

# RESEARCH & PROJECT SUBMISSIONS



## **Program:**

**Course Code:** CSE322

**Course Name:** Operating Systems

## **Examination Committee**

**DR. Sahar M. Mahmoud Haggag**

**Ain Shams University  
Faculty of Engineering  
Spring Semester – 2020**



## Student Personal Information

Student Name: محمد عمرو احمد طة منسي  
Student Code: 1601247  
Class/Year: 3<sup>rd</sup> electrical year computer department Section 3

## Plagiarism Statement

I certify that this assignment / report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they are books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment / report has not been previously been submitted for assessment for another course. I certify that I have not copied in part or whole or otherwise plagiarized the work of other students and / or persons.

Signature/Student Name: محمد عمرو احمد طة منسي

Date: 10/6/2020

## Submission Contents

01: research

02: project



**01**

*research*



### process scheduling algorithms\_ brief:

- algorithms are non-preemptive or preemptive Non\_ preemptive is a algorithms in which the process cant not preempted until finish it's time in cpu but preemptive one is depend on priority or shortest one time in which Scheduler can preempt a low priority process in cpu at any time when high priority want to enter to cpu.

#### 1-FCFS:

- The first process come to running queue. is the first serve.
  - Can be preemptive or not
  - Its implementation by FIFO queue or based on it
  - Average\_wait time is high.so have low performance

#### 2-SJF:

- Can be preemptive or not
- Best approach why?! due to decrease waiting time.
- Easy to implementing it in Batch systems due to it's required cpu time is known in early
- Impossible to implementing it in interactive systems due to needed cpu time which is not known.

#### 3- Priority:

- Can be preemptive or not
- Each process have priority in which process have high one execute firstly
- While process have same priority executed by fcfs
- Priority decided based on memory, time or any resource requirements.

#### 4- Shortest Remaining Time:

- is the preemptive type of the SJF algorithm.
- impossible to implementing it in interactive systems due to required cpu time which is not known same as in SJF
- used in batch environments due to give preference to short jobs as needed



#### 5- Round Robin Scheduling:

- Can be preemptive or not.
- Each process have constant time to run, which called a **quantum**.
- Once a process is run in given time quantum, it is preempted after it 's quantum and other process run in given time quantum
- for not lost states of preempted processes should context switching is used.

#### 6- Multiple-Level Queues Scheduling:

Multiple-level queues aren't independent scheduling algorithm why?!

Due to use group of other present algorithms and schedule jobs have

Common Characteristics

- Multiple queues are kept for processes with common characteristics
- Every queue have its private scheduling algorithms.
- Priorities are given to each queue.

#### [memory allocation algorithms\\_brief:](#)

##### 1-First Fit:

In it allocate depend on the first free partition or hole large enough which can put up the process. It finishes on finding the first suitable free partition.

- Fastest algorithm why?! Due to it searches as little as possible
- The request for larger memory requirement can not be accomplished why?! Due to



remaining unused memory areas left after allocation become waste I when it is too smaller

#### 2-Best Fit:

It is allocate the smallest free partition which have the requirement of the process which is request. In which that algorithm first searches the whole list of free partitions and take the smallest hole that is adequate or most fit. After that It tries to find a hole that is close enough to meet actual size of process that it needed.

- Memory utilization is best than first fit as dealing with smallest free partition first available
- It is slower
- It may tend to load memory with tiny useless holes.

#### 3- Worst fit:

It is locate process in large available free partition in which it left a big partition enough to be useful. It is opposite of best fit.

- Decrease the rate of production of small gaps.
- if a process requiring larger memory arrives in later stage so it cannot be put up as the largest hole is already split and occupied(full).
- .

#### 4- Next fit:

Next fit is a modified version of first fit.as It begins as first fit to find a free partition. In which the different is that When called next time it starts searching from where it left off, not from the beginning.



## Process scheduling algorithms

### UNIX:

- UNIX is a multi-user system in which :

scheduling algorithm is design to provide a great reaction to interactive processes in which divided two:

- First: low-level algorithm is used to schedule the processes in the memory and ready to run
- Secand: high-level algorithm is used to make the decision on which process to fetch in the memory from the disk and to be ready to run.
- The low-level algorithm has multiple waiting queues. In which queue partakes a unique range of non-overlapping priority values. Processes run in user mode have positive values but processes run in kernel mode have negative values( Negative values take the highest priority)( large positive values take the lowest)
- Notice that: Only processes that are in memory and ready to run are located on these queues.
  - The low-level scheduler searching the queues begin from the highest priority until it finds a queue in which is full, first process in this queue is then chosen to run.
  - The processes within the same priority range are using the FCFS Basis and sharing the CPU by a round robin algorithm.
  - The running process allowed to run for a maximum of one quantum or till it has been block, If the process used up its quantum time without being ended, it is put back on the end of its queue and Then scheduler searches again.



- Notice that The priority value for each process in the system is recalculated every second.

In which the Recalculating priority values of all the processes every second changes

process priorities in a dynamic way.

✚ formula used to calculate the priority: which equal base + nice + CPU usage.

- The counter tick is increasing by 1 every clock ticks to increasing priority value of the process currently using the CPU to putting it on a lower-priority queue
  - However CPU\_usage value decaying with time, Different versions of the UNIX OS do the decay slightly different
- One way is that t tick count for every process is dividing by 2 before process priorities are recalculated by using the shown above formula
- The weighting algorithm is very fast due to it has one addition and one shift
- The formula indicates that UNIX give high priority to processes that used less CPU time in the recent past. so, the lower the recent CPU \_usage of a process is, the lower its priority value is, and the higher its priority is.

Components	Representations
base	An integer usually having a value of 40 or 60.
nice	A positive integer usually with a default value of 0, ranged from -20 to +20; each process has a nice value which can be set in the range 0 to 20 by using the nice command.
CPU_usage	The number of clock ticks for which the process has used the CPU

Windows:

- Windows implements: such that a priority\_driven and preemptive scheduling system in which the highest-priority runnable thread always runs, with the limitation that the thread selected to run might be limited by the processors at which the thread is allowable to run, a phenomenon called *processor affinity*.





- When a thread is carefully chosen to run, it runs for a quantity of time called a *quantum*. A quantum mean' s the length of time a which is allowed to thread to run before another one have same priority level (or higher, that occur on a system have multiprocessor) that is given a turn to run.
  - Notice that: Windows implements a pre-emptive scheduler so if another thread with a higher priority come to be ready to run, the currently running thread may be preempted before finishing its time slice.
- scheduling code of Windows is implemented in the kernel. There's no single <scheduler> module or routine such that the code is spread throughout the kernel at which scheduling related events happen The routines that perform these duties are called the kernel's *dispatcher*.
- Windows must determine which thread should be run next. When Windows selects a new thread to run, so make *context switch* to it.
  - Notice that you must: first understanding the priority levels to understand the thread\_scheduling algorithms.
- Priority Levels: Windows have 32 priority levels, extending from 0 through 31.
  - Sixteen real-time levels 16 to 31.
  - Fifteen variable levels 1 to 15.
  - One system level (0) is reserved for the zero page thread. .
- When a process has only a single base priority value in which each thread has two priority values: current and base. Scheduling decisions are make based on the current priority, the system under certain conditions
- increases the priority of threads in the dynamic range (1 to 15) for brief periods,



Windows never modifies the priority of threads in the real-time range (16 to 31), so they always have the same base and current priority.

- A thread's initial base priority is inherited from the process base priority. A process, default, inherits its base priority from the process that made it.

#### Memory allocation algorithms

##### UNIX:

- Memory Allocation Algorithms in Swapping:
  - Swapping is bringing each process in physical memory completely and running it. When the process is no longer used, the process will be terminated or is swapped out to disk.
  - In which linked lists is used in keeping track of memory usage.
- The allocated memory segments have processes that be located in currently memory, and the free memory segments are empty holes that's between two allocated segments
- With one list, the segment list is sorted by address and each entry in the list can be set allocated or empty with a bit marker.
- With two separate lists one for allocated and another for free memory segments each entry in the lists have the address and size of the segment, and two pointers. One pointer for the next segment in the list; the other for the last segment in the list.
- Algorithms used to allocated memory for a new process or an existent process that swap in, in which scan the list of free memory segments and after allocated segment, the chosen segment is transferred from the free to allocated list.



