This lab helped us learn the concepts of inheritance, polymorphism, and a continuation of exception handling. We used inheritance to derive the Student class from the Item class and the Stack and Queue class from the Student class. This means that the Stack and Queue class can access the Item's class based on whatever accessor type we give it. Polymorphism was used with our Add Item function which chose between the Stack or Queue AddItem() based on whatever the user chose at the beginning of the program. We also added a few "try catch" blocks to this program to cover any exceptions. These concepts are important for a career in engineering because inheritance is one of foundation concepts to all large code bases. There is a massive amount of code and efficiency is a big thing to think about all the time so having a good inheritance and using polymorphism can make your program easier to use and edit.

## Task 2:

```
C:\Users\Zack Johnson\source\repos\slitersr\EEC
                                  ~~~MENU~~~
Select Option to Start Program -
                                  1. AddItem
1. Set max size
                                  GetItem
2. Set as default (20)
Enter: 1
                                  IsEmpty
Enter max size: 5
                                  4. IsFull
                                  Size
~~~MENU~~~
                                  6. ClearList

    AddItem

GetItem
                                  7. Exit
IsEmpty
4. IsFull
5. Size
                                  Enter: 2
6. ClearList
7. Exit
                                  mNumber: 2
Enter: 1
                                  First Name: 2
- Create student parameters -
Enter M number (integers only): 1
                                  Last Name: 2
Enter First Name: 1
                                  Birthday 2
Enter Last Name: 1
                                  GPA: 2
Enter Birthday: 1
Enter GPA: 1
                                  Student Removed.
Student added.
```

Set Max Size and Add Item

GetItem

## ~~~MENU~~~

- 1. AddItem
- 2. GetItem
- IsEmpty
- 4. IsFull
- 5. Size
- 6. ClearList
- 7. Exit

Enter: 3

The list is empty

~~~MENU~~~

- 1. AddItem
- 2. GetItem
- 3. IsEmpty
- 4. IsFull
- 5. Size
- 6. ClearList
- 7. Exit

Enter: 4

The list is not full

**IsEmpty** 

IsFull

## ~~~MENU~~~

- 1. AddItem
- 2. GetItem
- IsEmpty
- 4. IsFull
- 5. Size
- 6. ClearList
- 7. Exit

Enter: 5

Size of List: 1

ClearList

## Task 3:

```
Select Option to Start Program -
1. Set max size
2. Set as default (20)
Enter: 1
Enter max size: 5
                                   ~~~MENU~~~

    AddItem

 Select Type of List -
                                    GetItem
1. Stack
                                    IsEmpty
2. Queue
                                    4. IsFull
Enter: 1
                                    5. Size
                                    6. ClearList
Creating a Stack...
                                    7. Exit
~~~MENU~~~
                                    Enter: 1

    AddItem

                                    - Create student parameters -
2. GetItem
                                    Enter M number (integers only): 1
IsEmpty
                                    Enter First Name: 1
4. IsFull
                                    Enter Last Name: 1
5. Size
                                    Enter Birthday: 1
6. ClearList
                                    Enter GPA: 1
7. Exit
                                    Student Added.
Enter:
```

**Set Max Size and Create Stack/Queue** 

AddItem

```
~~~MENU~~~
 1. AddItem
 2. GetItem
IsEmpty
4. IsFull
5. Size
6. ClearList
 7. Exit
 Enter: 2
mNumber: 1
 First Name: 1
 Last Name: 1
Birthday 1
GPA: 1
 Student Removed.
GetItem
~~~MENU~~~
```

 AddItem 2. GetItem IsEmpty 4. IsFull 5. Size 6. ClearList 7. Exit Enter: 3 The list is empty

~~~MENU~~~

**IsEmpty** 

```
~~~MENU~~~
1. AddItem
                           1. AddItem
                           2. GetItem
2. GetItem
IsEmpty
                           IsEmpty
4. IsFull
                           4. IsFull
5. Size
                           5. Size
6. ClearList
                           6. ClearList
7. Exit
                           7. Exit
Enter: 4
                           Enter: 5
The list is not full
                           Size of List: 2
```

IsFull Size

```
Enter: 6

List has been cleared.

~~~MENU~~~
1. AddItem
2. GetItem
3. IsEmpty
4. IsFull
5. Size
6. ClearList
7. Exit

Enter: 5
```

ClearList

Size of List: 0

```
Student Added.

~~MENU~~~

1. AddItem
2. GetItem
3. IsEmpty
4. IsFull
5. Size
6. ClearList
7. Exit

Enter: 3

The list is not empty
```

isEmpty with data in the Stack/Queue

```
1. AddItem
2. GetItem
3. IsEmpty
4. IsFull
5. Size
6. ClearList
7. Exit
Enter: 4
The list is full
```

IsFull with data in the Stack/Queue