

Course Project
Phase 2: Memory Management

Table of Contents

Introduction	1
Tools.....	1
Input File	1
How to execute the Project.....	4
Project Report.....	6
First – Fit.....	6
Best – Fit	7
Worst-Fit	8
Final Output.....	9
Packages	9

Introduction

The project is to create a Memory management algorithm simulator, which will be able to emulate the memory management usually used by OS. The project here is reading the list of process from a text file and based on requirement is doing one of the below scheduling algorithm;

1. First-Fit
2. Best-Fit
3. Worst-Fit

Used CLI User Interface and modular design fashion using different modules based on the operating system function perform during memory scheduling.

Tools

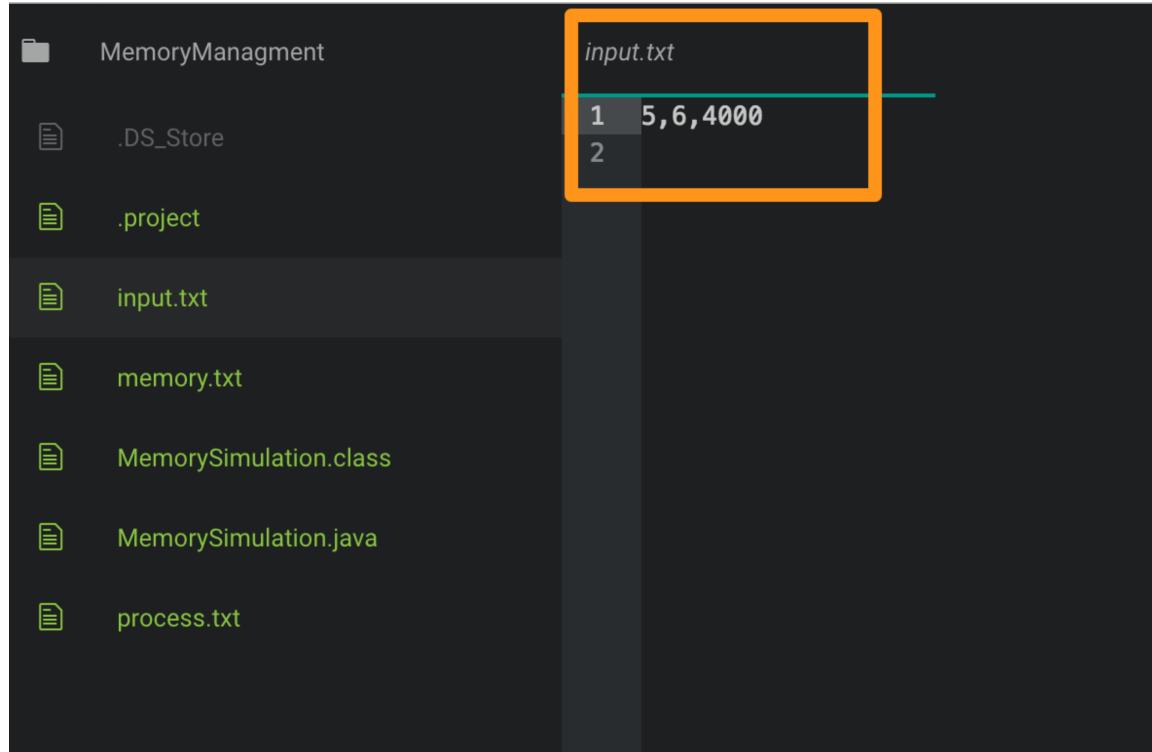
The project is built using ATOM IDE (we can use any IDE) and using java v1.8.

