CS3100 - 001

Professor Koh

12/18/2020

Gerardo Alcazar,

Ethan Machado,

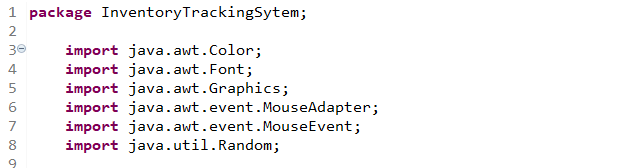
Jorge Sanchez

CS 3100 Final Report

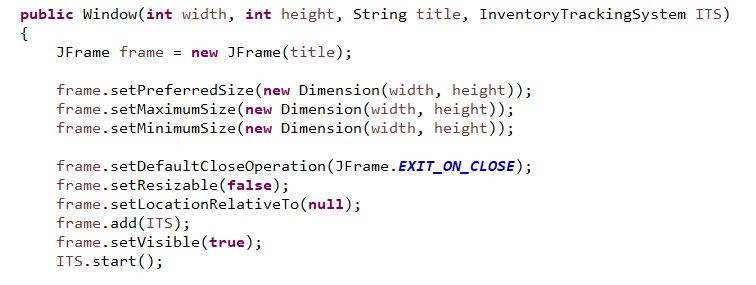
Introduction: Our project is based upon an inventory tracking program that allows the user to scan, translate, store, and track inventory. The program also allows the user to manually correct/alter product availability if desired. Our system first scans an item into inventory, then stores the item into its holding slot, and allows for the tracking of the item to check its availability and/or pull it out of inventory. We have added additional functions and expansions in our final project such as giving each item its information report and enlarging the program’s holding capacity to 16 slots. Our goal for the final project was to expand on our mini-project by making it more complete and robust. This system is supposed to mimic the functions of larger-scale inventory systems that corporations such as Amazon, Walmart, and Target use in a much smaller package. We chose an inventory tracking program for this project because of the inspiration we drew from our occupations and the systems that they used to store and manage their product. Creating an inventory system like the ones our companies use was an idea that we collectively supported because we believed that creating such a system would provide us with valuable coding practice and better knowledge of how these invaluable systems work.

Background/Related studies: We were inspired to construct an inventory tracking system because of our background knowledge of our employers’ inventory programs. We notice these systems constantly working in the background by the way that they interact with physical inventory and how they make our jobs much more efficient and simple. We thought it would be interesting and fulfilling to try to make a smaller system that could do the same work and help us to better understand what these commercial systems invisibly accomplish at our jobs. Additionally, we wanted to construct a system using a data structure that was mentioned in one of the modules of the course text. We chose a linked list for reasons that will be named later. Our mini-project was further motivation to keep building a larger inventory tracking system for our final project because we wanted to create a more feature-rich version.

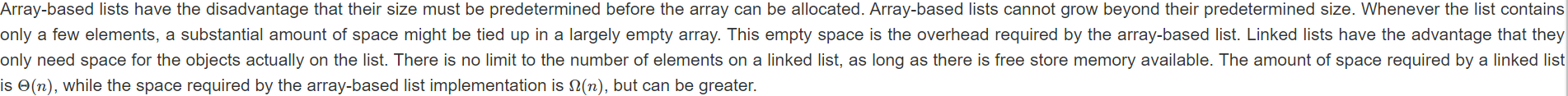
Methods: Our project was implemented by combining several different classes using the Java programming language. We drew inspiration from an open-source Snake-like game called Wayz 2 that can be written in Java that uses the JFrame interface, velocity, frames per second, parameters, and many Java visual interface classes to create its graphical interface. These classes that are shown below were used in our program to create a similar graphical interface:

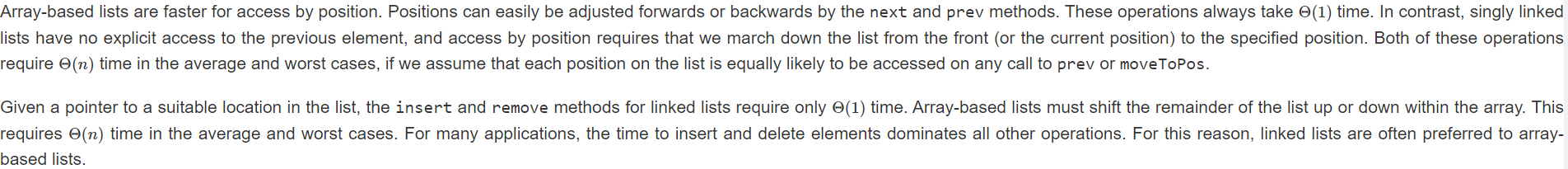


The classes listed above allowed us to pick colors, fonts, and render graphics for our heads up display (HUD) menu. The mouse classes track mouse movements, clicks, and other functions that are used to interact with our inventory system. Here is our JFrame method that creates our inventory system HUD menu:



Our user interface was designed to have a fixed frame size that inserts new inventory in its unique box in a perfectly centered manner. Additionally, we designed an algorithm that utilizes a linked list to add and remove objects because we were unsure of the number of items that will be stored in our inventory system at any given time. The linked list data structure provides our inventory tracking system with the size flexibility that it needs while also being efficient in both space and time performance. A linked list was also the preferred choice because it has constant time insertion and removal methods that our program constantly uses. An array-based implementation would have to slowly shift all of its elements to add or remove an element. In the screenshots of our textbook below, it verifies that the space requirement of a linked list is Θ(n). The cost for position accessing is Θ(n) and inserting and removing are Θ(1).

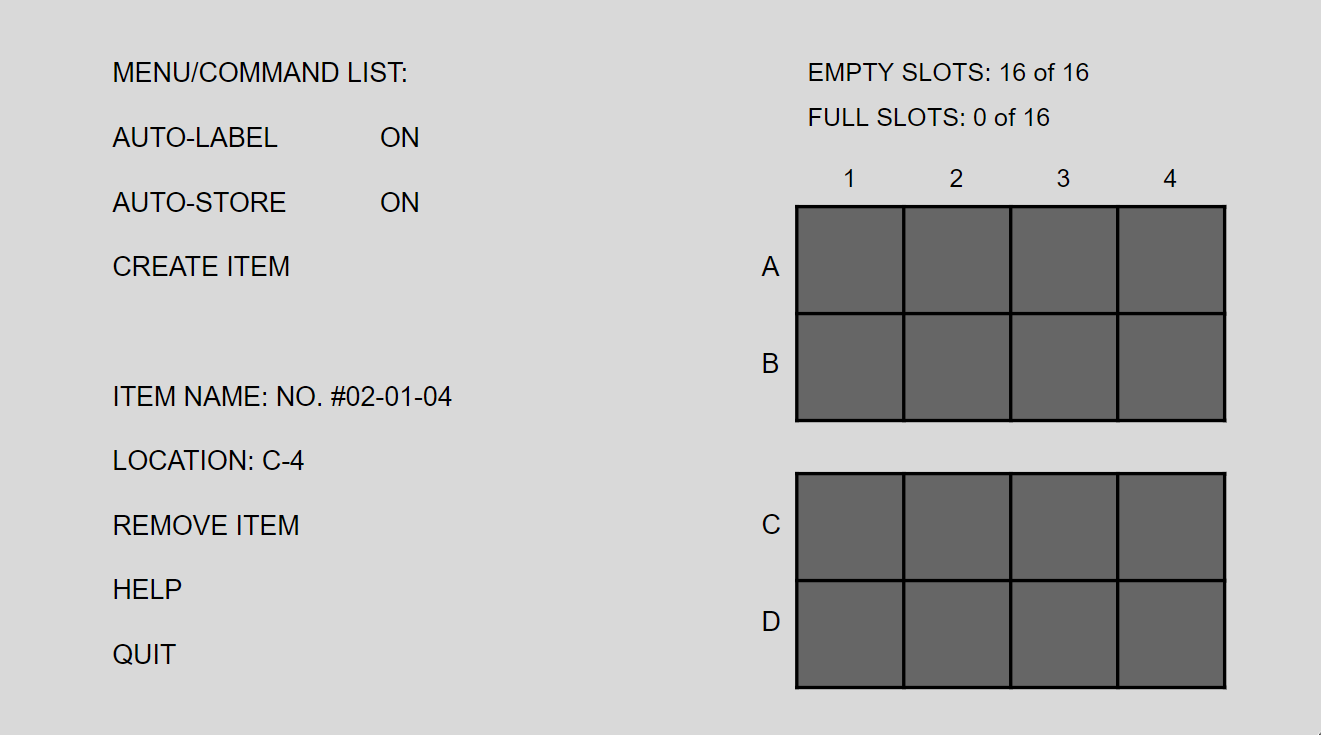


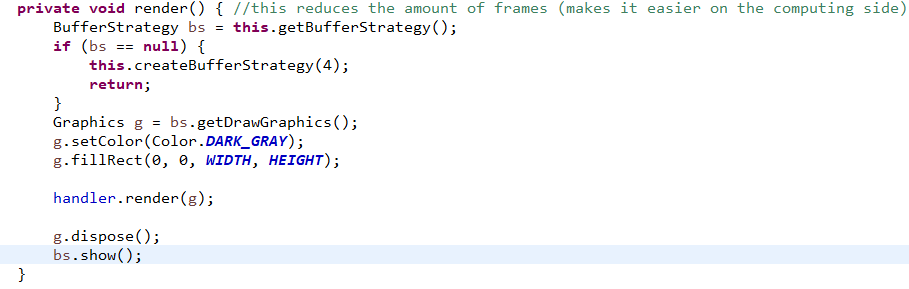


A buffer strategy is implemented as well to reduce the frames per second of the program and to make it easier on the user’s computer to run our inventory system. An additional method is used in our program that counts fps to let the programmer know that the program’s visual effects and animations are being implemented correctly.

