Benjamin Tate
3/15/16
CS 162 – Section 400
Final Project Design and Reflection

Design:

Character Class:
Protected data members: int movesLeft, string action, string item, string inspection, string movement, int keyCount, int whiskeyCount, vector of strings for inventory, vector of strings for surrounding spaces

Public member functions:

- Character()
  - Sets movesLeft to 150
  - Sets keyCount and whiskeyCount to 0
  - Creates Car pointers for each space
  - Creates pointer to train object with spaces as parameters
  - Runs game()
- Void Game(Train*)
  - Print intro
  - While action != exit, end is not reached, and movesLeft > 0
    - Print movesLeft
    - Prompt for action
      - Run inspect(), use(), or move() accordingly, or quit
    - Check if key or whiskey should be added to inventory vector
      - If so, add them and increase respective count
- Void Move(Train*)
  - Call buildSpaces()
  - Prompt user for where to move, out of available spaces
  - Accept input
  - Call train->moveChar()
- Void buildSpaces(Train*)
  - spaces = train->getSpaces()
- void CheckInv(Train*)
  - print inventory
- use(Train*)
  - Prompt for what to use (from checkInv()
  - Store in item
  - If item is key or whiskey
    - Call train->use(item)
- Void inspect(Train*)
  - Prompt for what to inspect
  - Store in inspection
  - Call train->inspect(inspection)

Train Class:

Protected data members: Struct TrainNode (like Queue, but also has pointers updwn, spec1, and spec2; nodeVals are Car*, constructor takes car* and sets to nodeCar (nodeVal)), bool ending, TrainNode ptrs charCar, build, front, back

Public member funcitons:

- Train(Ten X Car*)
  - Initialize ending to false
  - Build each spaces pointers, using build
- Car* getCar(TrainNode* temp)
  - Initialize Car* thisCar as temp->nodeCar
  - Return thisCar
- Vector<string> getSpaces()
  - Declare vector of strings surrounding, with size = 5
  - Int count = 0
  - If charCar->next !NULL
    - If(if current car's name != "passenger car 3" OR current car's getExtra() is true
      - Surrounding[count] = current car's name
      - Count++
  - If charCar->prev !NULL
      - Surrounding[count] = current car's name
      - Count++
  - If charCar->updwn !NULL
      - Surrounding[count] = current car's name
      - Count++
  - If charCar->spec1 !NULL
    - If current car's name != "engine room" OR current car's getExtra() is true
      - Surrounding[count] = current car's name
      - Count++
  - If charCar->spec2 !NULL
      - Surrounding[count] = current car's name
      - Count++
  - Return surrounding
- Void moveChar(string move)
  - If charCar->next !NULL AND move == next's name
    - charCar = charCar->next
    - getCar(charCar)->enter()
  - else if charCar->prev !NULL AND move == prev's name
    - charCar = charCar->prev
    - getCar(charCar)->enter()
  - else if charCar->updwn !NULL AND move == updwn's name
    - charCar = charCar->updwn
    - getCar(charCar)->enter()

    - else if charCar->spec1 !NULL AND move == spec1's name
        - charCar = charCar->spec1
        - getCar(charCar)->enter()
    - else if charCar->spec2 !NULL AND move == spec2's name
        - charCar = charCar->spec2
        - getCar(charCar)->enter()
- void use(string use)
    - getCar(charCar)->use(use)
- void inspect(string inspect)
    - string inv = getCar(charCar)->inspect(inspect)
    - if(inv == "key")
        - getCar(back->prev)->changeExtra()
    - if(inv == "whiskey")
        - getCar(front)->changeExtra()
    - if(inv == "end")
        - ending = true
- bool getEngine()
    - return getCar(front)->getExtra()
- bool getBtw()
    - return getCar(back->prev)->getExtra()
- bool getEnd()
    - return ending

<u>Car Base class:</u>
Protected data members: int checkCount, string entrance, string newEntrancen string carName, bool extraCondition, string inspection, string item

Public member functions:

- Car()
    - Set extraCondition to false
- String getName() return carName
- Void enter()
    - If(!newEntrance.empty())
        - Print new entrance
    - Else
        - Print entrance
- Bool getExtra()
    - Return extraCondition
- Void changeExtra()
    - If !extraCondition
        - extraCondition = true
    - else
        - extraCondition = false
- virual string inspect(string) = 0

- virtual void use(string) = 0

Derived classes of Car: Engine, compartment, passenger, bathroom, dining, btwcars, baggage, top
Public member functions:

- Constructor(string name)
    - carName = name
    - set entrance string
- virtual string inspect(string inspection)
    - string inv
    - provide responses to certain strings
    - in dining and compartment if right thing is inspected AND checkCount == 0, inv = “key” or “whiskey”
        - checkCount++
    - return inv
- virtual void use(string item)
    - provide responses where appropriate

Results and reflection:

Since there were so many classes and objects to keep track of, this program was a beast to get compiling. However, all changes I had to make were purely syntax (such as a * here or there), and once I got those, my design held up. To test, I just played the game and made sure to explore everything, checking for things that were appearing when they shouldn’t. My design was pretty good though, so it didn’t really happen.