OPERATING SYSTEM

A coursework completed as part of the requirement for

SUBJECT NAME: **OPERATING SYSTEM**
SUBJECT CODE: **CC216**
LECTURER: **DR. KYAW KYAW HTIKE**

Entitled

ASSIGNMENT TITLE: **ASSIGNMENT 1**

Submitted on

DATE OF SUBMISSION: **7th AUGUST 2017**

Produced by

Name	Student ID	Course
IVON LAI JIA YI	**1001437696**	**BSc (Hons) Computing**
JOHN LOH ERN-RONG	**1001439193**	**BSc (Hons) Computing**
LOH JUN KEAN	**1001231369**	**BSc (Hons) Business Information Systems**
LOUIS JULIENDO	**1001540916**	**BSc (Hons) Computing**
REGINE LIM	**1001541495**	**BSc (Hons) Computing**
TOH KAI SIN	**1001437397**	**BSc (Hons) Computing**

Introduction

In this present age, technology is widely used in the world today, from homes, educational institutions to government sectors and big organizations. This has resulted to the frequent usage of computers and has already become a norm in our daily lives as users. Computers are used to accept information that is able to store and process many kinds of data. With the evolution of computer programs, computers nowadays can perform many programs at a time. However, this causes many processes to be running on a single system and these processes can go up thousands. Therefore, it is important to have proper management and organisation of these running processes.

The existence of Operating Systems, which is the broker between hardware and software, that manages both hardware and software resources, where one of the most important functions that it provides is Disk Scheduling. In the aspect of Disk Scheduling, the concept of seek time is crucial on Operating Systems. An increased seek time causes the system to slow down. This is due to the linking of disk requests into queues. Therefore, algorithms are needed to be performed to manage these processes. For example, Disk Scheduling Algorithms are used to reduce the total seek time of a disk request. This algorithm is able to service the disk I/O requests and fully maximize the use of hardware efficiently. Through this algorithm, it provides fast access time and increase the disk bandwidth where total number of bytes can be transferred more.

In our project, we have included three different types of Disk Scheduling Algorithm which are C-SCAN, LOOK, and C-LOOK. These three algorithms provide different implementations and also offer different advantages and disadvantages. The explanation of each algorithm will be shown below:

C-SCAN

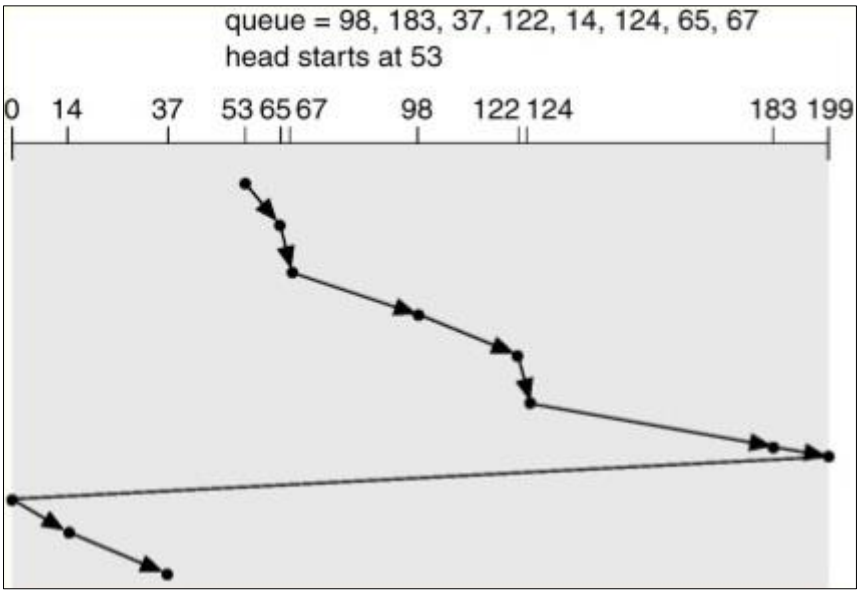


Diagram 1

In C-SCAN, the disk head starts its scanning from one end of the disk towards the nearest end, servicing disk requests along its movement and goes all the way to the end of the system. At the point when the disk arm reaches the other end of the disk, the disk arm returns to the beginning of the disk (or the end, depending on the starting of the disk head).

Based on Diagram 1, the disk head starts at request 53. The nearest request from request 53 is request 65. According to the arrows based on the nearest requests, it goes on to request 67, request 98, request 122, request 124 and request 183. At this point, the disk arm will move along to the nearest end of the disk, which is request 199. After that, it returns to the beginning to the disk (request 0), and continue servicing requests based on the nearest request queue, which is request 14 and then lastly, request 37.

C-LOOK

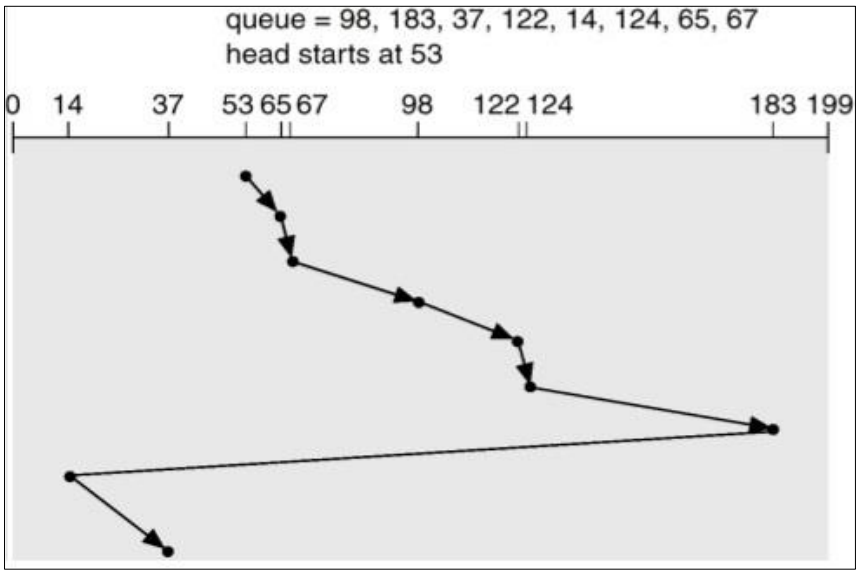


Diagram 2

C-LOOK is also known as the enhanced version of C-SCAN. In C-LOOK, the disk head will only go all the way to the last and final request on each direction. It does not go past the final request along its direction, it also does not go towards the nearest end of the disk (like in C-SCAN). After that, it goes all the way back to the opposite direction, without going all the way to the beginning of the disk.

Based on Diagram 2, the disk head starts at request 53. The nearest request from request 53 is request 65. According to the arrows based on the nearest requests, it goes on to request 67, request 98, request 122, request 124 and request 183. At this point, the disk arm will not move along to the nearest end of the disk (sector 199), instead it returns all the way in its opposite direction to the next furthest request, which is request 14 and then lastly, request 37.

