Andrew Elliott

CS162 Final Project Write-up

## Discovering Requirements

For the final project of CS162 fall term, we were tasked to create a text based game, where a user navigates through a map of linked rooms or compartments to achieve some sort of game goal. We were given a large amount of freedom on how to implement this, but I decided to use one of the themes suggested by a member in the class, the train theme. Obviously, the purpose of a final project is to integrate all of the topics of the term into one program, but here are some important class themes I focused on during this project:

- Implementation of inheritance
  - Base class formation
  - Derived class formation
  - Encapsulation
- Polymorphism
  - Use of virtual functions
  - Late binding
  - Base class pointers
- Linked data structures & iterators
  - Use of pointers to connect compartment classes
  - Iterator class to dial through the connected classes

After rereading the assignment multiple times, glancing through Piazza, and discussing with classmates, I made the following assumptions on discrete program requirements that my program will meet:

- Text based game that a player moves through
  - Set game map linked by pointers
- Structure will be linear
- Minimum 10 rooms
- Room ‘super class’ – aka, an abstract base class. Mine will be called Car
- 3 derived classes for rooms that have unique characteristics. They will be:
  - Unique items
  - Unique smells
- Include a data member to indicate which room the player is in
- Include a backpack to hold the collected items.
  - This bag must have a size limit
- Include an answer key for the grader

## Designing a possible solution

My first step in my design was to determine the actual theme using the requirements I listed above. Here's the answers to those requirements:

- I will use the train theme, with an engine (locomotive) car, a diner car, and 8 passenger cars, derived from a Car superclass consisting of virtual functions
- Each car will have a unique scent that helps hints the player as to the possible important items in the car. Each car will have a multitude of unique items to collect
- The player has a backpack with a set size of 3. Given this is a race against time, the player may only pick up a total of 3 items, so he or she must be certain she is picking up the correct items or else they will lose. This backpack will be a string array.
- The train conductor is in need of medical attention because he ate a bag of peanuts (he has a peanut allergy). He also says it would help him feel better if you retrieved him a frozen burrito and a cool, refreshing Diet Coke. The user must fetch the conductor some Antihistamine, a frozen burrito, and a Diet Coke.

Next, I needed to figure out how to design the base and derived classes by determine their possible attributes. Here's how I designed the base class, "Car":

| Car Class – Base |
|---|
| Protected:<br>• String carName<br>• String items<br>• String smell |
| Public:<br>• Void set name<br>• Void set items<br>• Void set smell<br>• Virtual get name<br>• Virtual get items<br>• Virtual get smell<br>• Pointer to next car<br>• Pointer to prior car |

Using past knowledge of abstract base classes, I knew that accessor functions had to be virtual functions. Using polymorphic classes and functions helped make my game loop very easy to code as I only needed to use the iterator class, which points to a base class pointer.

Because I'm choosing to hard-code the values for each instantiated passenger car object, the design for the derived classes is very brief:

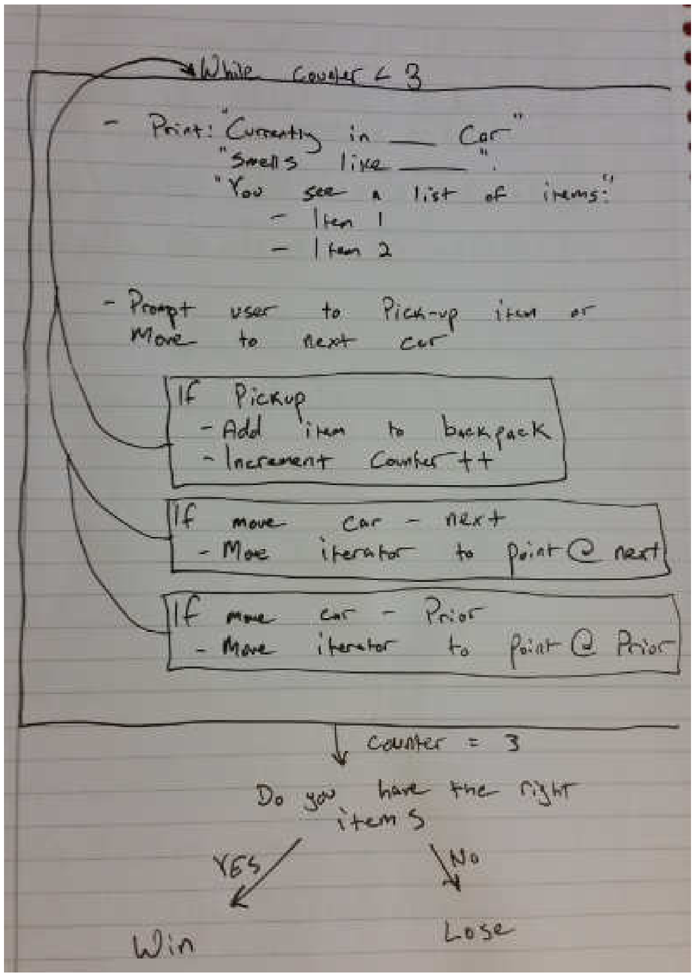
| Derived Class : Car |
|---|
| Public:<br>• String get car name<br>• String get items<br>• String get smell |