Input File

Input.txt: This contains three inputs

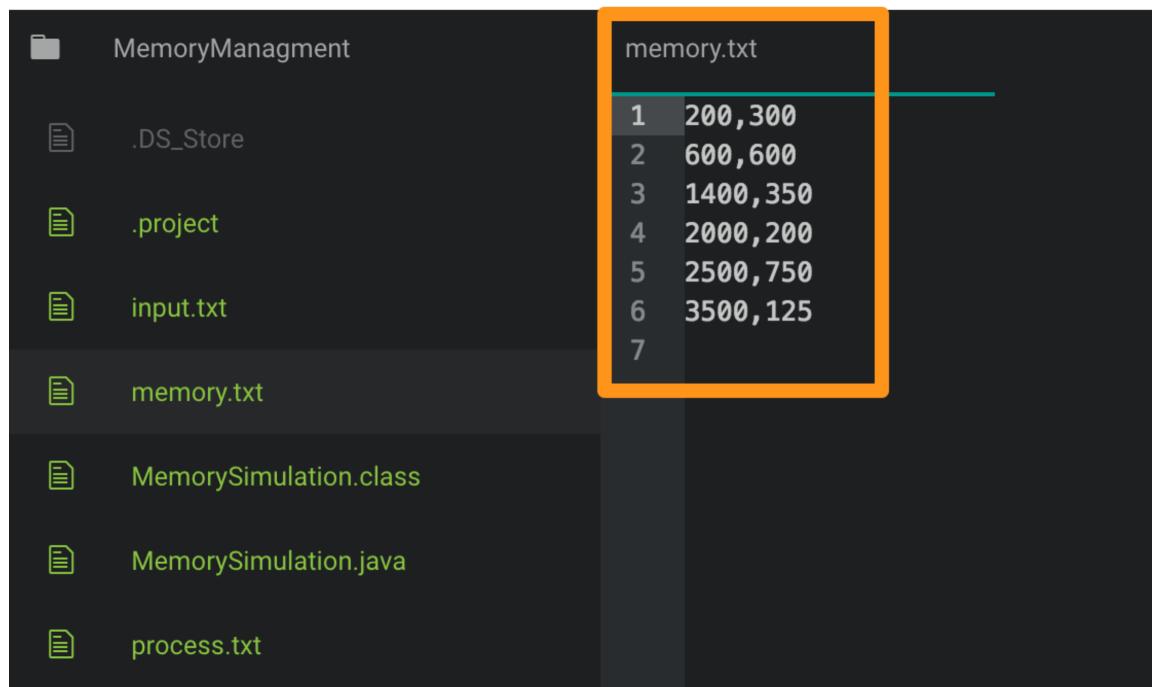
- Maximum number of process (for this project M = 5

- number_of_processes_for_memory_allocation)
- Maximum number of memory block (for this project N = 6 Number_of_available spaces)
 - Maximum memory (for this project Size_of_memory = 4000)