We used YouTube to find the Wayz 2 game program that we drew our inspiration from. It showed us how we could implement a HUD for our inventory program that utilizes fps and other game-like animations. Additionally, our course textbook was used to figure out which data structure to use based on our use cases for the program and taught us how to implement the Java atw classes previously mentioned. GeeksforGeeks was also used for general tips and instructions on coding issues that we ran into.

Implementation: For our inventory tracking system, we chose to create a JFrame interface HUD with five main functions for the mini-project. The first is a method that creates items to be put into inventory. The second one being an auto-store function that places new inventory in a single slot from left to right. The third is an auto-label method that gives an inventory item its unique identifier. The fourth is a remove function that allows the user to click on an item and remove it from inventory. The remaining method is a help menu that informs the user how to use the program. The system also has a quit button that terminates the program. For the final project, we have added information reports to every stored inventory item, increased the holding capacity of our system, and added an animation that tells the user when the system is full. Information reports of inventory items consist of a name/number and its storage location. The mini-project version of our program had only eight available inventory slots on the HUD compared to the 16 present in our final project. Below is a template of the HUD of our expanded inventory system that shows all of the methods previously mentioned, contains 16 slots, and the status of each slot:



Discussion: Our inventory tracking system is complex in that it uses multiple classes and has many methods using a large number of parameters, so it took a lot of patience, testing, and problem solving to find solutions. One problem, in particular, was not implementing the buffer strategy because, without it, the program runs with an fps of over 21 million that puts severe stress on the user’s computer and renders the program useless. When correctly implemented, the buffer strategy reduces the fps down to a much more manageable 200 to 300 fps. 2D games like Wayz 2 use buffer strategies to reduce their fps to a playable number similar to how we implemented it into our system. Here is the buffer strategy: Another problem that we faced was making sure that the scope of the program was correct. For example, the menu window (HUD) has to be loaded in before the class that contains all of the systems’ functions. If the program was not organized correctly, it would not run properly and would present an error. Parameters also needed to be written correctly, or else the program would throw an error when trying to execute. After creating this program, we would like to explain to the class how important it is to use all of the coding resources that one has available. For example, we had to use GeeksforGeeks quite often to make sure that the code we were implementing was written correctly. Additionally, we would inform the class of the importance of knowing how to create and use parameters correctly to ensure that all their references and functions work properly. Lastly, our group would like to stress the importance of using correct syntax because even the smallest single-digit spelling/syntax mistake can cause a program to fail such as calling an incorrectly spelled class.

Communication was also an obstacle for this program because of the missing face to face contact. Everyone in the group has had their different responsibilities throughout the semester, so it was difficult arranging meetings that everyone could attend. We overcame these problems by using different forms of digital mediums such as Snapchat, Discord, Zoom, and email. Email and Snapchat allowed us to communicate with one another on a convenience basis, while Discord and Zoom were the applications that we used for live meetings. These discussion techniques helped us to collaborate virtually and also taught us alternative ways of communicating that we might not have used if we were taking in-person classes. The initial communication barrier that we faced ended up being a learning experience for our group.

Conclusion: We learned several lessons throughout the creation of this inventory system project such as the importance of repeatedly testing our program to make sure that all variables printed out stored values correctly and how to use multiple classes and parameters effectively. Building a larger program with multiple classes has been great for practicing and learning more about code and the programming process. This learning did not come without its difficulties, however, as we had to dedicate a lot of time to finding and fixing errors. Our problem solving abilities were also sharpened, as we had to figure out solutions to many problems. A project like this was an important experience because it better simulated what programming is like professionally where most systems are designed using teamwork and communication. The online-only aspect of this project made things more strenuous, but it taught us how to collaborate on a project virtually. The experience gained from this project has helped us become more advanced in our programming knowledge and strategy which are both very important skills to build upon to do well professionally. Most of all, we believe that the communication and group working skills that we gained from this team-based project will help us the most in the long run because of the professional world’s emphasis on group development.

References: YouTube, GeeksforGeeks, and the CS3100 course textbook.