

Bus Class

Description: The Bus class is a derived class of Space. It acts like a Node, thus it has pointers to the Space node that is up/right/down/left of it. There can be multiple Bus objects in the map.

Key data members:

1. string spaceType = “Train”
2. string printType =

```
“\n
 _ _ _ _ _ _\n
|             |\n
     Bus      \n
|_ _ _ _ _ _|\n”
```

Key functions:

1. genRandomEvent()
   - Overrides function
   - Generates a random event number
     1. Bus is running late
        - stopwatch +5
        - stress level +1
     2. No event
2. topUp()
   - added function for bus to top up transitMoney if <=$5
   - transitMoney +$10
   - stopwatch +5
3. takeBus()
   - transitMoney -$5
   - stopwatch +10
   - stress level +2

Shop Class

Description: The Shop class is a derived class of Space. It acts like a Node, thus it has pointers to the Space node that is up/right/down/left of it. There can be multiple Shop objects in the map.

Key data members:

1. string spaceType = “7-11”
2. string printType =

```
“\n
 _ _ _ _ _ _\n
|             |\n
     7-11     \n
|_ _ _ _ _ _|\n”
```

Key functions:

1. genRandomEvent()
   - Overrides function
   - Generates a random event number
     1. Buy a coffee
        - stopwatch +5
        - stress level -1
     2. Donate $5 to a homeless man outside
        - stopwatch +5
        - stress level -2

Design Description

Program name : Project5_Design_Description_Schaefer_Kristin.pdf
Author : Kristin Schaefer
Date : 06-11-2019

Game Class

Description: The Game controls the simulation of the game as well as the creation and printing of the game map. The game map consists of a linked list structure of Space objects, and it treats each Space object like a node.

Key data members:

1. Space *headPtr, Space *tailPtr
    - Head pointer (Home)
    - Tail pointer (Work)
2. Space *posPtr
    - Player's position on the map pointer
3. int spaces
    - Keeps track of the Space's index in order to set the correct up/down/left/right positions
    - 16 total spaces on the map
4. bool atWork
    - Boolean value representing if the Player has arrived at Work or not
    - the Game simulation stops if atWork is true

Key functions:

1. Game()
    - Constructor calls function setMap() to create the board of Space objects representing the map
    - Map is a queue structure
2. setMap()
    - Sets the Space object head ptr to a Home object
    - Sets the Space object tail ptr to a Work object
    - Randomly sets the remaining Space objects in the queue
    - Total of 16 spaces
2. set/getFront(), set/getBack(), set/getPos()
    - Setters and getters for the pointers to the head, tail and pos Space objects
3. printMap()
    - Prints the board of Space objects representing the Game map
    - Overrides the map where the Player currently is in order to print the Player's position
4. startSim()
    - Controls the rounds of the game simulation
    - Needs to get the user's direction choice and update the Player's position on the board accordingly
    - The game ends if:
        - The player has $0 or less money
        - The player's stress level reaches 10 or greater
        - The player runs out of time
        - The player arrives at Work (the tail pointer)
    - If the player loses, a message is printed stating why
    - The player needs to have their laptop, phone, and id pass to go to work and win the game
4. int genRandomNum(int min, int max)
    - Generates a random number from min to max
    - Helps to randomly assign Space objects for the board/map
5. printRound()
    - Prints the Player's money, transit money, stress level and current time
6. bool gameWonLost()
    - Return boolean value atWork

Design Description

Program name : Project5_Design_Description_Schaefer_Kristin.pdf
Author : Kristin Schaefer
Date : 06-11-2019

Player Class

Description: The Player class contains a queue structure of strings representing the player's bag and the items within it. It also holds the variables for the player's money, transit money, stress level and the time.

Key data members:

1. queue <string> bag;
    - The queue structure is a container representing the Player's work bag
    - The bag can be filled with items which are represented by strings
        1. string wallet = "wallet"
        2. string phone = "phone"
        3. string laptop = "laptop"
        4. string transitPass = "transit pass"
        5. string workID = "work ID"
    - The player can lose items from their bag as the game is played. If the bag is empty, the game ends
    - The player must have their laptop, phone and workID to go to Work and complete the game
2. int money
    - Integer representing the Player's amount of money
    - The game ends if the Player's balance is $0 or less
    - The Player starts out with $20
3. int transitMoney
    - Integer representing the Player's amount of money on their transit pass
    - The Player starts out with a balance of $10
    - If the balance is $0 or less, the Player must top up their card to continue traveling to Work (advance Spaces)
3. int stress
    - Integer representing the Player's stress level
    - If the stress level goes over 10, the game ends
4. int stopwatch
    - Integer representing the amount of time (min) the Player has left to get to Work (derived class of Space)
    - Incremented / Decremented in units of 10
    - If the value reaches 60 (1 hr), and the Player is not at Work, the game ends

Key functions:

1. Player()
    - Constructor fills queue (bag) with strings representing items
    - Assigns int values to money, transitMoney, stress and stopwatch
2. set/getMoney(), set/getTransitMoney(), set/getStress(), set/getStopwatch()
    - Setters and getters for the integer private member variables of the Player class
3. addBack()
    - Adds a string representing an item in the Player's bag to the back of the queue
4. removeFront()
    - Removes the front string item from the Player's bag / queue
    - Represents that the player has lost an item from their bag
5. isEmpty()
    - Checks if the queue is empty
    - If the queue is empty, the game ends (represents that the Player has lost all items in their bag)
6. checkBag(string itemCheck)
    - Check if specific item is in the Player's bag

Program name : Project5_Design_Description_Schaefer_Kristin.pdf
Author : Kristin Schaefer
Date : 06-11-2019

Menu function

Description: The Menu functions include a start menu, a direction menu for the Spaces on the map and a yes/no function to get a simple choice from the user. All of the menu functions check the user's input for validity.

Key functions:

1. bool menuStart()
    - Prints the name of the game
    - Prints the game objectives
    - Asks the user if they would like to:
        1. Play Game (bool True)
        2. Exit Game (bool False)
2. int menuDirection(Space *posPtr)
    - int dir = 4
    - Checks the posPtr for nullptrs, and decrements the dir count if the up, down, left or right ptr == nullptr
    - Asks the user to enter an int representing which direction they would like to travel on the board:
        1. Up
        2. Right
        3. Down
        4. Left
    - If the up, down, left or right ptr == nullptr, then the direction option is not displayed
    - Returns the int choice to the Game class so that the Player's position can be updated
3. bool menuYN()
    - Asks the user to enter either:
        1. Yes
        2. No

Design Description

Program name : Project5_Design_Description_Schaefer_Kristin.pdf
Author : Kristin Schaefer
Date : 06-11-2019