Finally, I need an iterator class to dial through the list of linked objects to avoid a huge chunk of if/else or switch statements. I will be using this iterator in the main program loop to pull objects and run through the different compartments.

| Car Iterator class |
|---|
| Public:<br>• Car class pointer object<br>• Void next car<br>• Void prior car |

The "Car class pointer object" allows this iterator to point to the base class, and allows polymorphic functionality.

The next step in my design was to develop a process flow of the actual game loop. I started with a drawing I sketched at work:



While counter < 3
- Print: "Currently in ___ Car"
"Smells like ___".
"You see a list of items:"
- Item 1
- Item 2
- Prompt user to Pick-up item or Move to next car
If Pickup
- Add item to backpack
- Increment Counter ++
If move car - next
- Move iterator to point @ next
If move car - Prior
- Move iterator to point @ Prior
Counter = 3
Do you have the right items
YES
No
Win
Lose

I used this to start my pseudocode to design how my main function would interact with the classes.

```
instantiate derived classes
instantiate iterator pointer - 'iter'
```

```
set names, items, and smells of the car classes
instantiate the backpack array to hold items

Manually link car classes
   First car prior car pointer is null
   first car next car is car 2
   car 2 prior car pointer is first car
   ...
   final car next car pointer is null.

Print welcome to the train escape with opening sequence
Print you need to bring the conductor antihistamine, a frozen burrito, and a diet coke
print your backpack only holds 3 items - choose the 3 items wisely

pack counter = 0
while pack counter is less than 3

   currently in iter->carname
   smells like iter->get carsmell
   you see iter->get items

   print:
   use num pad to pick up an item (case 1 - 4)
   move to prior car (case 5)
   move to next car (case 6)
   see the answer key  (case 7)
   quit (case 8)

   receive user input

   switch statement (user input)
      case 1 - 4
         backpack at current counter index = iter->get item
         pack counter++
         break
      case 5
         if in the front car
            Cant do that, youll die
         set car pointer to prior car address
         break
      case 6
         if in back car
            cant do that, youll die
         set car pointer to next car address
         break
      case 7
         show answer key
         break
      case 8
         return 0
      default
         invalid entry

Print your back pack is full
bool variables to determine if 3 items are winning items
for loop
   print out contents of backpack
   check if backpack at i = burrito, antihistamine, or Diet Coke
      set win variable to true

if all win variables == true
```

```
   winner
else
   loser
```

The most difficult part to implement was the use of the iterator. I knew that using it would save me hours of hard coding in the end and help prevent spaghetti code, but it took me a couple hours to get it working. I'm glad I did, because its quite sleek.

Designing the remainder of the game loop was easy, especially since I used polymorphism. I feel like I finally have a thorough understanding of how it works.