- ✓ First-fit : scans the list of memory segments from the begin till it find the first segment which is big enough to hold the process. selected free segment is tried to split into two pieces. One piece is enough for the process. If the rest piece is greater than or equal to the minimum size of free segments, one piece is allocated to the process and the other segment rests free. If the rest piece is less than the minimum size of free segments, the whole segment is allocated to the process without divided. The list entries are updating.
- ✓ Next-fit algorithm: works same as first-fit except in next time it is called to look for a free segment, it starts scanning from the place where it stopped last time. so, it has to record the place where it finds the free segment every time. And when the searching reaches the end of the list, it goes back to the beginning of the list and continue.
- ✓ Best-fit algorithm. It searches the whole list, takes the smallest free segment that 's enough to holding the process, allocates it to the process, and updates the list entries.
- ✓ Quick-fit algorithm. It has different lists of the memory segments and puts the segments with the same level of size into a list.

➤ Page Replacement Algorithms in Demand Paging:

- For demand paging, only a number of pages of a process be located in in memory

When some page has to be referenced but not brought in memory yet, it is called a page fault. When a page fault occurs that's mean no room available in memory, the memory management system should choose a page to be remove from memory to



make room for the page that has to be brought in. In a virtual memory system, pages in main memory may be either clean or dirty so there's two cases

- A. If page is clean, the page hasn't been modified and will not be rewritten to the disk when it is removed from the memory because the disk copy is the same as the one in memory
- B. If page is dirty, the page has been changed in memory, and must be rewritten to the disk if it is removed from the memory. Then the new page to be read will be brought in memory and overwrite old page.

Algorithms: to know how good a page replacement algorithm is?! We should examine how

Frequent the thrashing happens when the algorithm is used.

- ✓ The optimal page replacement algorithm: It is an ideal algorithm, and of no use in real systems. But usually, it can be used as a basis reference for other realistic algorithms. It removes the most optimal page that will be referenced the last among processes presently in memory. But it is difficult and costly to look for this page.
- ✓ The first-in-first-out (FIFO) page replacement algorithm: It removes the page that is in memory for the longest time amongst all the processes presently exist in in memory. The memory management system can use a list to maintain all pages presently in memory, and put the page that arrives the most recently at the end of the list. When a page fault happens, the first page in the list( the first comer), is removed. The new page is the most recent comer, so it is put at the end of the list.
- ✓ The least recently used (LRU) page replacement algorithm: It assumes that pages that haven't been used for a long time in the past it will remain unused for a long time in the future. Thus, when a page fault occurs, the page unused for the longest time in the past will be removed.

To implement LRU paging, it is essential to have a linked list of all pages in memory. When a

page is referenced, it is put at the end of the list. Thus, the head of the list is the least in recent



times used page, which will be chosen as the one to be removed when a page fault happens

- ✓ The clock page replacement algorithm. It puts all the pages in memory in a circular list and maintains the list in a clock-hand-moving order ,Regularly, the virtual memory in a computer system has a status bit associated with each page,.
- ✓ The revised clock page replacement algorithm. If the virtual memory management in a computer system has two status bits linked with each page, one for reference (R) and the other for modification (M). That is, R is set when the page is referenced, and M is set when the page is modified(written or dirty). The bits can be kept in check in each page table entry. If the hardware does not have these bits, they could simulate with a data structure. Same as check page replacement algorithm, when a page fault occurs, the system start its page scanning along the clock-like list. If both of its R and M bits are zero ,the page is chosen to remove, and replaced with the new page. The scanning pointer moves to point to next page in the list and stops there until the next page fault.

## Windows

- Windows can access up to 4GB of physical memory on 32 bit
- It's have its own 4GB logical address space allowed to each process
- It's kernel mode have upper 2GB .
- It's user mode have lower 2GB of the address space
- It's uses cluster demand paging n which:
- when page needed are brought in the memory
- eight pages are brought in the memory all together.



- It use of working set model that's mean the amount of memory presently assigned to the process.
- the pages that are set in the main memory.
- Size of working set can be changed in view of that.

Address of windows is divided into :

1-page number

2-offset number

Windows uses First in First out Page Replacement Algorithm in which

- Old page is chose by system to replacement.
- increase number of frame may increase page fault rate
- may have low performance

Windows divide virtual pages (Modified – Stand\_by- Free Page -Zeroed )Page List .

Pages that have been modified put in modified page list. they are stored in standby list. After they writing to disk Pages that have'nt modified go to stand by list. The modified page list is called dirty list while standby list is called clean list.



# 02

## *Project*

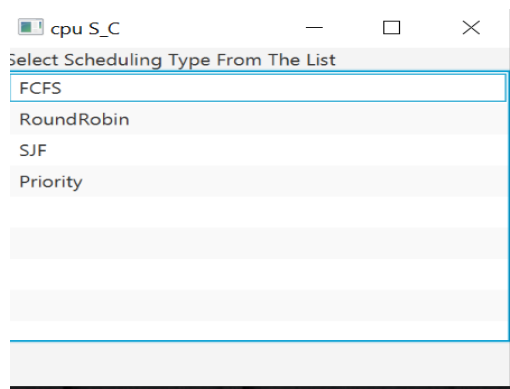
### Scheduler project:

Every snapshot the explanation is below it.

My part is: GUI and FCFS coding and Scheduler class coding and relate GUI with classes of project:



- this is welcome page when user select start the Scheduler button the set on action function is run the list of option stage as in fig below:
- this stage (page) can from it select the algorithm you want when selected it set on function run





```
Label l1=new Label("Select Scheduling Type From The List");
Label l2 = new Label("");
ObservableList<String> account=
    FXCollections.observableArrayList(
        "FCFS", "RoundRobin", "SJF", "Priority"
    );
ListView<String> ivw=new ListView<String>(account);
// جزء الخاص باختيار من الالست
MultipleSelectionModel<String> imode=ivw.getSelectionModel();
imode.selectedItemProperty().addListener(new ChangeListener<String>(){
    @Override
    public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

        Stringalog=newValue;
        l2.setText("The Selected Scheduling Type is: "+newValue);
    }
});
```

Next stage as in fig upper is the code of making list in javaFX

- the observable-list class is responsible for represent the options in list.
- The multiple-selection-model class is responsible for get selected option from list.
- Changed function is responsible for run code in it every selected on list is changed.

```
Text defin(int i,String s,Map t1,Map t2){
    Text t = new Text (20, i, s);

    t.setOnMouseClicked(e->{

        System.out.println( t.getText());
        System.out.println(t1.get(t.getText()));
        t.setText(s+" (" +t1.get(t.getText())+" ", "+t2.get(t.getText())+" )");

    });
    t.setOnMouseMoved(e->{

        System.out.println( t.getText());
        System.out.println(t1.get(t.getText()));
        t.setText(s);

    });

    return t;
}
```

- Defin function is responsible for set the text “name of process ” at the vertical line of Giant chart
- Set-on-mouse-clicked function is responsible for represent start and end time of process  
When click on process text
- Set-on-mouse-moved function is responsible for hiding start and end time of process

When moved mouse a way

```
Text definyscale(float i,String s,int co){  
  
    Text t = new Text (100+100*i,5020, s);  
  
    co=(co)%3;  
    if(co==0){  
        t.setFill(Color.BURLYWOOD);  
    }  
    if(co==1){  
        t.setFill(Color.RED);  
    }  
    if(co==2){  
        t.setFill(Color.BLUE);  
    }  
  
    return t;  
}
```

- Defin-scale function is responsible for represent text of time of process According to coordinates that set to function
- Co variable is responsible for change color of text According to number of Process.

```
Line definli(int t,float t11,float t22,int co){  
  
    Line rawl=new Line(100+t11*100,t,100+t22*100,t);  
  
    co=co%3;  
    if(co==0){  
        rawl.setStroke(Color.BURLYWOOD);  
    }  
    if(co==1){  
        rawl.setStroke(Color.RED);  
    }  
    if(co==2){  
        rawl.setStroke(Color.BLUE);  
    }  
    rawl.setStrokeWidth(8);  
  
    return rawl;  
}
```