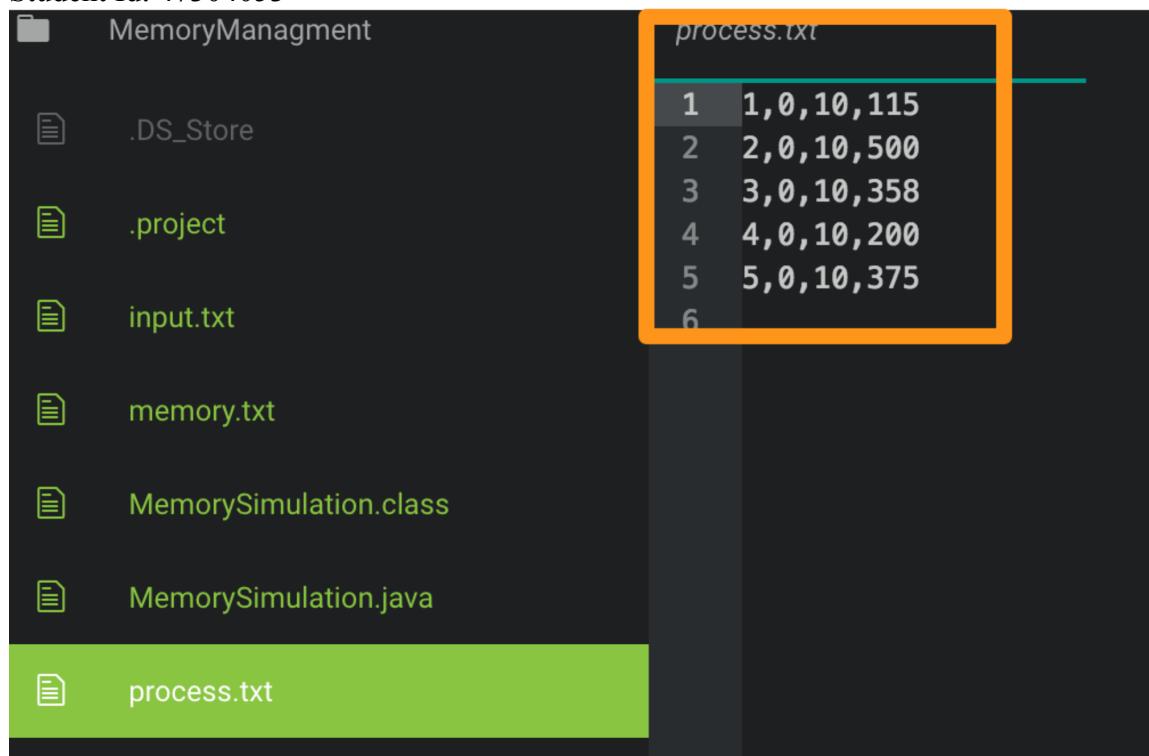
memory.txt: This contains two values

- Starting Memory Address
- Available space



process.txt: This file contain details about process

- PID
- Arrival Time
- Duration
- Size of memory



How to execute the Project

For Windows go to cmd prompt or Mac terminal . Navigate to the source code folder attached with the project.

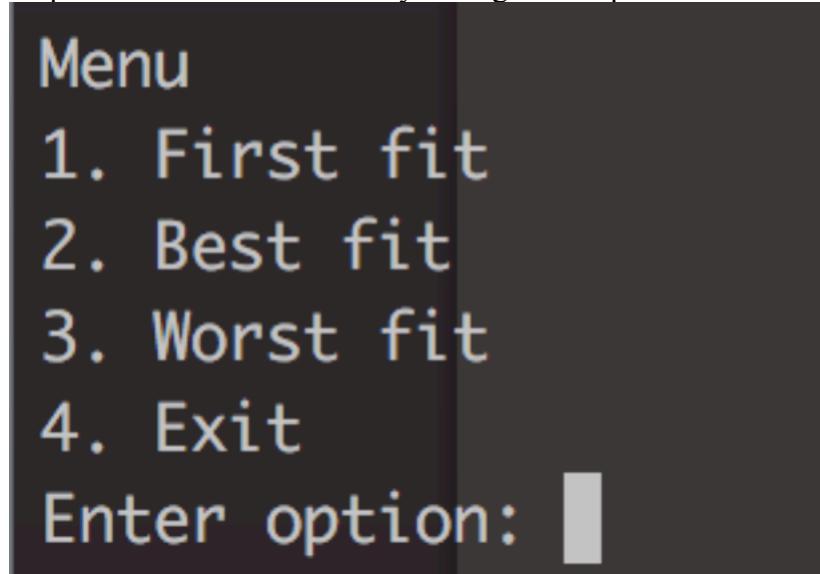
Step 1: Compile the code `javac MemorySimulation.java`

Step 2: Once compile successful execute the program with `java MemorySimulation`

Step 3: The next input is to allocate the Memory block as free or used. Enter 1 for used and 0 for free.

```
$ java MemorySimulation
Enter block usage scenario
Is block 1 used ( Enter 1 ) or free ( Enter 0 )?
0
Is block 2 used ( Enter 1 ) or free ( Enter 0 )?
0
Is block 3 used ( Enter 1 ) or free ( Enter 0 )?
0
Is block 4 used ( Enter 1 ) or free ( Enter 0 )?
0
Is block 5 used ( Enter 1 ) or free ( Enter 0 )?
0
Is block 6 used ( Enter 1 ) or free ( Enter 0 )?
0
Memory
```

Step 4: Next select the Memory management option



Project Report

First – Fit

Based on the algorithm, the allocator keeps a list of free blocks (known as the free list) and, on receiving a request for memory, scans along the list for the first block that is large enough to satisfy the request.

