**Afrah Abdulmajid**

**COMP2004**

**201158243**

**Assignment 5**

Table of Contents

[1. User Guide 3](#_Toc506416705)

[1.1 System Requirements 3](#_Toc506416706)

[1.2 Format 3](#_Toc506416707)

[1.3 How To Run 3](#_Toc506416708)

[1.4 Sample Execution 4](#_Toc506416709)

[2. Top – Down Design 7](#_Toc506416710)

[3. Dictionary of Variables 8](#_Toc506416711)

[4. Dictionary of Function and Methods 8](#_Toc506416712)

[5. The Code 9](#_Toc506416713)

[6. How the code works 12](#_Toc506416714)

[7. Limitations 12](#_Toc506416715)

# User Guide

The following program is coded in Java to simulate how partitions are dynamically allocated using the Best Fit algorithm with a series of adding and removal of jobs. This program will display the changes made to the memory with the addition and removal. The code also demonstrates how the free partitions are picked depending on the jobsSize of the job and remaing free space within the memory. The code uses User Input to enter the job and give it a size for it to be added, however for a job to be removed the user will just input the name of the job he/she would like to remove. At each given function call the changes made to the memory partitions and free list will be outputted. That is the jobs within the memory partitions and the freelist.

## System Requirements

The .java files contains the program specified, the following program can be edited with any IDE (Visual Studio, Eclipse. IntelliJ, etc.) or any text editor (Notepad++, XCode, Sublime Text, etc.). The program could be run on any operating System from the command prompt, provided that JDK (Java Development Kit) is installed on the machine the program is being run on.

## Format

The input format of the program is to correctly spell the words “add” and “remove” , when the program as the program prompts, and so the program will run based on the user input and produce the result accordingly. As well as correctly spelling “yes” or “no” when the program prompts you on the next action.

## How To Run

The file can be accessed from the USB file within the directory Assignment 5. Within the Assignment 5 folder there is a file called MemoryManagment.java. The steps to run the program are as follows:

**Step 1:** Open the Command Prompt (cmd), also known as the terminal

**Step 2:** Navigate to the directory that the program is saved in using the command ***cd /absolutepathname***, for example, */Users/Afrah/Desktop/cs2004*.

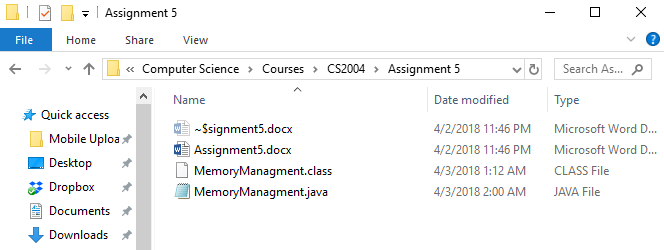
**Step 3:** Type in; Javac MemoryManagement.java

**Step 4:** Type in; Java MemoryManagement

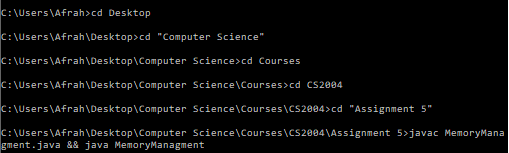
The program will now execute.

## Sample Execution

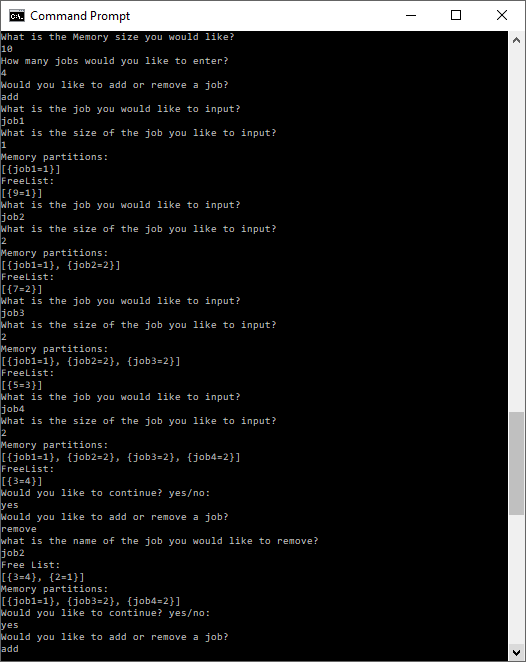
Below I have attached an example of how the program is run using the command prompt as well as the output corresponding to system calls.