LOOK

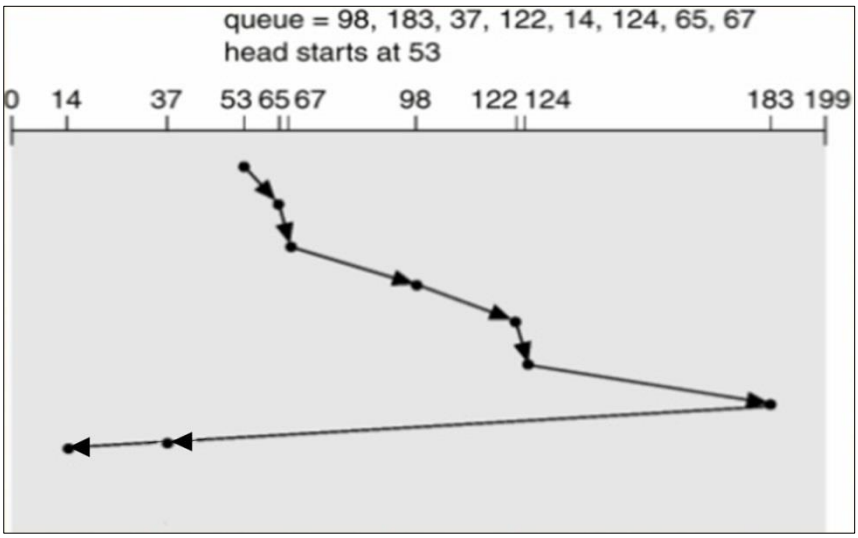


Diagram 3

In LOOK, the disk head starts its scanning from one end of the disk towards the other end. The servicing of disk requests goes along its movement and all the way to the end of the system where the disk arm movement moves to the opposite direction and the servicing of disk requests continues. Unlike other similar disk scheduling algorithms, known as SCAN (which is not included in this project), the disk arm stops moving inwards (or outwards) when there is no more disk requests to be serviced exist in that particular direction.

Based on Diagram 3, the disk head starts at request 53. The nearest request from request 53 is request 65. According to the arrows based on the nearest requests, it goes on to request 67, request 98, request 122, request 124 and request 183. At this point, the disk arm will not move along to the nearest end of the disk (sector 199), instead it returns all the way in its opposite direction to the next nearest request, which is request 37 and then lastly, request 14.

Background

The system, Disk Scheduling Algorithm System (DSAS), is developed for implementing disk scheduling algorithms. In DSAS, the three disk scheduling algorithm that is being chosen to be implemented include:

a. C-SCAN
b. LOOK, and
c. C-LOOK.

DSAS is written in Java programming language. The aim of this system is to help users to calculate the total disk arm movement according to the disk scheduling algorithm selected, and draw a graph according to the disk requests provided by the user. Some of the functions that are included in this program include:

a. **Allowing the users to select the type of disk algorithm to be used.** Users are able to choose from three of the following disk algorithm, which include C-SCAN, LOOK and C-LOOK.

b. **Accepting the input for number of disk request from users.** Users can input any numeric values, where the numeric values must be greater than 0. This is to determine the number of request for the disk to be processed.

c. **Accepting the input for disk requests from users according to the number of disk request entered earlier.** Users must input numeric values which are greater than 0. The number of disk request to be input will be determined by the data input earlier by the users.

d. **Accepting the input of the current disk arm position from users.** Users must input the current disk arm position to determine the start of the disk request. It must be different from the disk requests input earlier.

e. **Accepting the selection of the direction of disk arm movement from users.** After finish inputting the earlier required information, users must select the direction of disk arm movement. Users can choose from the following two options, which are towards the beginning and towards the end.

f. **Calculating the total disk arm movement.** The program will calculate the total disk arm movement according to the disk request input by the users.

g. **Generating the graph.** The graph for the disk request will be generated. Users can have a clearer view on how the algorithm is like through the graph.

h. **Saving the graph.** Users are able to save the graph in image for future reference.

The system benefits users by giving accurate calculation of disk arm movement. This helps in increasing the accuracy for users while working with the algorithm.

Methodology

The system that is being developed, Disk Scheduling Algorithm System, is a multifunction system. It is used to take in the input of users to come out with the correct queue for disk scheduling according to the inputs by users. In our system, we have used few types of error validation, including:

i. Checking if users enter any input in the text field,
ii. Checking if users enter only numeric values in the text field,
iii. Checking if the input from users is valid for performing disk scheduling algorithm, for example, the number of disk requests should not be less than 1
iv. Checking is there any redundancy of data in disk request, which means that users are unable to enter two same values of disk request,
v. Checking is the disk request entered within the range of the number of cylinders entered, and
vi. Checking is the current arm position a unique number, which means that the current arm position must not be the same as disk request.

To use the system, firstly users will need to choose from one of the disk scheduling algorithms, which are C-SCAN, LOOK and C-LOOK. Users can choose the type by selecting the radio button, and click on "Next" button to continue with the next step. Then, users will need to enter number of disk requests and total number of cylinders in the next window before proceeds to the next step. The next window, "Disk Request" will prompt out after users enter the values in the previous window. The text field for users to input the request will be generated according to the number of disk requests entered in the previous window. For instance, if users enter "8" for number of disk requests, then eight text fields will be generated to receive inputs from users.

After the users enter the requests, users have to click "Next" button to continue with the process. Users will then need to enter the current arm position to determine the head of the queue. Then, users have to choose the direction of the arm movement direction, which is either towards the beginning or the end of the disk. The results will be shown in the next window after users click "Next" button. The results shown will consist of the scheduled queue and also the total disk arm movement of

the disk. Graph can be generated using the result. Users can click “Draw” button to generate the graph according to the input. If users wish to continue with a new disk scheduling algorithm, they can click on “New” button to continue with a new disk scheduling algorithm. Otherwise, users can click “Exit” button to exit from the system.

The codes are written in Java programming language which is being run and compiled using the NetBeans IDE. There are a total of 6 windows in our program. Overall, in the case of checking conditions, most of the codes are implemented using if-else method to check the condition. Window1, Window2, Window4 and Window5 are designed using the Graphical User Interface (GUI) builder built-in inside NetBeans IDE, also known as a JFrame form. The GUI design of Window3 on the other hand is hardcoded. Lastly, Window6 uses an imported library, known as JFreeChart, and the GUI design is also hardcoded as well.

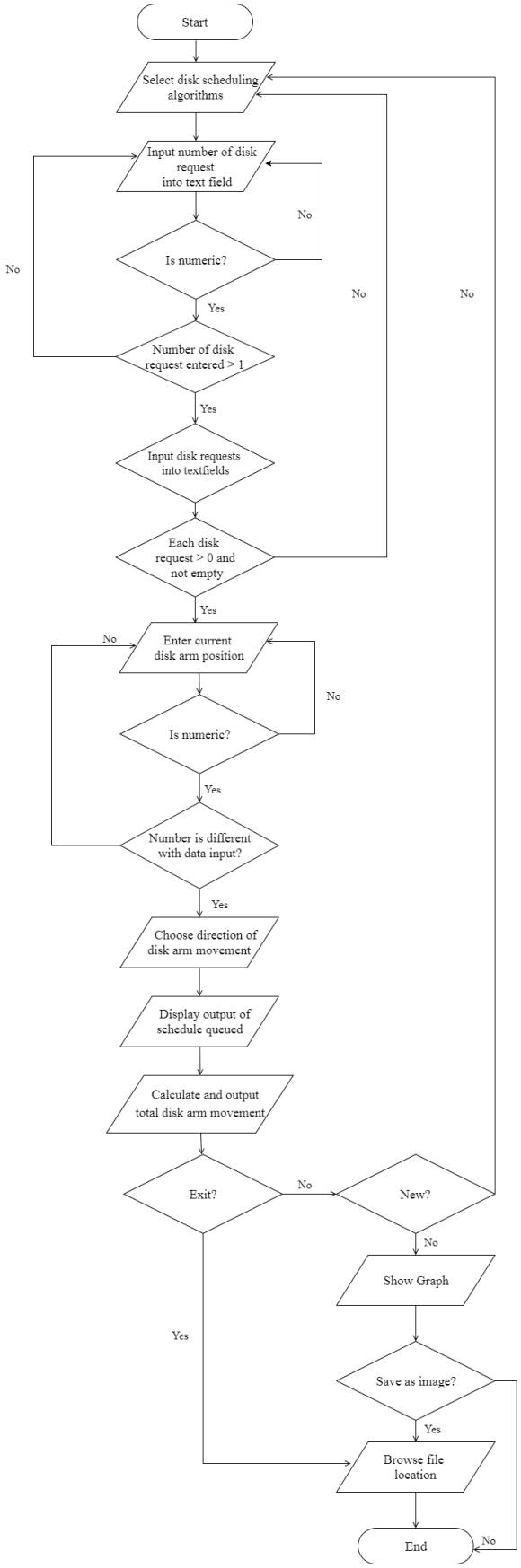
In the aspect of exception handling, the try and catch method is implemented along with the JOptionPane function to alert the user with a message to prompt the exception for the user to know. Many Boolean methods are also created, such as contains() that takes in two arguments, an array and an integer respectively. This method is to check whether an element in an array is bigger or equals to the specified integer. Next, the containsOne() is also very similar to contains(). The containsOne() method checks if an element in an array is less than or equals to a specified integer. Other than that, the checkLoop() method is used to check if any disk requests input by the user is the same or not. This method does not take in any argument. Lastly, the checkHeadnEnd() method checks if the disk head is less than or equals to 0 or if the disk head is more than the total number of cylinders.