The output is as shown below:

```
<-----Algorithm: FIRST FIT----->
-----
Assigned Process 1 is in block 1
Starting address of Process 1 is of 200
Process memory need 1 is of 115
-----
Assigned Process 2 is in block 2
Starting address of Process 2 is of 600
Process memory need 2 is of 500
-----
Assigned Process 3 is in block 5
Starting address of Process 3 is of 2500
Process memory need 3 is of 358
-----
Assigned Process 4 is in block 3
Starting address of Process 4 is of 1400
Process memory need 4 is of 200
-----
Assigned Process 5 is in block 5
Starting address of Process 5 is of 2500
Process memory need 5 is of 375
-----
<-----FINAL REPORT----->
Blocks used = 5
Blocks free = 1
Number of process blocked= 0
Number of process = 5
Total used space = 1548
Total free space = 2452
Total Blocking Probability= 0.0
Total Memory Utilization = 38.7
<-----END----->
```

Best – Fit

The best-fit deals with allocating the smallest free partition which meets the requirement of the requesting process. This algorithm first searches the entire list of free partitions and considers the smallest hole that is adequate. It then tries to find a hole, which is close to actual process size needed. The output is as shown below:

```
<-----Algorithm: BEST FIT----->
-----
Assigned Process 1 is in block 6
Block size after allocation block 110
Starting address of Process 1 is of 3500
-----
Assigned Process 2 is in block 2
Block size after allocation block 2100
Starting address of Process 2 is of 600
-----
Assigned Process 3 is in block 5
Block size after allocation block 3392
Starting address of Process 3 is of 2500
-----
Assigned Process 4 is in block 4
Block size after allocation block 40
Starting address of Process 4 is of 2000
-----
Assigned Process 5 is in block 5
Block size after allocation block 517
Starting address of Process 5 is of 2500
<-----FINAL REPORT----->
Blocks used = 5
Blocks free = 1
Number of process blocked= 0
Number of process = 5
Total used space = 1548
Total free space = 2452
Total Blocking Probability= 0.0
Total Memory Utilization = 38.7
```

Worst-Fit

In worst fit approach is to locate largest available free portion so that the portion left will be big enough to be useful. It is the reverse of best fit.

The output is below:

```
-----Algorithm: WORST FIT-----
-----
Assigned Process 1 is in block 5
Block size after allocation block 1635
Starting address of Process 1 is of 2500
Process memory need 1 is of 375
-----
Assigned Process 2 is in block 5
Block size after allocation block 2135
Starting address of Process 2 is of 2500
Process memory need 2 is of 375
-----
Assigned Process 3 is in block 2
Block size after allocation block 3242
Starting address of Process 3 is of 600
Process memory need 3 is of 500
-----
Assigned Process 4 is in block 3
Block size after allocation block 4150
Starting address of Process 4 is of 1400
Process memory need 4 is of 358
```

```
The Process 5 is in block state
<-----FINAL REPORT----->
Blocks used = 4
Blocks free = 2
Number of process blocked= 1
Number of process = 5
Total used space = 1173
Total free space = 2827
Total Blocking Probability= 20.0
Total Memory Utilization = 29.325
`           END`
```

Final Output

First-Fit:

```
<-----FINAL REPORT----->
Blocks used = 5
Blocks free = 1
Number of process blocked= 0
Number of process = 5
Total used space = 1548
Total free space = 2452
Total Blocking Probability= 0.0
Total Memory Utilization = 38.7
<-----END----->
```

Best-Fit:

```
<-----FINAL REPORT----->
Blocks used = 5
Blocks free = 1
Number of process blocked= 0
Number of process = 5
Total used space = 1548
Total free space = 2452
Total Blocking Probability= 0.0
Total Memory Utilization = 38.7
<-----END----->
```

Worst-Fit:

```
<-----FINAL REPORT----->
Blocks used = 4
Blocks free = 2
Number of process blocked= 1
Number of process = 5
Total used space = 1173
Total free space = 2827
Total Blocking Probability= 20.0
Total Memory Utilization = 29.325
<-----END----->
```

Packages

```
import java.io.*
import java.util.*
```