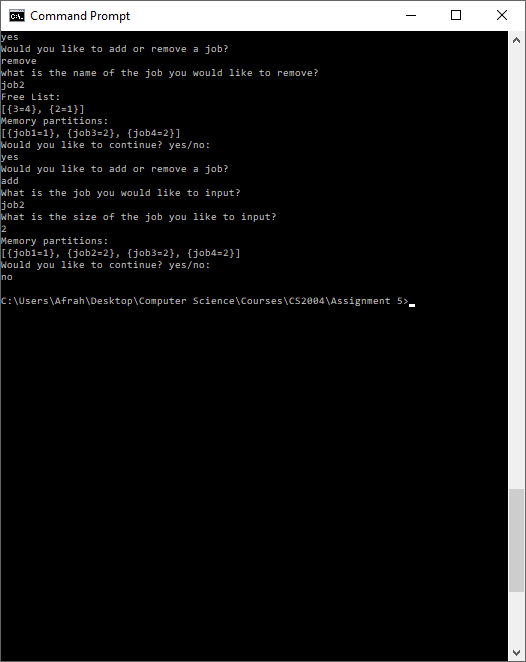
*Figure 1.0: The directories and the files stored in CS2004 folder*



*Figure 2.0: The following are the commands used to navigate to the directory the file is saved in.*



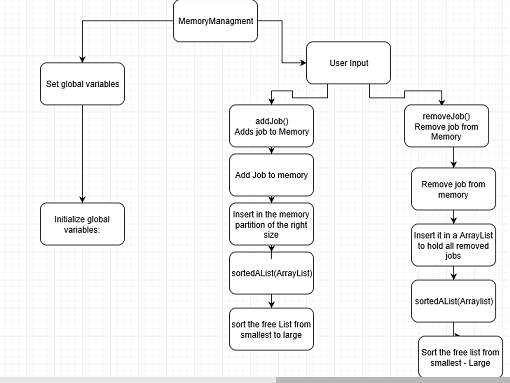
*Figure 3.0: Sequence of inputs to add/remove jobs and allocate a partition to it, each memory partition stores a type treeMap that stores Key and Value, jobname and JobSize*



*Figure 4.0: Sequence of removing or adding jobs cont’d*

The following program will continue to run or stop based on the User input

# Top – Down Design



# Dictionary of Variables

The following global variables are initialized in the program. Where they are passed in to functions

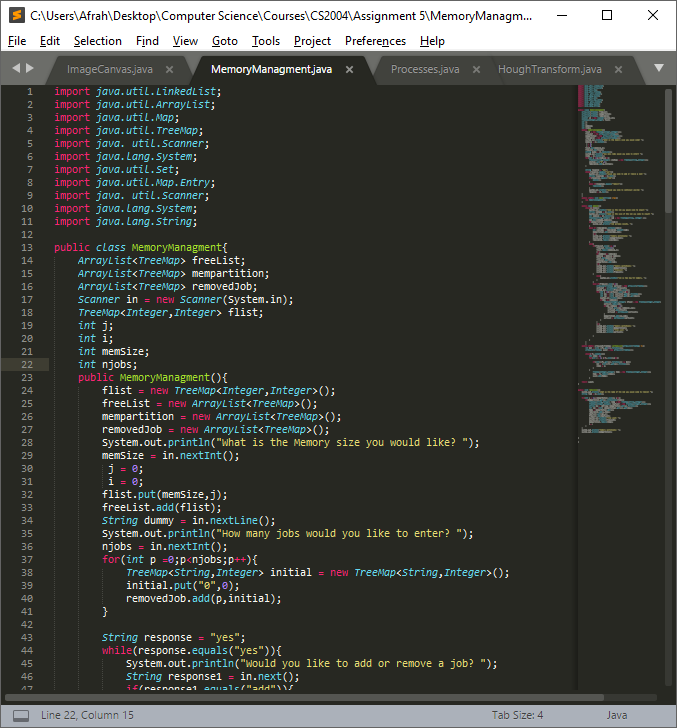
|  |  |
| --- | --- |
| freeList | It is an ArrayList of Treemap, that stores the size of a free partition at the memory and the index at which the free partition is. |
| mempartition | It is a ArrayList of TreeMaps that stores the Jobs, with a given name and size |
| In | It is a Scanner object that processes the input from the command prompt |
| removedJob | It is a ArrayList of TreeMap that store jobs that have been temporarily removed. |
| flist | It’s a TreeMap that stores the size of the free partition and the index at which the partition is located. |
| memSize | The size of the memory. |
| njobs | The number of jobs the user would like to add |
| i | Counter to keep track of the number of jobs |
| j | Counter to keep track of freelist index |