One of the most challenging part for us at creating this program is Window3. As mentioned above, the GUI design of Window3 is hardcoded which has added much more difficulty for us. This is because JTextField needs to be created based on the user input from Window2. After that, all of the user input on the JTextFields created have to be stored in an array. To achieve this, we created an array of JTextFields, and then created another array to store all of all the input from JTextFields through a for loop, by using the getText() method.

In an overview of our program in general, all constructors in every window has a number of arguments to transfer some of the user's input window by window, such as the boolean value of CSCAN, LOOK and CLOOK that check which JRadioButton is selected at the very start of the program by the user. Moreover, to get the number of disk requests, the current arm position, and which JRadioButton is selected to check the arm movement direction, whether if it is towards the beginning, or towards the end.

In our disk scheduling algorithms codes, apart from the exception handling and the if-else methods to check for conditions, we have used one significant built-method in Java called Arrays.binarySearch() under Arrays. In our case, the Arrays.binarySearch() method is used to find the index of our disk head in the array of disk requests created before. There are two arguments in this method, the first where it takes in array (to check a particular index of a certain value), and the second one where it takes in an integer to check which value in the array from the first argument matches the integer from the second argument. Apart from Arrays.binarySearch(), we have also used Arrays.sort() method where it takes in one argument, an array, to sort all elements of integer in an array in an ascending order. In our case, this is used to sort disk requests so that we can get a scheduled queue from the left to right, which is used in our disk algorithm.

Flowchart



Flowchart showing the system flow.

Pseudocode

BEGIN

SELECT either one scheduling algorithm

LOAD Disk Information window

INPUT number of disk request

IF (input entered >1) THEN

LOAD Disk Requests window

END IF

INPUT disk requests

IF input is numeric, text field is not empty and is different with disk request entered THEN

LOAD Position and Direction window

END IF

INPUT current disk arm position

SELECT direction of disk arm movement

IF disk arm position is numeric and is different with disk request entered THEN

LOAD Result window

END IF

DISPLAY output of scheduled queue

DISPLAY total disk arm movement

```
        IF exit THEN

        END

        ELSE IF perform new algorithm THEN

                BACK to Disk Scheduling Algorithm window

        ELSE IF show graph THEN

                LOAD Graph window

                OUTPUT graph

                        IF save as image THEN

                                SELECT file location

                        END IF

        END IF

END
```

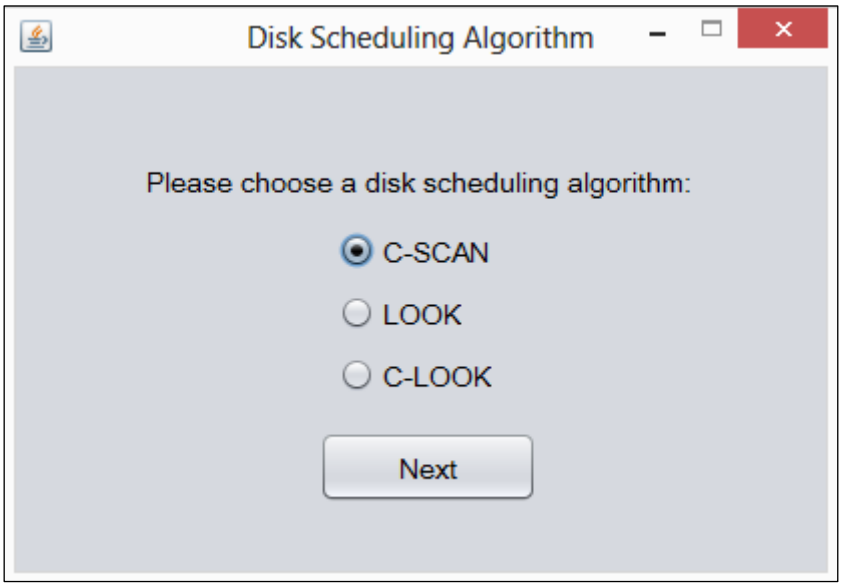
Results

The system that is being developed, named Disk Scheduling Algorithm System (DSAS) has the following features:

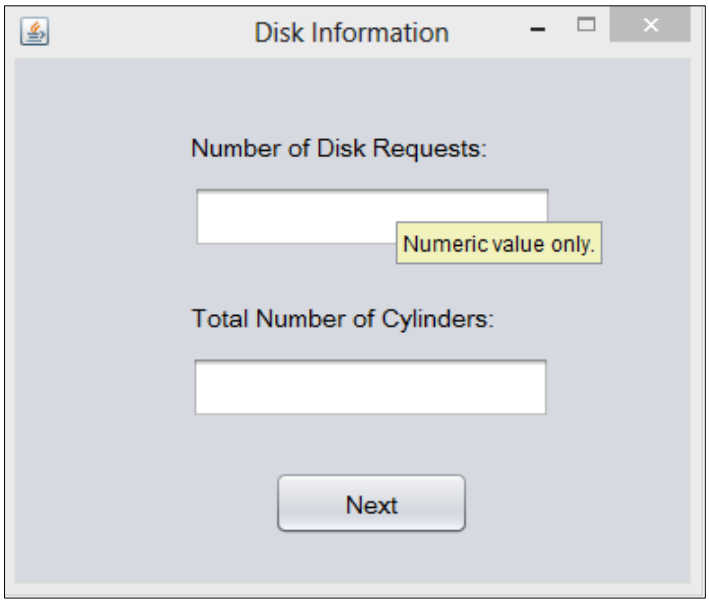
i. Allowing users to select the type of disk algorithm
ii. Accepting inputs including number of disk requests, disk requests and current disk arm position from users
iii. Allowing users to select the direction of disk arm movement
iv. Calculating total disk arm movement
v. Drawing graph based on the queue of the disk request
vi. Saving graph as image
vii. Error validation
viii. Tool tip text available as small reminder for users

Below are the screenshots of the outcome for the system.