- Definli function is responsible for represent of line start from start time of process to end time of process that pass to function (t11andt22)

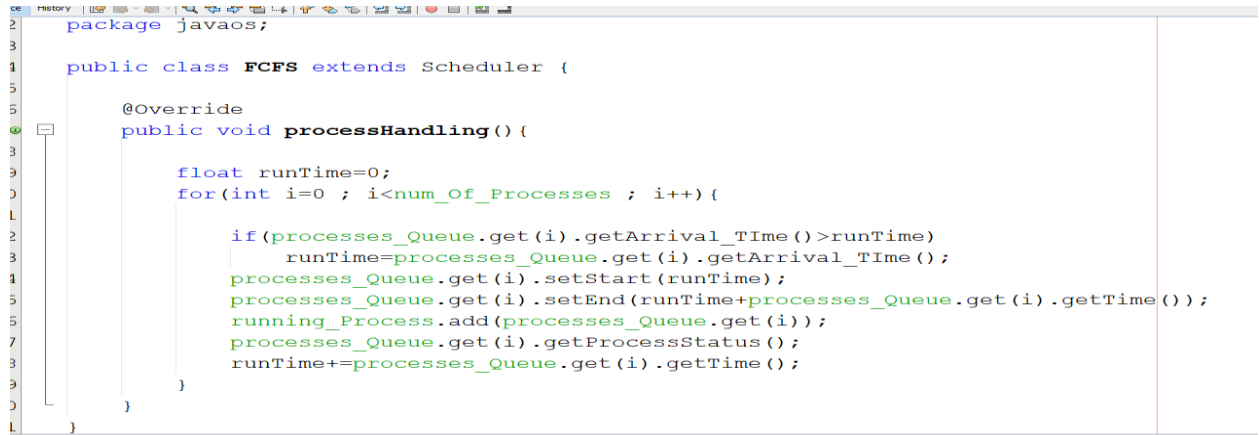
```

1      Text texlin(int i,String S,float t11,float t22){

      Text t=new Text((100+(t11)*100)+10,i,S);
      t.setFill(Color.BLACK);
      return t;
-    }

```

- Textlin fuaction is responsible for represent of text of process on line that discuss above according to coordinates t11->start time. t22->end time.



```

2      package javaos;

3
4      public class FCFS extends Scheduler {
5
6          @Override
7          public void processHandling() {
8
9              float runTime=0;
10             for(int i=0 ; i<num_Of_Processes ; i++){
11
12                 if(processes_Queue.get(i).getArrival_Time()>runTime)
13                     runTime=processes_Queue.get(i).getArrival_Time();
14                 processes_Queue.get(i).setStart(runTime);
15                 processes_Queue.get(i).setEnd(runTime+processes_Queue.get(i).getTime());
16                 running_Process.add(processes_Queue.get(i));
17                 processes_Queue.get(i).getProcessStatus();
18                 runTime+=processes_Queue.get(i).getTime();
19             }
20         }
21     }

```

- Logic of class: the process is come first is enter cpu is serve first and come second and third and so..on is wait in running state queue until first is finish.
- Process-queue is arraylist of type process class which have data of Process.
- So I implemented that as ->> for is looping on each process according to arrival time and see if arrivelttime is greater than runtime( which is store time of process is run in Cpu in order) then runtime is store is new value of start of new process and add in Process-queue else runtime is same,
- Function setstart() is set the start of each process in order in Process-queue,
- Function setend() is set the end of each process in order in Process-queue,
- Running – queue is arraylist of type process class, which have order of Process
- Function add() is set number of process in order in Running – queue.
- Function getprocesstatus () calculating wait time.
- Runtime is then update with sum of start time is in old runtime and burst time that user enter by gettime() to have endtime.
- Processhandling() is override in FCFS and SJF and RR and priority classes

To handel the Process.



```
package javaos;

import java.util.ArrayList;

public abstract class Scheduler {

    String[] scheduler_Type = {"FCFS", "SJF_P", "SJF_NP", "PERIORITY_P", "PERIORITY_NP", "RR"};
    int num_Of_Processes;
    float time_Quantum;
    float avgWaitingTime = 0;
    ArrayList<Process> processes_Queue = new ArrayList<>();
    ArrayList<Process> running_Process = new ArrayList<>();
    boolean preemptive;

    public abstract void processHandling();
}
```

- Scheduler class is abstract class will deal in main function in which have :
  - Scheduler\_type variable that have type of Scheduler.
  - Time-Quantum variable that have Quantum time.
  - Avgwaitingtime variable that have Average waiting time.
  - Preemptive is Boolean variable to check Preemptive or not.
  - Processhandling() is define in that as abstract function.

Example on FCFS in GUI:

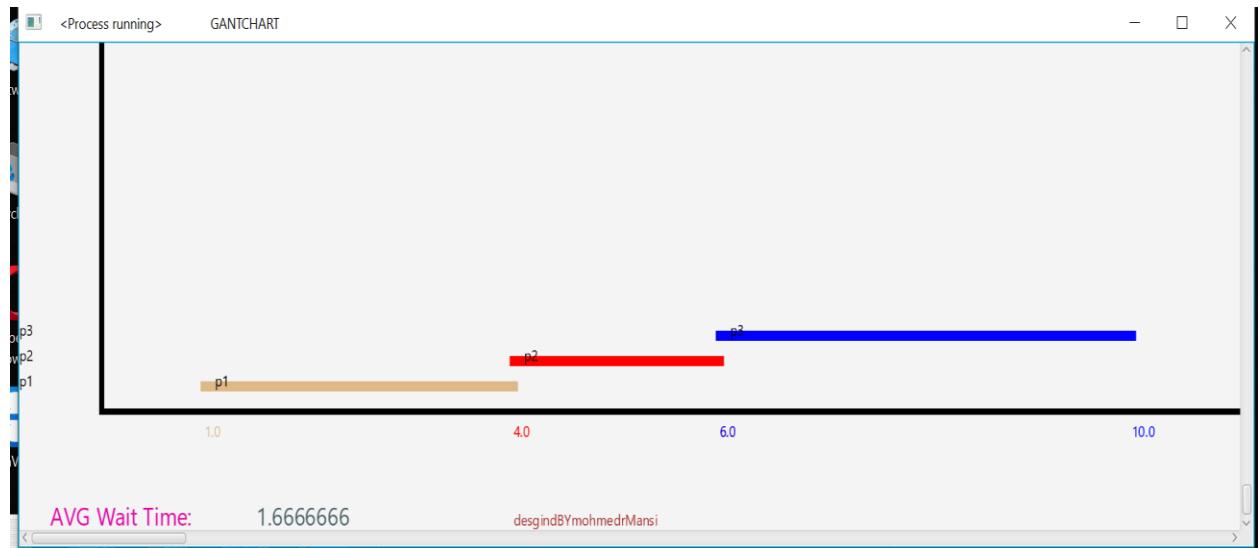
Input->



The screenshot shows a window titled "<<FCFS>>" with a black background. It contains three input fields: "Number Of Processes" with the value "3", "Arrival Time" with the value "1 2 3", and "Time Of The Process" with the value "3 2 4". Below these fields, the text "Valid Input" is displayed in red. An "ENTER" button is located to the right of the input fields. A small watermark "WALLPAPERBUD.COM" is visible in the bottom right corner of the window.