## Testing and Debugging

| **Test Case** | **Input** | **Expected Outcome** | **Did I pass?** |
|---|---|---|---|
| File start | None | File opens, prints out the intro bar and the story context. Prompts user to press 1 to continue. | Yes |
| Error catching | 2 | While loop prohibits any number other than 1 to be entered | Yes |
| Initial game menu start | 1 | Program prints<br>• Items you need to find<br>• Directions on what to do<br>• Backpack only holds 3 items<br>• Currently in engine car 1 of 10, with items, current smell, and a menu with commands | Yes |
| Linked pointer/iterator operation & Abstract class pointer operation | 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, then an eleventh 6. | Program moves through all the cars going towards the back and prints (10 times):<br>• Car name (x of 10). Car name (x+1 of 10), etc<br>• Car smell<br>• 4 items in the car<br><br>On eleventh car (passenger car 8), prints cant do that, null pointer, youll die. Then prints menu options for passenger car 8 | Yes, see picture 1 |
| | 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, then an eleventh 5. | Program moves through all the cars going towards the front and prints (10 times):<br>• Car name (x of 10). Car name (x-1 of 10), etc<br>• Car smell<br>• 4 items in the car<br><br>On engine car, prints cant do that, null pointer, youll die. Then prints menu options for engine car | Yes, see picture 2 |
| Item pick up, array operation, game loop | 1 – 4 in any car (incorrect items), for three items | Program prints what items you've picked up, then reprints current car menu options. After picking up three items, game loop breaks, program prints<br>• your pack is now full | Yes, see picture 3 |

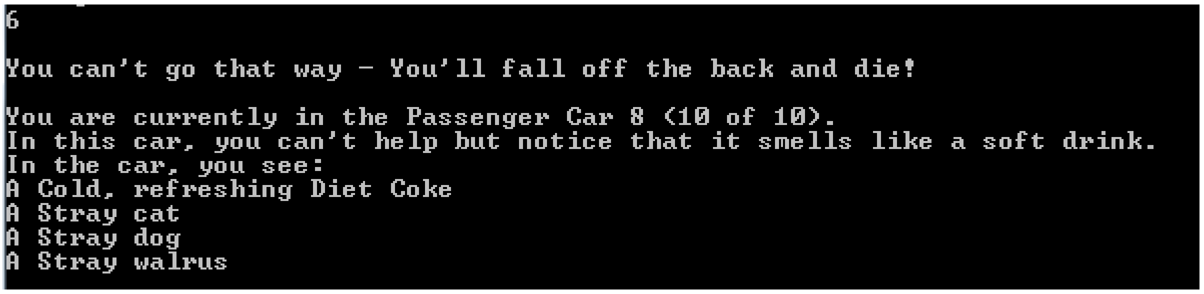
| | | | |
|---|---|---|---|
| operation, win checking | Car 4 of 10, item 2<br>Car 8 of 10, item 4<br>Car 2 of 10, Item 2 | • you return to engine car with pack containing items<br>  o Pair of mismatched socks<br>  o Random dictator<br>  o Frozen burrito<br><br>Print you lose, didn't bring back the correct items | |
| | Correct items, in any order<br><br>Car 2 of 10, item 2<br>Car 6 of 10, item 1<br>Car 10 of 10, item 1 | Program prints what items you've picked up, then reprints current car menu options. After picking up three items, game loop breaks, program prints<br>• your pack is now full<br>• you return to engine car with pack containing items<br>  o Burrito<br>  o Antihistamine<br>  o Cool, refreshing diet coke<br><br>Print you win | Yes, see picture 4 |

## Reflection

In reflecting on this class' final project, I'm glad I spent the time I did designing the flow diagram and taking the time to correctly implement an iterator. I can already see how using this tool will decrease the size of my previous assignments, and how I can implement this to start my own game projects.

I need to spend time over the winter break working on linked data structures and containers. Unfortunately, I didn't see I need to implement a container, and given that we received less than 20 minutes of lectures, zero of which included concrete examples, I decided it wasn't a reasonable expectation that it should be included in the project. That being said, if I choose to scale this project into a much larger game, a container would be very useful in holding picked up objects, and manipulating them according to how the user wishes to play the game.