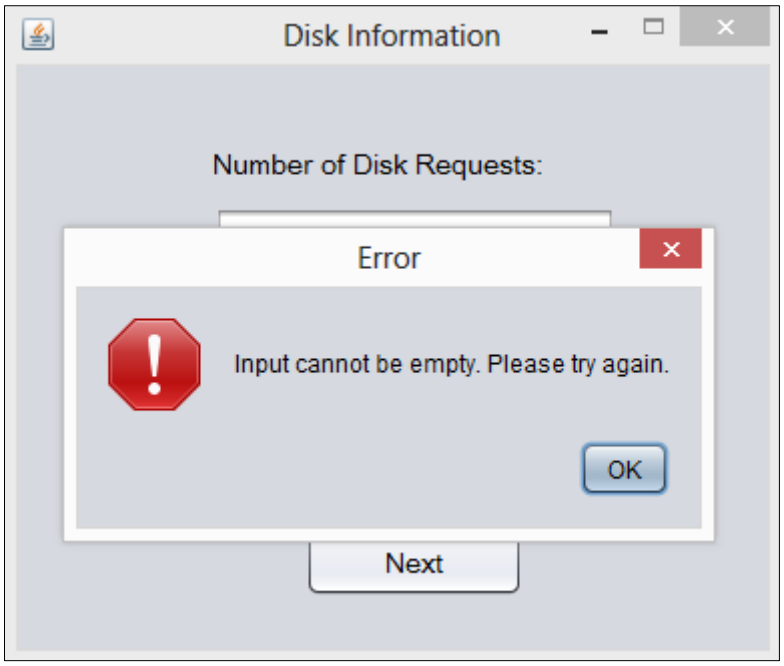
Screenshots of the System



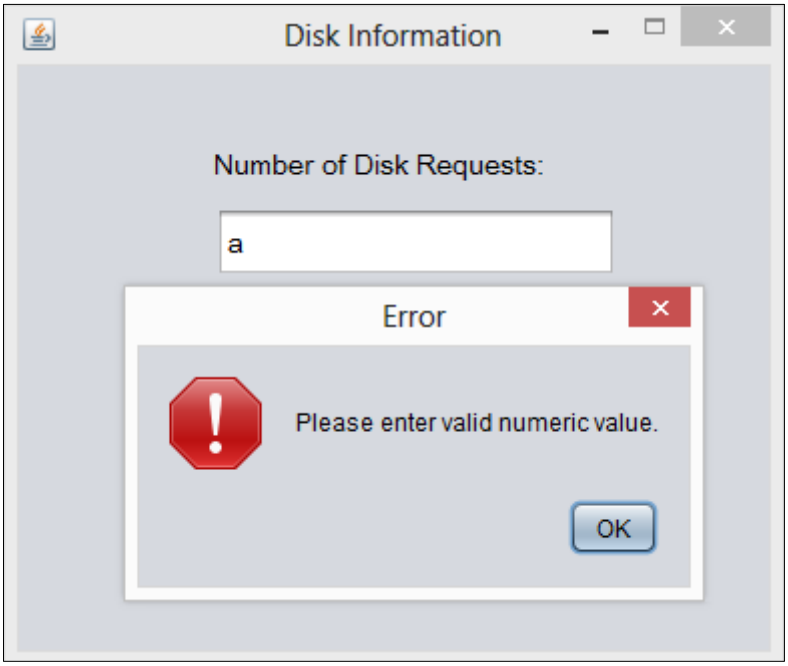
- Selecting the type of disk scheduling algorithm from three of the following:
    i. C-SCAN
    ii. LOOK
    iii. C-LOOK



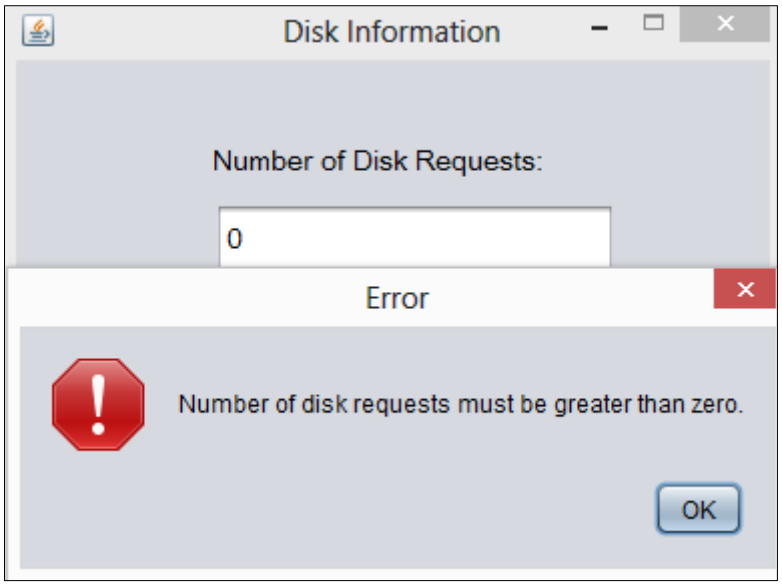
- Tool tip text is set to remind users that only numeric value is accepted.



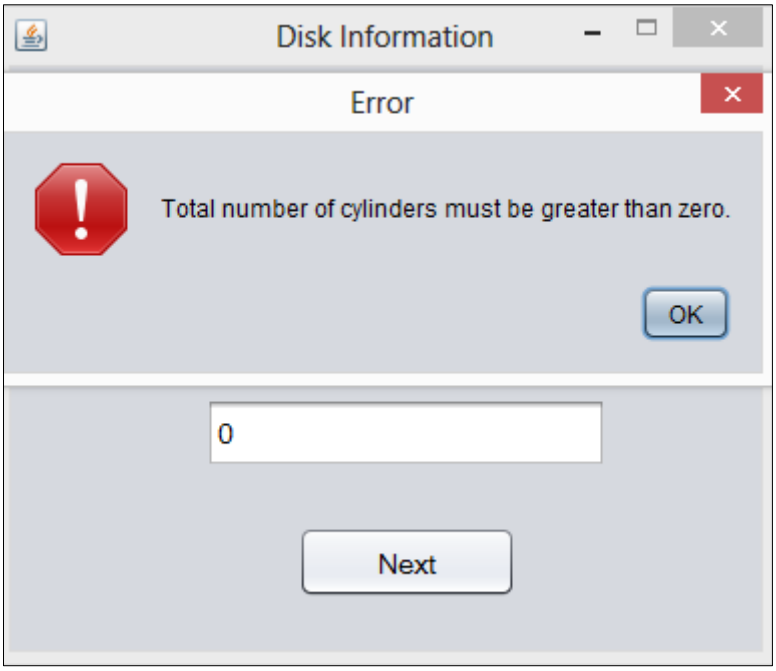
- Error message will prompt out if users proceed to the next step without entering any inputs.



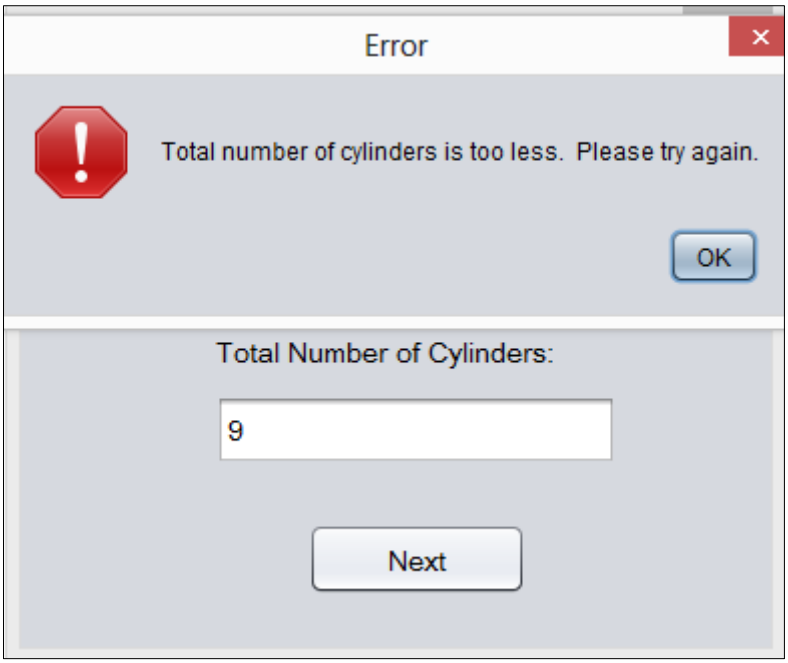
- Error message will prompt out if users enter input that is not numeric.