- Valid input is show due to checking on inputs

Output->



Now we will have some of code of GUI :

```
float[] arrivalTime = new float[no_ofprocess];
float[] processTime = new float[no_ofprocess];
float[] priorityProcessArray = new float[no_ofprocess];

arrivalTime = stringtoint(no_ofprocess, arivelltime);
processTime = stringtoint(no_ofprocess, process);
priorityProcessArray = stringtoint(no_ofprocess, pitorty);

Process[] p = new Process[no_ofprocess];
int flagdraw=0;
for(int i=0 ; i<no_ofprocess ; i++){

    if(!alog.equals("Priority"))p[i]= new Process(processTime[i] , arrivalTime[i] , ""+(i+1));
    else p[i] = new Process(processTime[i] , (int)priorityProcessArray[i] , arrivalTime[i] , ""+(i+1));
}
sort(p , 0 , p.length-1);
int flag_0=0;
float[] ttt1 = new float[100]; //{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20};
float[] ttt2 = new float[100]; //{3,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23};
int[] pp= new int[100]; //{1,2,3,4,1,3,4,2,5,6,7,9,8,10,11,12,13,14,15,16};
Map< String,Float> t_1 =
```

- Stringtoint() function change input is in string into int after cut it and put it in array.
- Process is array of type Process class
- For loop is used set inputs in process class as [ processtime , arrivelttime, process number ] if the algorithm is not priority but if priority we add priority of process to last  
Variable [ processtime , arrivelttime, priority ]
- ttt1 is array will use in giant chart that have start time of process



- ttt1 is array will use in giant chart that have end time of process
- pp is array will use in giant chart that have name of process

```
FCFS fcfs = new FCFS();
SJF sjf = new SJF();
RR rr = new RR();
Priority priority = new Priority();
int rrsz=0;
float wait=0;
// "fcfs", "Roundrobin", "sjf", "priority"

if (alog.equals("FCFS")) {

    fcfs.num_Of_Processes=p.length;

    for(int i = 0 ; i<p.length; i++){
        fcfs.processes_Queue.add(p[i]);
    }

    fcfs.processHandling();
}
```

- in first four line of that code is define data-type of algorithm  
uses to fill in number of Process in processes\_queue array and then  
will use to call Processhandling()

```
for(int i = 0 ; i<fcfs.running_Process.size(); i++){

    if (i<fcfs.num_Of_Processes)
        fcfs.avgWaitingTime+=(p[i].getWaitingTime());
}
fcfs.avgWaitingTime=fcfs.avgWaitingTime/fcfs.num_Of_Processes;

wait=fcfs.avgWaitingTime;
```

- here is calculating average wait time of each process by function getwaittime()
- then divided it by number of process to variable of avgwaitingtime of type fcfs in scheduler abstract class then set in variable wait that will use in giant chart.



```
for(int f=0;f<fcfs.running_Process.size();f++){  
    // System.out.println("hello in for before");  
    ttt1[f]=fcfs.running_Process.get(f).getStart();  
    // System.out.println("hello in for after");  
  
    ttt2[f]=fcfs.running_Process.get(f).getEnd();  
    int temp2=Integer.parseInt(fcfs.running_Process.get(f).getProcess_Number());  
    pp[f]= temp2;  
    System.out.println(ttt1[f]);  
    System.out.println(ttt2[f]);  
    System.out.println(temp2);  
  
}  
  
rrsize=fcfs.running_Process.size();  
}
```

In this part will fill :

- array tt1->start time of Process
- array tt2->end time of Process
- array pp-> Process number
- get size of running\_process by size() and set it in rrsz variable



```
}  
if(extraprocess==-1&&flagdraw==1){  
for(int i=1;i<=rrsize;i++){  
    cc3=cc3-20;  
    xx=i-1;  
    String str1 = Integer.toString(pp[xx]);  
    String ss2="p"+str1;  
    root.getChildren().add(defin(cc3,ss2,t_1,t_2));  
  
    float t111=0;  
    float t222=0;  
    if(flag_==0){  
  
        // t_1.put(ss2, new Float(ttt1[xx]));  
        // t_2.put(ss2, new Float(ttt2[xx]));  
        t111=ttt1[xx];  
        t222=ttt2[xx];  
        flag_=1;  
  
    }  
    else if(flag_==1){  
        // t_1.put(ss2, new Float(ttt1[xx]));  
        // t_2.put(ss2, new Float(ttt2[xx]));  
        t111=ttt1[xx];  
    }  
}
```

In this part will explain how giant-chart is drawing :

- extraprocess flag check if there is an extra process.
- drawflag check if the input is correct or not to show giant-chart or not.
- XX variable have number of process in every iteration of for loop
- Ss2 variable have name of process in every iteration of for loop
- Root,getchildren().add() is function that add text of name process Through defin()function.
- T111 variable store in it start time in every iteration of for loop
- T222 variable store in it end time in every iteration of for loop.

```
else if(flag_==1){
    // t_1.put(ss2, new Float(ttt1[xx]));
    // t_2.put(ss2, new Float(ttt2[xx]));
    t111=ttt1[xx];
    t222=ttt2[xx];
}

String tt1=t111+"";
String tt2=t222+"";
root.getChildren().add(definscale(t111,tt1,xx));
root.getChildren().add(definscale(t222,tt2,xx));

root.getChildren().add(definli(cc3,t111,t222,xx));
root.getChildren().add ( texlin(cc3,ss2,t111,t222));
System.out.println(70+t111*20);
System.out.println(70+t222*20);
System.out.println(ttt1[xx]);
System.out.println(ttt2[xx]);
}
```

- Tt1 variable string change start time from int to string get from t111.
- Tt2 variable string change end time from int to string get from t222.
- Root,getParent().add() is function that add time of each process Through definscale()function we use it twice one for starttime by passing tt1 and t111 And one for end time by passing tt2 and t222.
  - Root,getParent().add() is function that add line of each process Through definli()function.
  - Root,getParent().add() is function that add text of each process Through textlin()function.

```

        root.getChildren().addAll(raw,colum,au,tm,waittimename, waittime);

ScrollPane scrollPane = new ScrollPane();
scrollPane.setFitToHeight(true);
scrollPane.setFitToWidth(true);
scrollPane.setContent(root);

waittimename.setOnMouseMoved(e->{
//   Colour color=new   Colour();
waittimename.setFill(Color.color(Math.random(),Math.random(),Math.random()));
waittimename.setFont(Font.font(20));

    waittime.setFill(Color.color(Math.random(),Math.random(),Math.random()));
    waittime.setFont(Font.font(20));

});

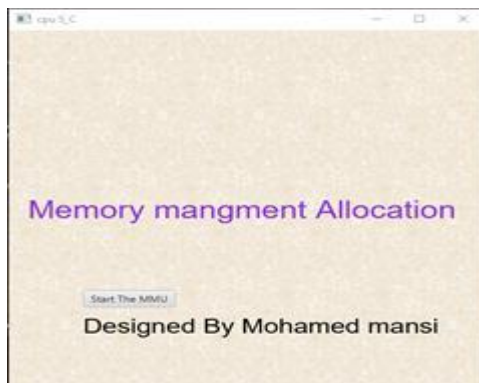
```

- Root,getchildren().addall() is function that add wait time and column(vertical) line  
Raw(horizontal) line to giant chart in same stage with same root  
We use scrollane to make to giant chart more fixable in show.