**Picture 1**

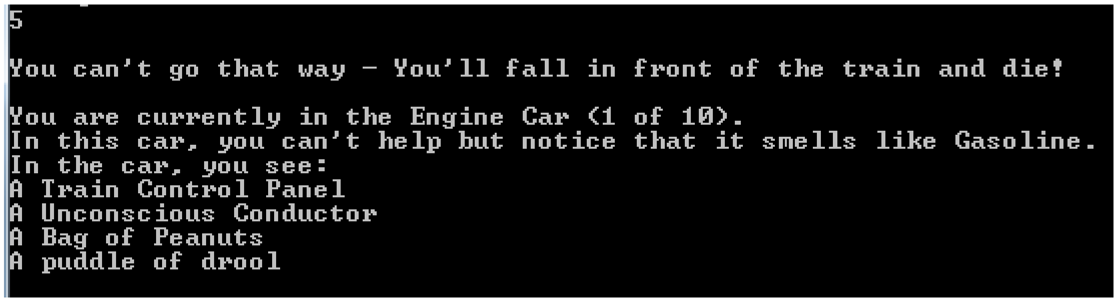


```
6

You can't go that way - You'll fall off the back and die!

You are currently in the Passenger Car 8 (10 of 10).
In this car, you can't help but notice that it smells like a soft drink.
In the car, you see:
A Cold, refreshing Diet Coke
A Stray cat
A Stray dog
A Stray walrus
```

**Picture 2**

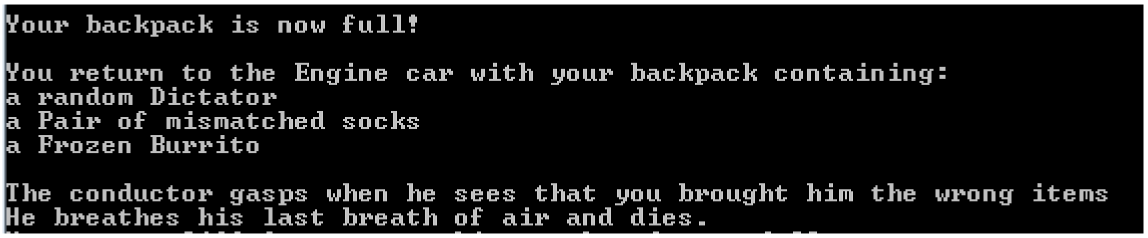


```
5

You can't go that way - You'll fall in front of the train and die!

You are currently in the Engine Car (1 of 10).
In this car, you can't help but notice that it smells like Gasoline.
In the car, you see:
A Train Control Panel
A Unconscious Conductor
A Bag of Peanuts
A puddle of drool
```

**Picture 3**

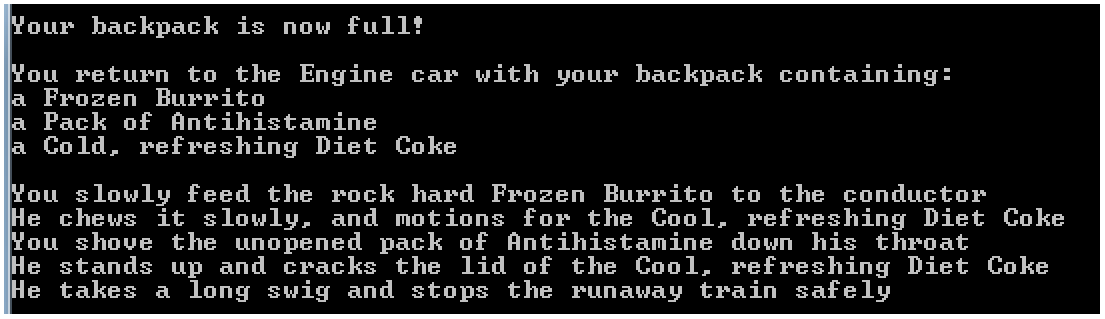


```
Your backpack is now full!

You return to the Engine car with your backpack containing:
a random Dictator
a Pair of mismatched socks
a Frozen Burrito

The conductor gasps when he sees that you brought him the wrong items
He breathes his last breath of air and dies.
```

**Picture 4**



```
Your backpack is now full!

You return to the Engine car with your backpack containing:
a Frozen Burrito
a Pack of Antihistamine
a Cold, refreshing Diet Coke

You slowly feed the rock hard Frozen Burrito to the conductor
He chews it slowly, and motions for the Cool, refreshing Diet Coke
You shove the unopened pack of Antihistamine down his throat
He stands up and cracks the lid of the Cool, refreshing Diet Coke
He takes a long swig and stops the runaway train safely
```