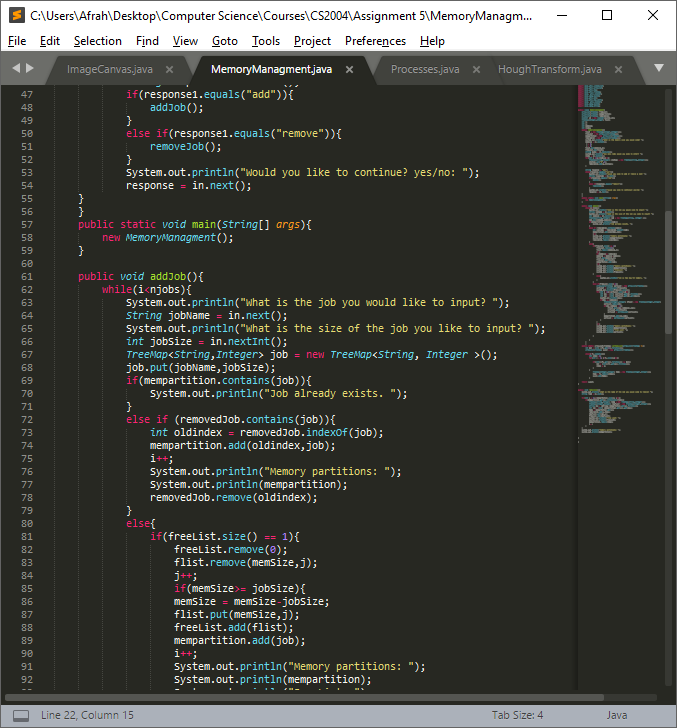
# Dictionary of Function and Methods

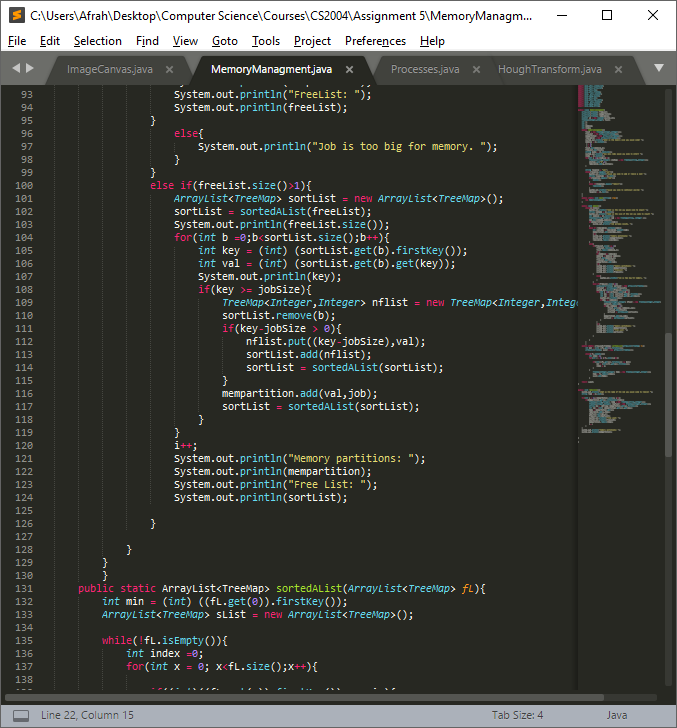
Listed below are the list of function or methods that are used with the program. There are two types of functions in these programs, some are builtin and the other functions are ones that was written and implanted by the author.

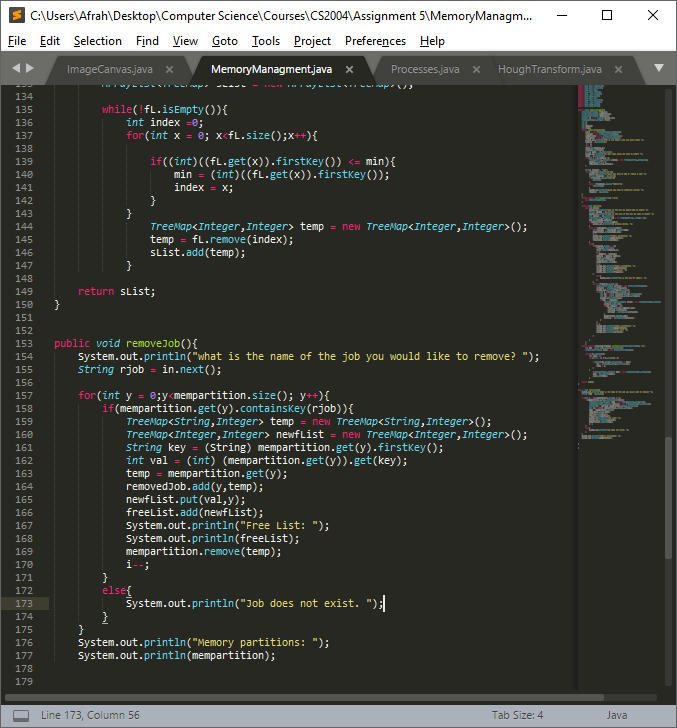
|  |  |
| --- | --- |
| main(String[] args) | The function that runs the program |
| MemoryManagment() | Constructor of the MemoryManagment class |
| addJob() | This function adds jobs to the memory ArrayList |
| removeJob() | This function this function removes a job from memory ArrayList |
| sortedAList(ArrayList<TreeMap> fL) | This function sorts the free list according to its size from smallest to largest |

# The Code









# How the code works

The code is designed to create a memory, with a given size with user interface. When the memory is created the User is prompted to add a certain number of jobs to the memory, as the jobs are being created they are allocated partitions dynamically. The jobs created will be a TreeMap type object that will store the JobName and JobSize to be added or removed in a {Key: Value} pair. Partitions are created as jobs are being added, we keep track of the free space we have remaining as we are adding the jobs and removing them as well. If a job is removed and brought back again, the job will be allocated the previous partition it was given due to tight binding.

The memory is implemented as a dynamic ArrayList of TreeMaps that allows partitions to be created dynamically. The user will then choose the sequence of jobs he/she would like to add or remove. As those calls take place the output at each call will be the current contents of the Memory and the FreeList.

# Limitations

The limitations that come with the code is the current implementation is that when only one job can be removed at a given location.