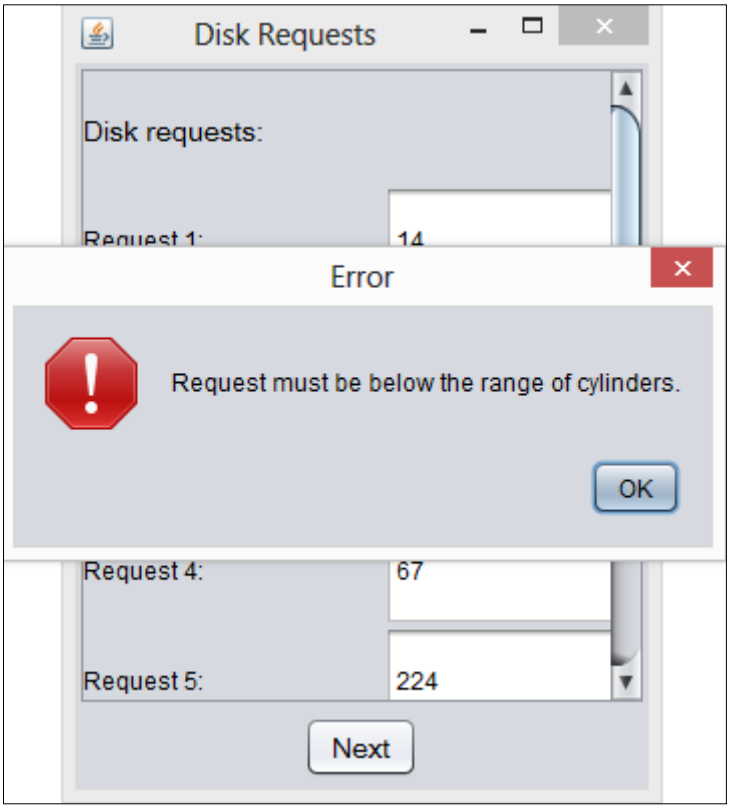
- Error message will prompt if users input number of disk requests with number 0.



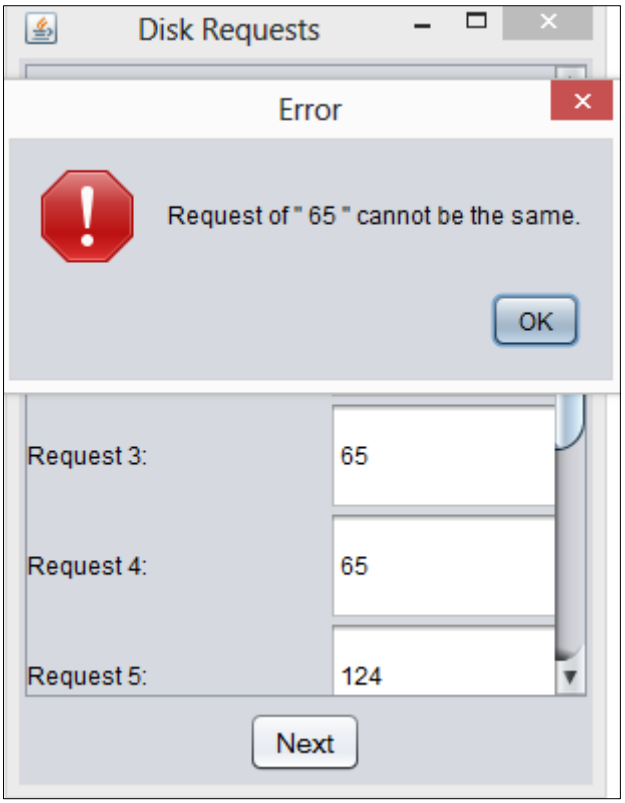
- Error message will prompt out if the users input the total number of cylinders with number 0.



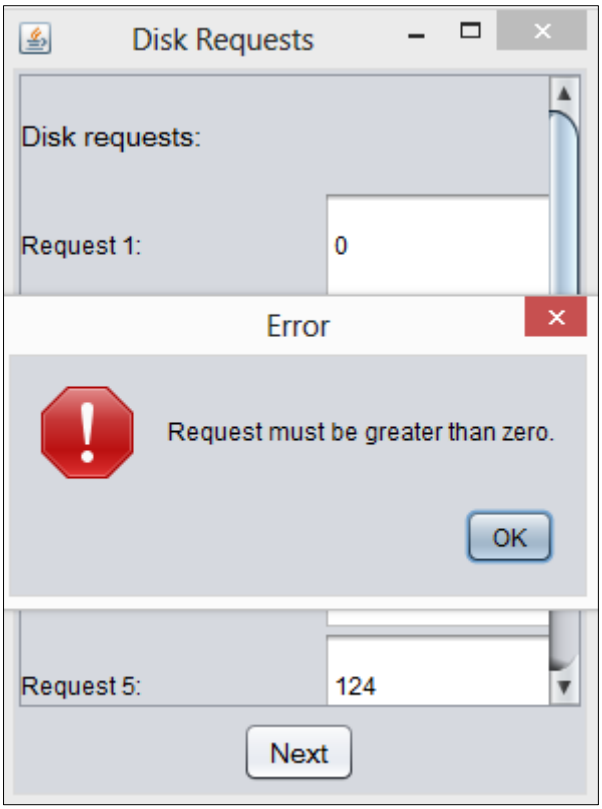
- Error message will prompt out if the number of total number of cylinders entered is too less.



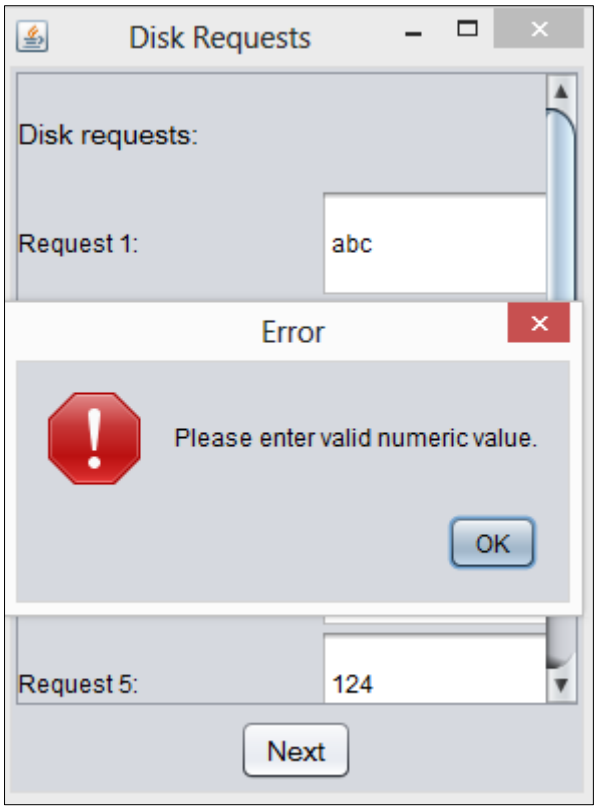
- Error message will prompt out if users input request that is out of the range of the cylinder.



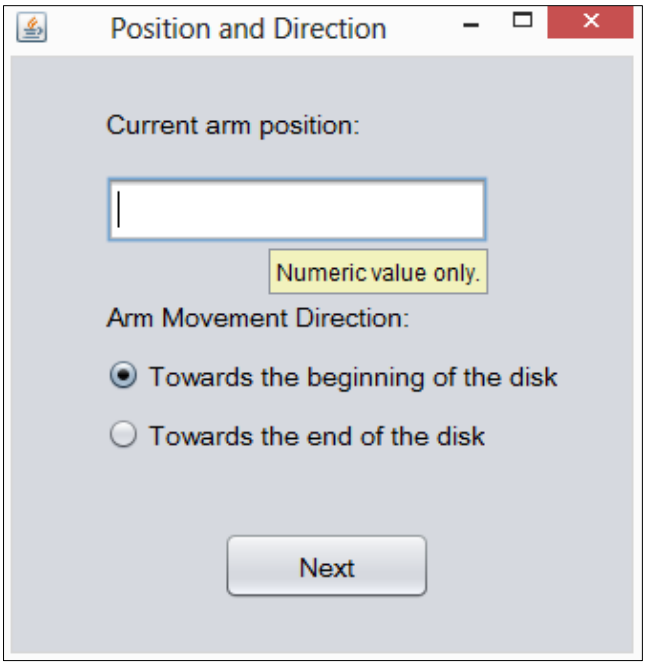
- Error message will prompt out if users input two same values of requests.



