Design Reflection

Program name :	Project5_Design_Reflection_Schaefer_Kristin.pdf
Author :	Kristin Schaefer
Date :	06-11-2019

Design Changes and Problems Encountered

Due to the open-ended design format of the final project, I spent one day to create a game theme and a thorough design plan. The time investment helped tremendously, and I was able to write the program with few revisions.

The part of the program where I spent the most time and made the most revisions was the Game map. Initially I had planned to use a linked list structure with a counter variable and head and tail pointers. However, when creating the map I found that I would need to call certain Spaces with an index number, so I updated the map to consist of a dynamic 2d-array of Space pointers (similar to what our team used in the group project). Using an array allowed me to easily set the up, right, left and down Space pointers to the correct positions by using conditional statements regarding the row and column numbers. It then allowed me to easily call specific Spaces by their index number. I kept the head and tail pointers, as they referred to specific Spaces in my game design, and I also added Space pointers to the player's position and previous position to keep track of the player's location and to allow the player's location to be reset.

Also related to the Game map, I encountered errors while trying to make the printMap() function. In the design plan I had expected that I would use a loop to go through each Space and print the printType variable of each Space. I had formatted each Space printType to be in the following format:

```
"\n
 _ _ _ _ _ _\n
|             |\n
    Type     \n
|_ _ _ _ _ _|\n"
```

It didn't occur to me in the design plan that this would result in the spaces printing in one vertical column! I then reduced the printType to consist of " Type " and printed the vertical and horizontal lines within the printMap() function.

The Space class had a few small changes in comparison to the design plan. The first change was to add a private member variable for a pointer to a Player object. The constructor for each Space derived class was then updated to take a Player object pointer (passed from the Game class) as a parameter and the pointer was then set within the constructor. This allowed for the functions within the derived Space classes to access the bag contents and to get variables such as money, transitMoney, time and stress. I also added an additional function, spaceEvent(), to handle all of the actions within each derived Space class. The genRandomEvent() function was then reduced to produce a random integer within a specific range. I made this change because the genRandomEvent() function was growing too long, and it did not make sense to call functions like takeTrain() or takeBus() within the genRandomEvent() function. Additionally, I added two parameters to two Space object pointers for the spaceEvent() function. This allowed me to reset the player's position if one of the random events called for the player to return to a specific space.

Keeping track of and checking all of the variables for the spaceEvent() functions also proved challenging. For each random event I needed to check different conditions in order to make the game feel more realistic. For example, in order to take the train or the bus, the user needed to have a specific amount of money both in their wallet and on their transit card. This required multiple conditional statements to produce different outputs based on the player's variables. In the end, I spent quite a bit of time combing through each event to check that the time, stress, money and transitMoney variables were updated relatively realistically.