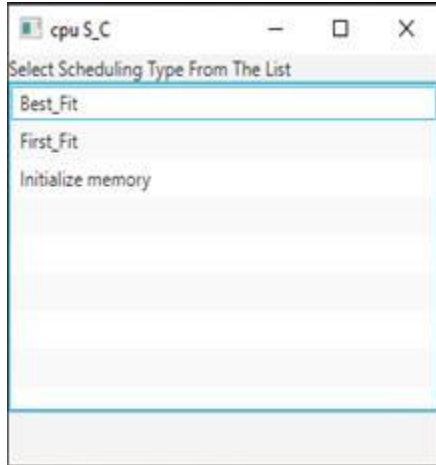
MM-Allocation-project:

My part is: segment or hole class and GUI coding and relate GUI with classes of project:

GUI general description:



- This is the welcome page that when select the start MMU button  
The setonaction() function in program is run the list code that is  
Explain in scheduler project part to select which option you want  
As below fig



- Here you must select in first the initialize memory option to Make initialize to the memory to fill memory with holes and See distribution of hole and reserved address space for initialize process As two fig below.
- Second you can select from two either options to do what you want From this algorithms

<<Initialize memory>>

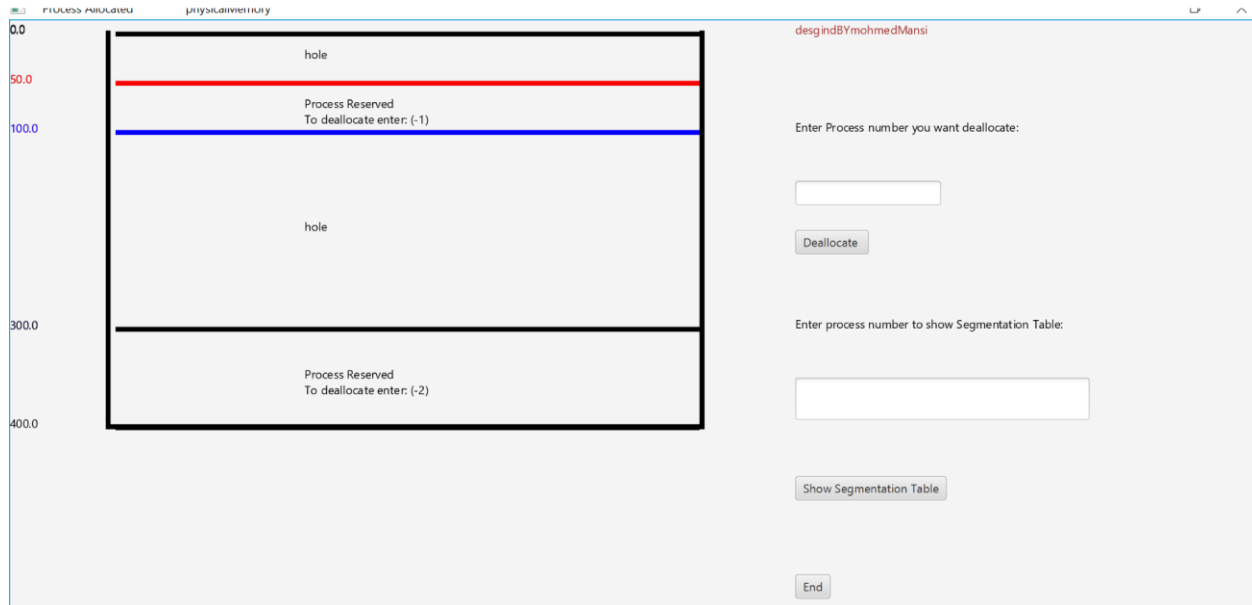
Total memory size: 400

Holes starting address: 0 100

hole size: 50 200

Number of holes: 2

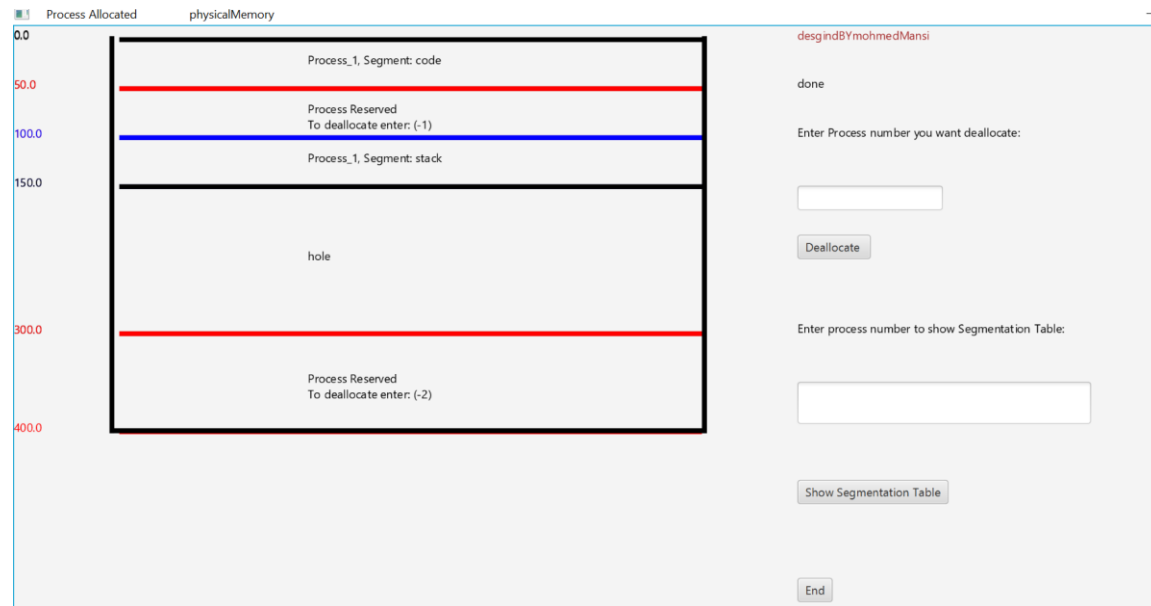
ENTER



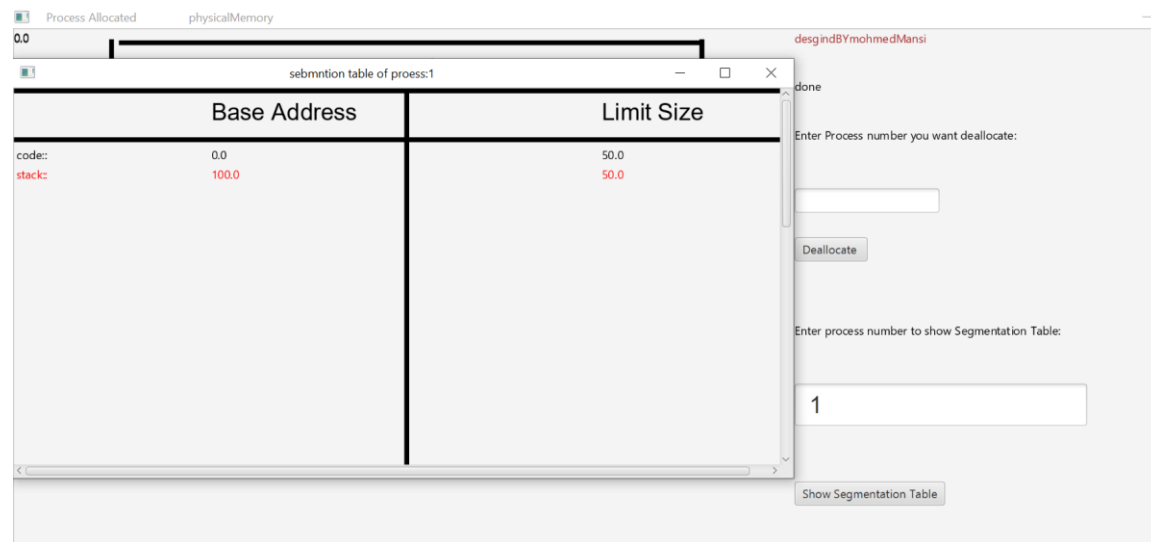
Then:

The screenshot shows a 'First Fit' memory allocation simulation interface. The title bar reads '<<First\_Fit>>'. The background is a light beige color with a pattern of small white stars. There are four input fields with labels to their left: 'Number of segments' with a value of '2', 'Size of segments' with a value of '50 50', 'Process number' with a value of '1', and 'Name of segments' with a value of 'code stack'. Below these fields is a blue button labeled 'ENTER'.

- After select first\_fit algorithm and initialize memory that window is Come to enter the data.