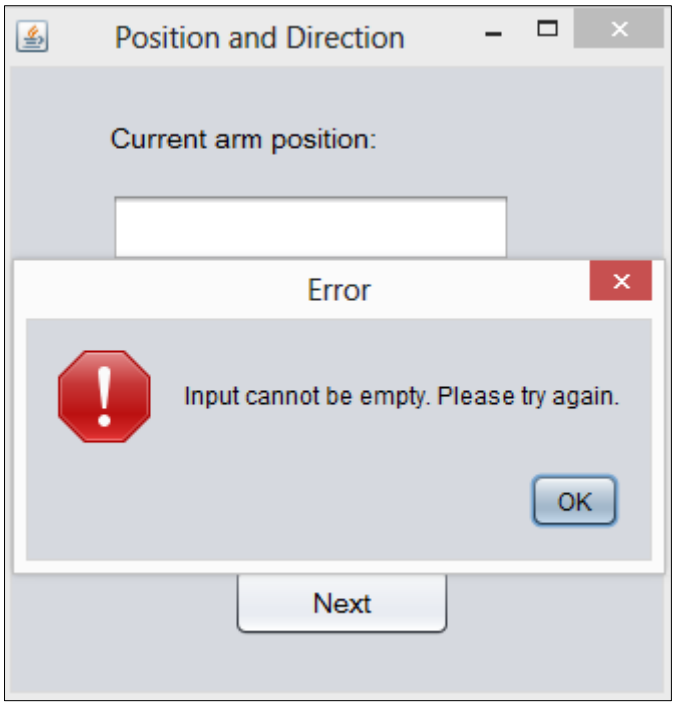
- Error message will prompt out if the users input the total disk request with number 0.



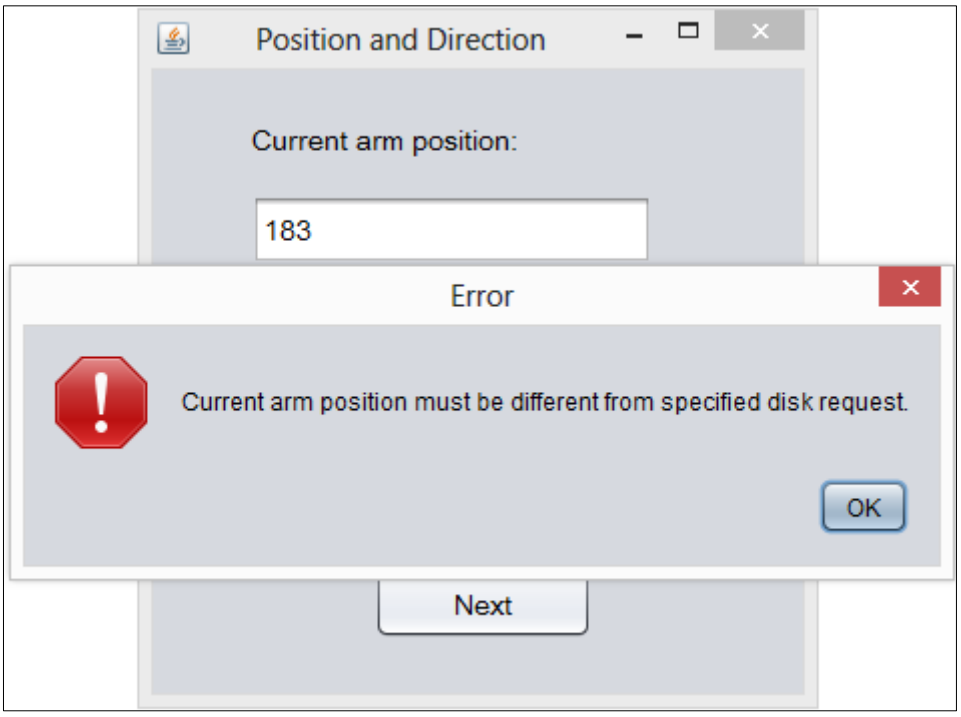
- Error message will prompt out if users enter non-numeric value.



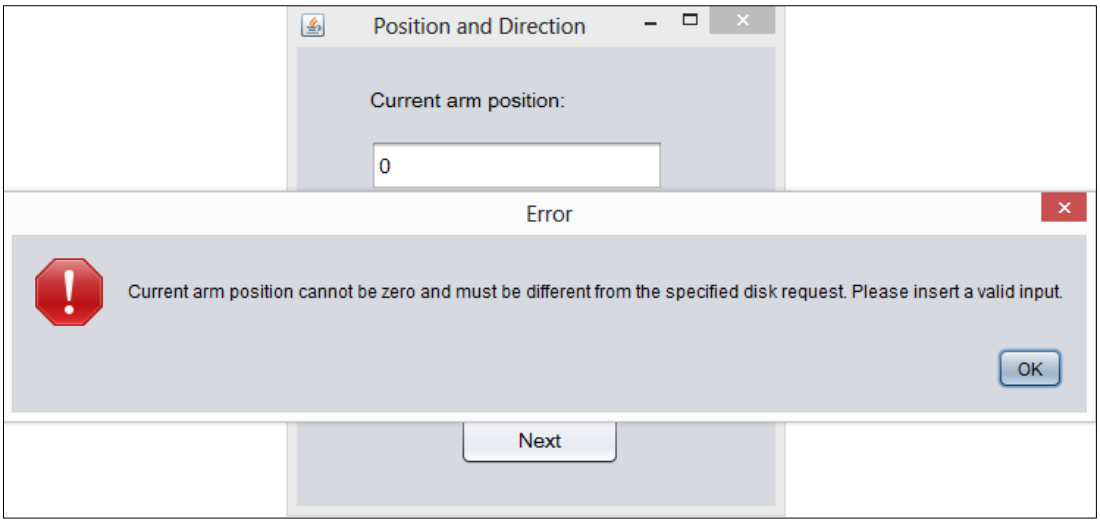
- Tool tip text is set to remind users that only numeric value is accepted.



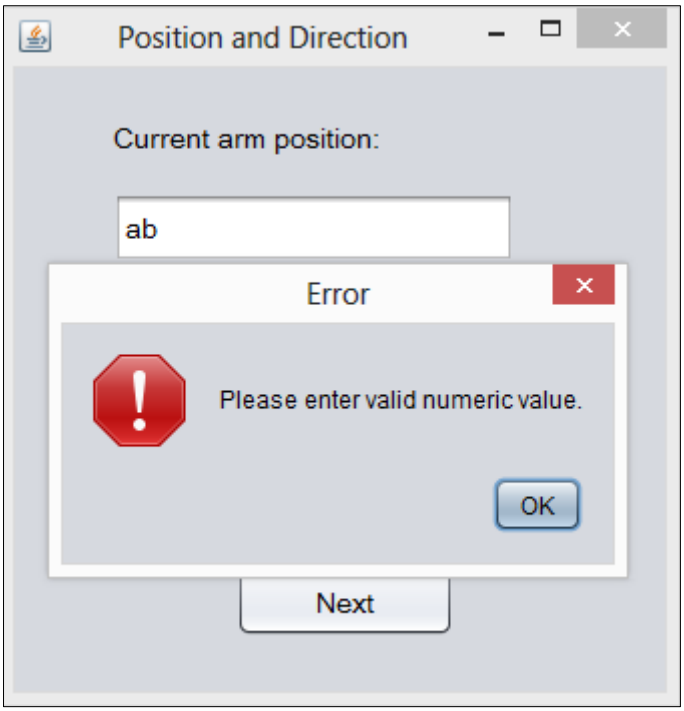
- Error message will prompt out if users proceed to the next step without entering any inputs.