- This window show memory after doing first\_fit algorithm the done message is Refer to segmentation fit is done successfully to all segment of process





- Here when you enter process number segmentation table is show

Fig delete process by number

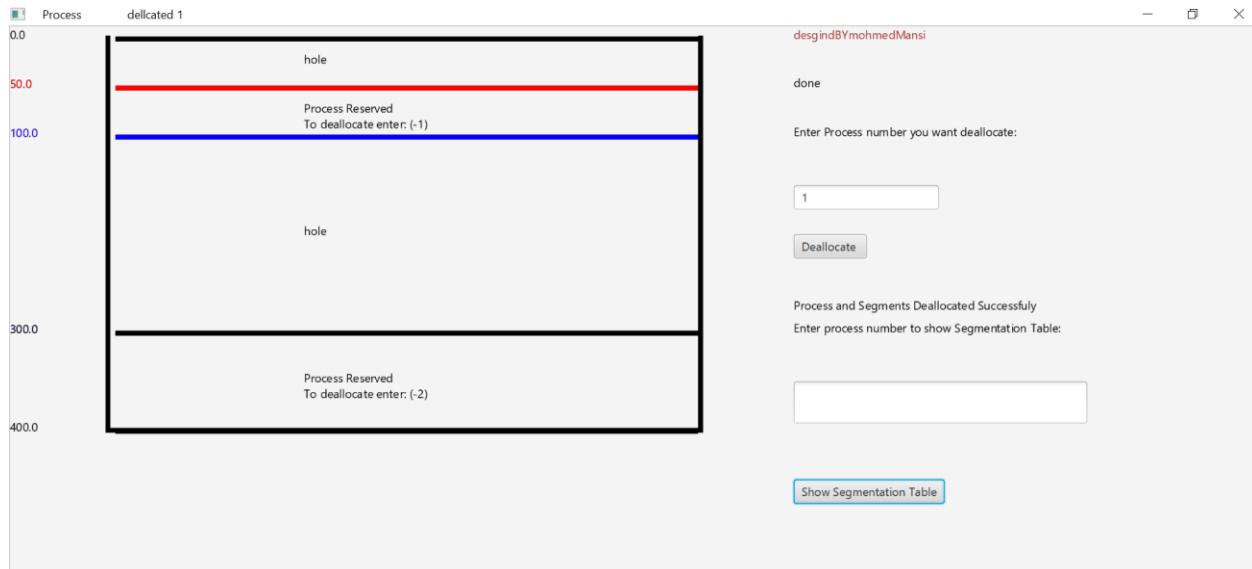
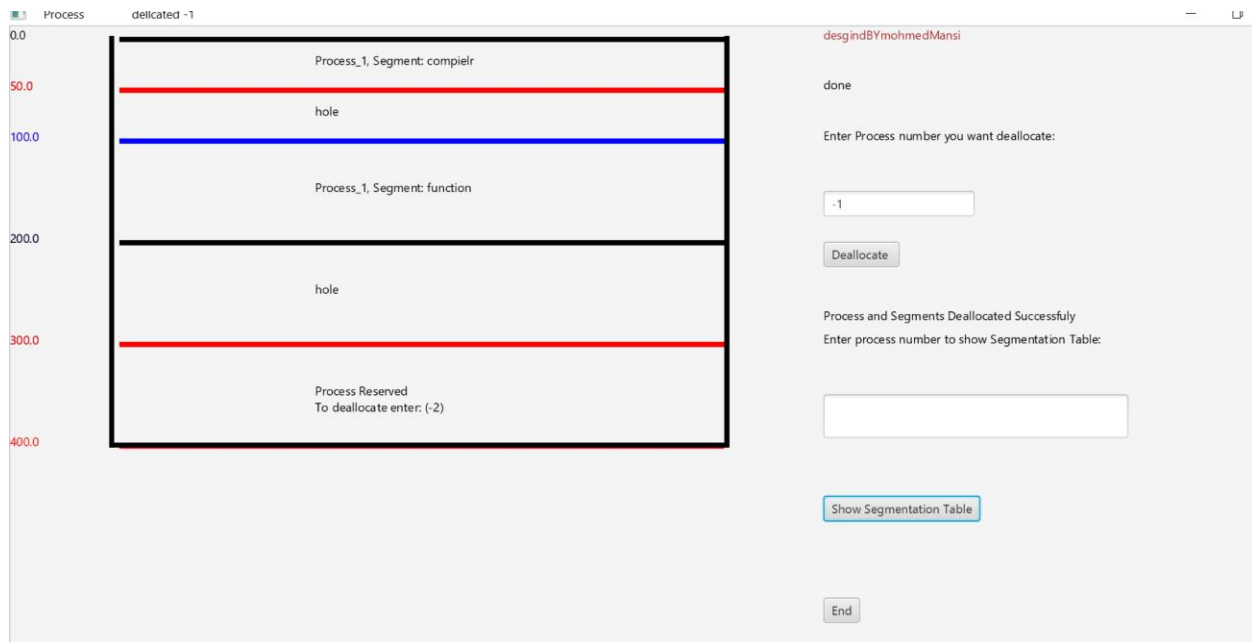


Fig delete process reserved





- Here in two fig above you can delete process by enter process number or segmentation name
- Here in two fig above you can delete process reserved by each number in negative number  
Example process reserved (1) is -1 as in fig

Here another example to best fit algorithm:

<<Best\_Fit>>

Number of segments

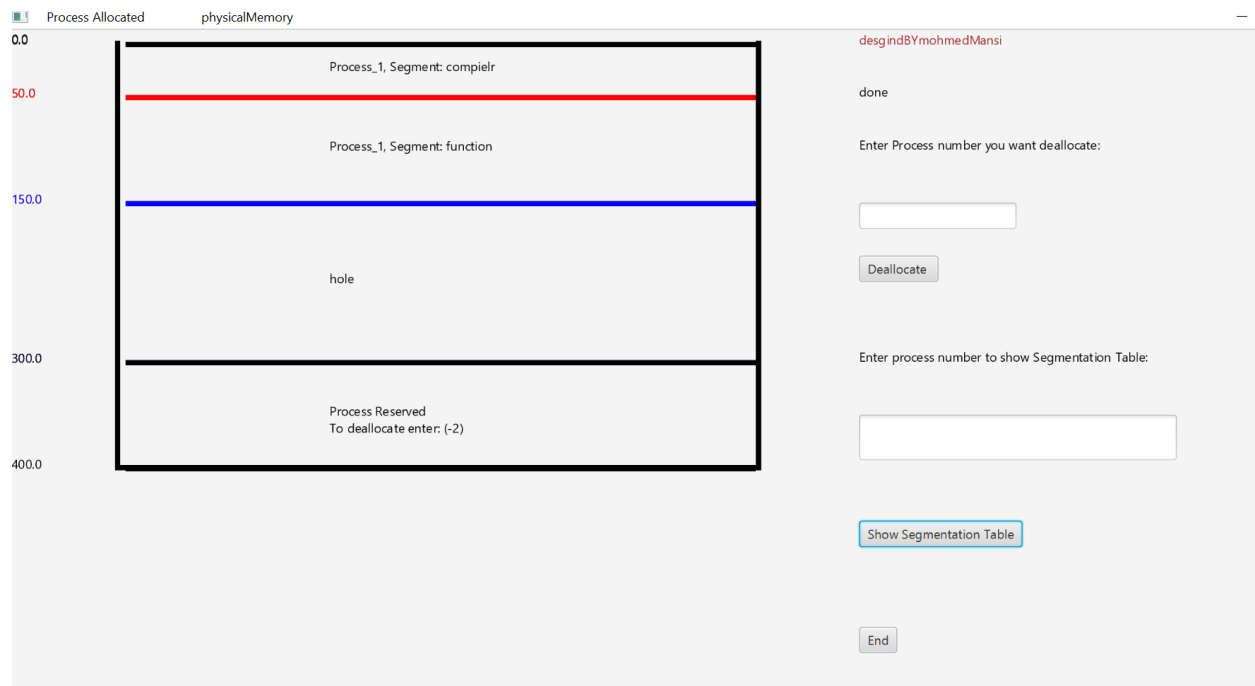
Size of segments

Process number

Name of segments

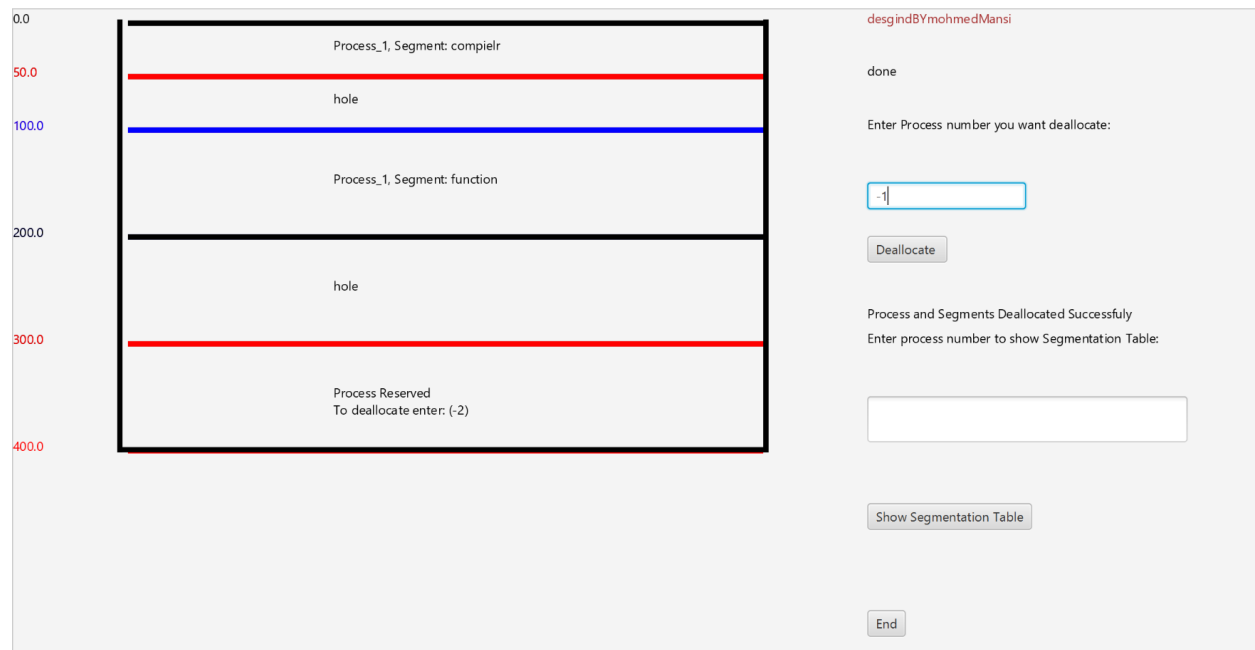
ENTER





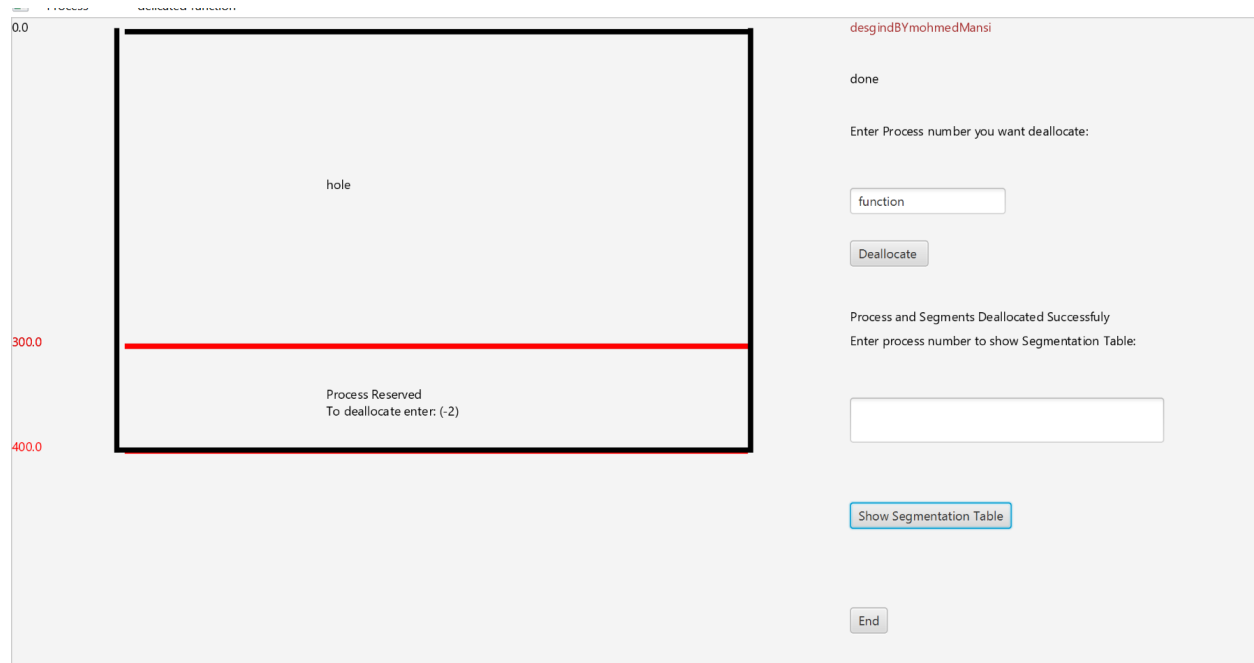
Delete process by segmentation name in two below fig:

Before :