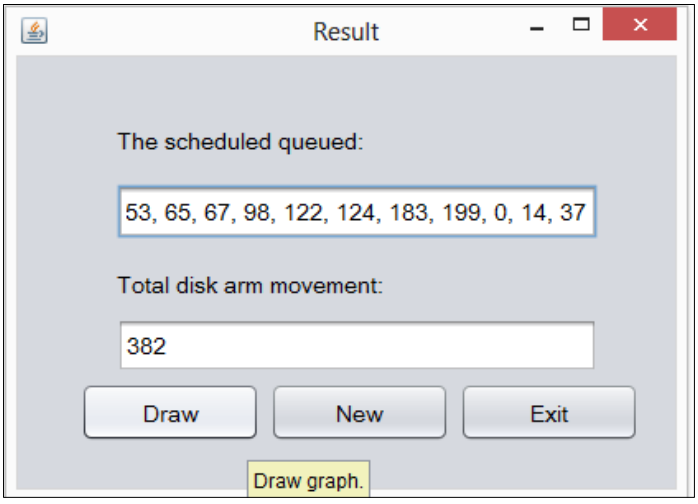
- Error message will prompt out if the current arm position is the same as one of the disk request.



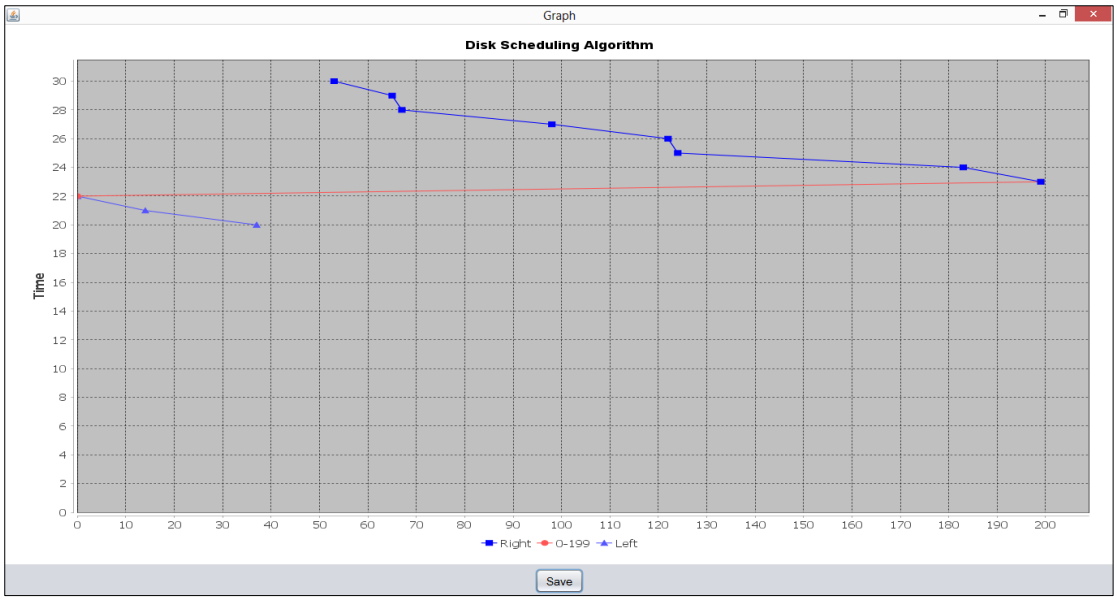
- Error message will prompt out if current arm position entered is 0.



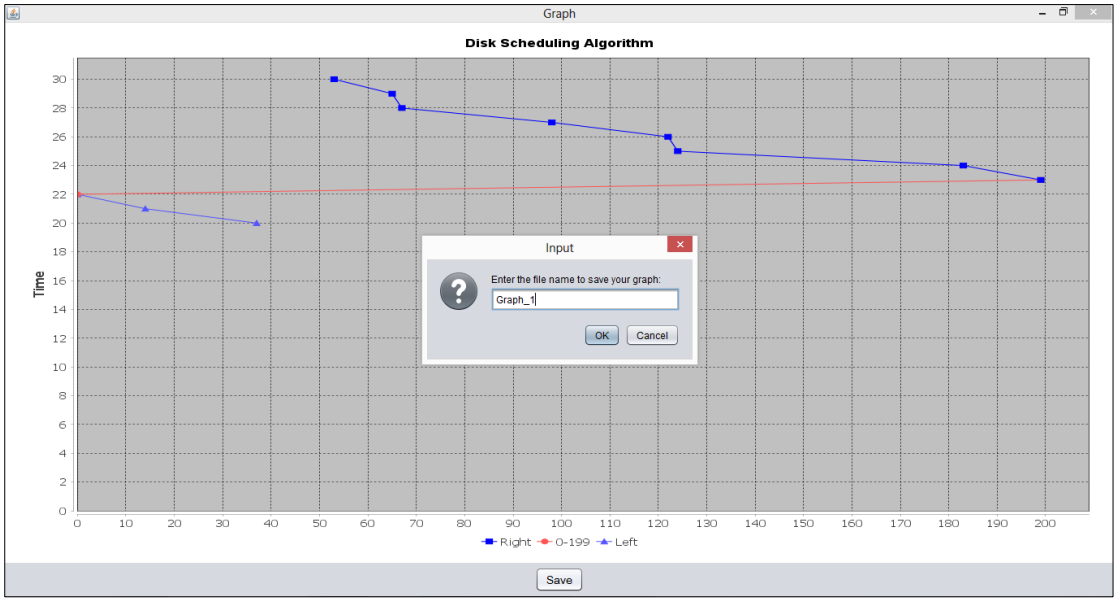
- Error message will prompt out if users enter non-numeric value.



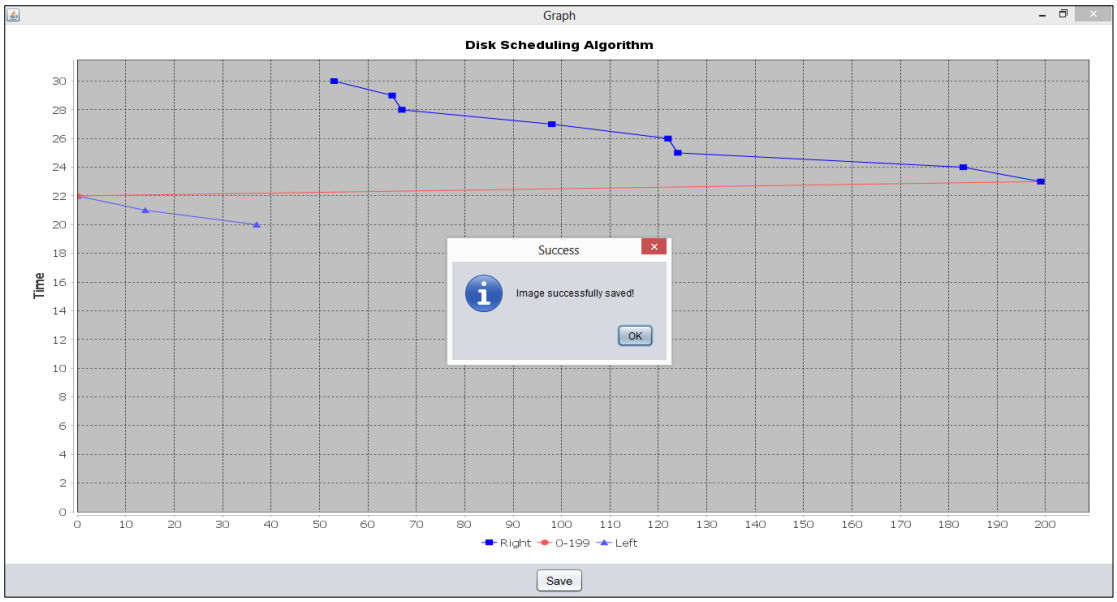
- Results will be shown.
- Tool tip text for specific button for the information of the users.



- Graph will be generated according to the result.



- Graph can be saved and users are able to name the graph as they like.



- Information message will prompt out if users successfully save the image.

Discussion

1. Strength of the System

The biggest strength of the system is that the design is simple and easy to use. Users can easily know how to use our system regardless of the experience status of the users. The system is designated in a step-by-step sequence, and users will be guided to the next process accordingly. The sequence will not be interrupted to avoid any confusion cause to the users.

The next strength of our system is that it matches between the system and the real world. In our system, only simple English is used. We also avoid using any technical words that only certain users can understand. This is to ensure that users can understand what is needed for the system to function. Besides, we ensure the consistency of the wordings used in our system. For example, to go from one window to another window, we always use “Next” button to guide the users to proceed to the next step.

Our system includes error validation which helps users to realize and recognize the error from their inputs, which makes it an important strength for our system. For instance, error validation used for checking the validity of the input is provided in our system. If users input any data that is not numeric, the error will be detected and error message will prompt out to notify the users. Users will then have to re-enter the input until valid input is detected by the system.

Another strength of our system is the function of generating graph based on the scheduled queue. After users get the result of the queue, users are able to generate the graph by just one click of the button. This additional function helps to ease users to have a clearer view of the disk scheduling algorithm. Other than generating graph, our system also allows users to save the graph as image.

Lastly, in our system, there is also another significant feature available, which is the tool tip text. Tool tip text is one of the strength of our system, as it is used to remind the users. For example, tool tip text is available in text field to remind users that only numeric value is acceptable as input. This is to aid users when inputting. Another use of tool tip text is in the "Result" window, where tool tip text can be found under "Draw" and "New" button as guidance for users.

2. Limitation and Future Improvement

There are some minor limitations in the system that is being developed. The first limitation is the consistency of components in "Disk Request" window. The size of text field that is generated in "Disk Request" is not fixed. The size of the text field will change according to the number. When there are more requests, the size of the text field will be smaller, and the size will be bigger if users have lesser requests. We have tried to set the size of the JTextField to standardize the size of text field, but still it is not working. The same goes to the size of the button in that window. As for other windows, the sizes of the buttons are being standardized to the same size, but the button in "Disk Request" window is unable to change. We have also tried to fix the size of the button but the size will still appear as default button size.

Another limitation in terms of consistency in "Disk Request" window is regarding to the tool tip text. In our system, we have included tool tip text to aid users when inputting data. However, there is error while adding tool tip text in that window. We have tried to add by implementing ".setToolTipText" in the for loop where text field will be generated. When we try to run the system, there will be error for switching from the previous window to "Disk Request" window.

The next small limitation is in the graph generated. The colour of the line for the graph is not consistent, which means the colour will change when the graph change direction. We hope to improve by standardizing the colour of the line graph so that it can be the same and will not cause confusion to the users. Another limitation regarding to our graph is there is no arrow showing the direction of the disk scheduling. There should be an arrow indicating the next point and direction of the graph.

As for the future improvement, we hope to be able to solve all the limitations listed. Other than that, we will improve our user interface to make it more attractive to the users, as the interface we are having now is not attractive enough to pull user's attraction. We also wish to include other type of disk scheduling algorithm in our system, such as SSTF (Shortest Seek Time First), FCFS (First Come-First Serve) and SCAN.

Teamwork

There are a total of six members in our group, who are John Loh Ern-Rong, Loh Jun Kean, Louis Juliendo, Ivon Lai Jia Yi, Regine Lim and Toh Kai Sin. We have conducted meeting on every Wednesday, Thursday, and Sunday starting from the end of June. We discussed about the things to do which include task division, schedules for meeting, consideration of (Graphical User Interface) GUI design, ways to implement the code and components to use.

Phase 1

We created the GUI in the JFrame form by following the assignment guideline. We brainstormed about the requirement for every GUI used and recorded our ideas for future use. After that has been done, the codes were given to Louis.

After that, we started to do research on the how to develop the Disk Scheduling Algorithm calculation by using Java. Regine, Louis and Ivon started to write their own version of Java codes for CSAN, CLOOK, and LOOK towards the end. In the end, we decided to use the algorithm provided by Louis since the codes are easier to understand and implement.

Then, Ivon and Kai Sin were in charge to draw the Flow Chart of our system process based on the early discussion. They were also in charge of doing pseudocodes based on the flowchart created.

Phase 2

Firstly, John, Louis and Jun Kean were assigned to implement LOOK, CLOOK, and CSCAN algorithm towards the end and Regine, Ivon, and Kai Sin were assigned to do LOOK, CLOOK, and CSCAN algorithm towards the beginning based on the codes provided by Louis in Phase 1.

After we completed our parts, Kai Sin, Ivon, Regine, and John were assigned to in charge in completing error checking and validation. We used few types of error validation in our system, including If-Else and Try-Catch methods. After Ivon and Regine developed the codes, they passed the codes to John and Kai Sin for

implementation. On the other hand, John and Jun Kean were responsible to do research on generating text field dynamically upon clicking a button, get data from those text fields and store it into an integer array. After they complete the research, they applied the codes to the GUI and passed to Louis.

Next, Regine and Jun Kean performed and managed the standardization of variable name, methods of name, and re-structured the codes indentation. John and Jun Kean applied the concept to the codes after they have done with their discussion. Meanwhile, Louis and Kai Sin were responsible to do research on how to use JFreeChart. Then, they wrote and linked the algorithm to other JFrame after testing the algorithm.

At the final step, Louis, Jun Kean and Kai Sin were assigned to write the algorithm for generating the graph. Jun Kean and Kai Sin explored on how to plot the graph for Disk Scheduling Algorithm. Then, they passed the codes to Louis for testing. John then implemented the save graph function in the last window.

Upon completing, every member tried out the codes to ensure that the codes can work well and the results given were accurate.

Conclusion

In a nutshell, there are indeed many disk scheduling algorithms that have been created that help a lot at dealing with multiple processes on CPUs. Not only there are algorithms that are mentioned above, C-SCAN, C-LOOK and LOOK are used by people, but there are also other algorithms that were made earlier such as First-Come, First-Served (FCFS), Shortest-Seek-Time-First (SSTF) and SCAN. Therefore, in a selection of a disk scheduling algorithm, we must first find out a few factors. For example, systems that place very heavy process (thus creating heavy load) should use the SCAN or the C-SCAN algorithm and they can perform better. This is due to the head movement that does not switch all the time as it only switches its direction when it reaches the end or the beginning of the disk. Of course, the performance of these systems can vary based on the number and the type of requests being processed. Also, the problem of starvation can also be solved using these two algorithms. But in normal and usual cases, SSTF or the LOOK disk scheduling algorithm can be used as they usually provide reasonable results.

Therefore, it is crucial for us to know and study the importance of these disk scheduling algorithms because of its provision of a better and wise solution to minimize the access time required to read and write data to the hard disk. Otherwise, there will be much more overhead in reading and writing of data. Through this project, we have definitely gained a more in-depth understanding in disk scheduling algorithms as to why operating systems is responsible for managing the hardware efficiently and that it truly relies on the type of algorithm it needs when dealing with multiple processes.