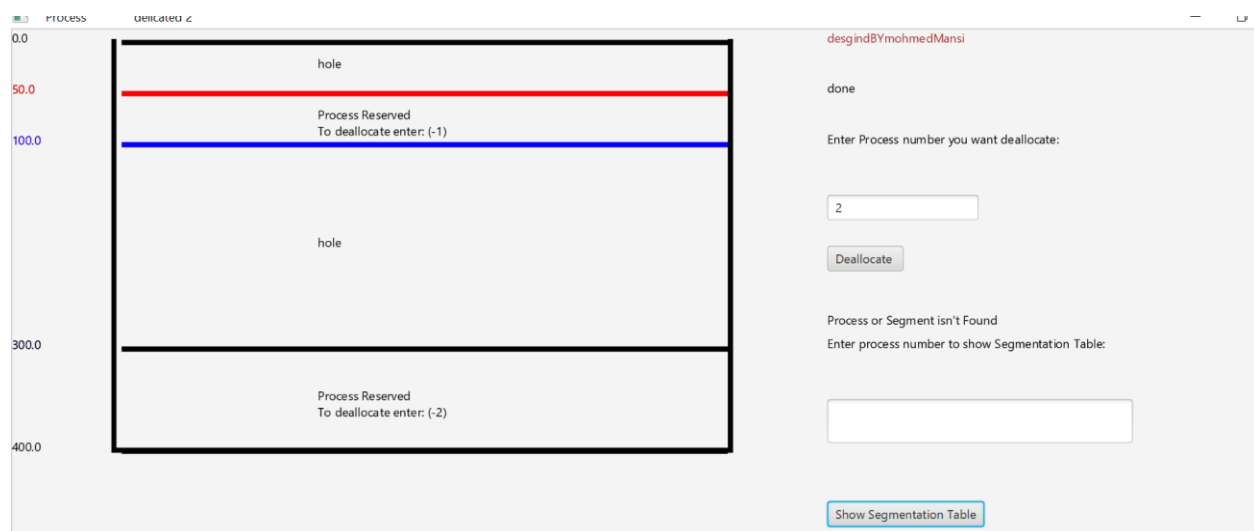




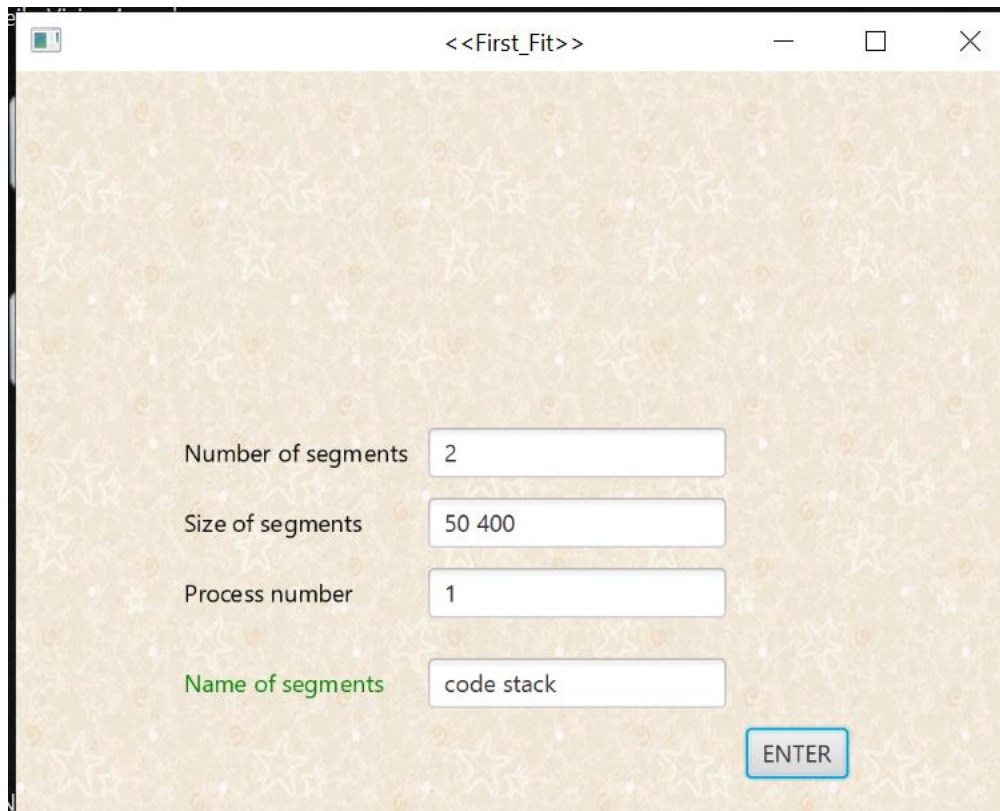
after:



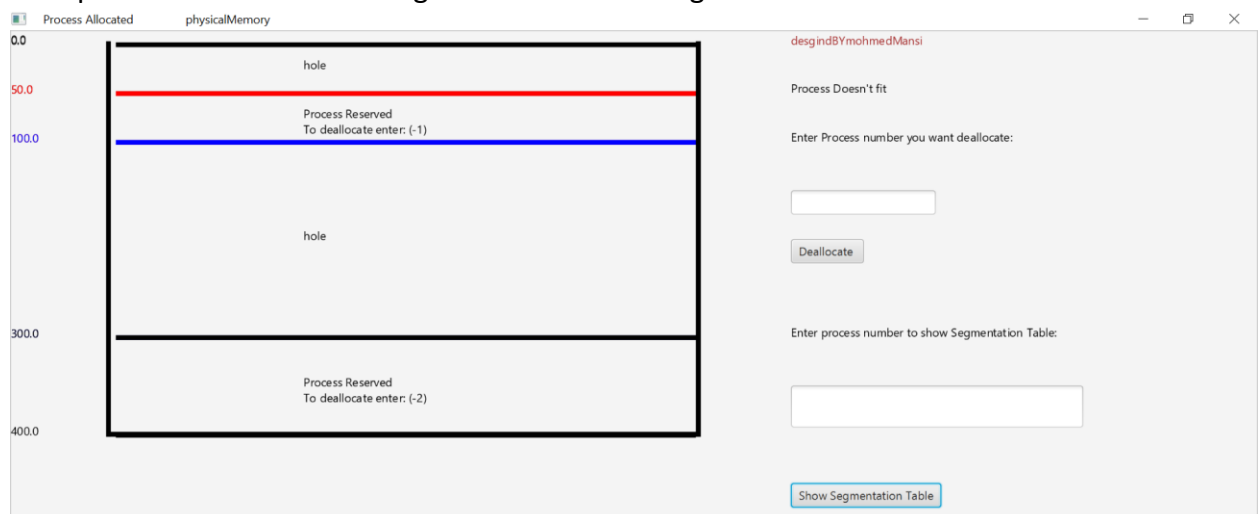
The here explain error message:



- Here error process or segment isn't fit message when there no process number or segment name found to delete



- Here enter process have segment have large size not fit  
So process doesn't message is show below in fig:





Here in this part we will explain the code:

First segment or hole class:

1-

```
public class SegmentOrHole implements Comparable<SegmentOrHole> {  
  
    private String type;  
    private int processNumber;  
    private float holeStartingAddress;  
    private float segmentStartingAddress;  
    private String segmentsName;  
    private float segmentsSize;  
    private float holeSize;  
    public static boolean trickyCompare = false;  
  
    public SegmentOrHole(float holeStartingAddress, float holeSize) { //hole constructor  
        this.type = "hole";  
        this.holeStartingAddress = holeStartingAddress;  
        this.holeSize = holeSize;  
    }  
  
    public SegmentOrHole(int processNumber, String segmentsName, float segmentsSize) { //segment constructor  
        this.type = "segment";  
        this.processNumber = processNumber;  
        this.segmentsName = segmentsName;  
        this.segmentsSize = segmentsSize;  
    }  
}
```

2-



```
public String getType() {  
    return type;  
}  
  
public int getProcessNumber() {  
    return processNumber;  
}  
  
public float getHoleStartingAddress() {  
    return holeStartingAddress;  
}  
  
public float getSegmentStartingAddress() {  
    return segmentStartingAddress;  
}  
  
public String getSegmentsName() {  
    return segmentsName;  
}  
  
public float getSegmentsSize() {  
    return segmentsSize;  
}
```

3-

```
public float getHoleSize() {  
    return holeSize;  
}  
  
public void setType(String type) {  
    this.type = type;  
}  
  
public void setHoleStartingAddress(float holeStartingAddress) {  
    this.holeStartingAddress = holeStartingAddress;  
}  
  
public void setSegmentStartingAddress(float segmentStartingAddress) {  
    this.segmentStartingAddress = segmentStartingAddress;  
}  
  
public void setHoleSize(float holeSize) {  
    this.holeSize = holeSize;  
}  
  
public void setSegmentsSize(float segmentsSize) {  
    this.segmentsSize = segmentsSize;  
}
```

incharfml tava | tmana tava | ElabrvitStream tava

4-

```
public float getEndAddress() {  
    if (type.equals("hole"))  
        return this.holeStartingAddress + this.holeSize;  
    else  
        return this.segmentStartingAddress + this.segmentsSize;  
}
```

- In this class I made is holding every data for segment and hole and define the type of If that hole or segment in private string type variable .
- The constructor of the class segment or hole is used depend on When you instances the class one for hole and one for Segment according to their type to initialize that data [holeStartingAddress-holeSize] data of hole constructor and [processNumber-segmentsName-segmentsSize] data of segment constructor .
- getType() is return the type if that hole or segment.
- getProcessNumber() is return the process number which need know process which I need to de allocated if I want.
- getHoleStartingAddress() is return the Hole Starting Address which is used in anther classes.
- getSegmentStartingAddress() is return the segment Starting Address.
- getSegmentsName() is return the segment name which need for DE allocation process.
- getSegmentsSize() is return the segment size.
- getHoleSize() is return the segment size.
- setType() is set type in Type variable.
- setHoleStartingAddress() is set Hole Starting Address in HoleStartingAddress variable.
- setSegmentStartingAddress() is set Segment Starting Address in SegmentStartingAddress variable.
- setHoleSize() is set Hole Size in HoleSize variable.
- setSegmentsSize() is set Segments Size in SegmentsSize variable.
- getEndAddress() is return End Address that if type is hole will add Hole Starting Address and Hole size but if type is Segment will add Segment Starting Address

And Segment size then return it.

Second GUI code important part:

```
if(aalog.equals("Initialize memory")){  
    hssize= t3.getText().toString();  
    hsad=t2.getText().toString();  
  
    tmsize=Integer.parseInt(t1.getText().toString());  
    holenum=Integer.parseInt(t7.getText().toString());  
}
```

- here if user is select initialize memory option I will get for first time
  - 1-total memory size in tmsize variable by gettext() through t1 textarea .
  - 2- hole starting address in hasd variable by gettext() through t2 textarea .
  - 3- hole size in hsize variable by gettext() through t3 textarea .
  - 4- hole number in tmsize variable by gettext() through t7 textarea .
- then user go to the list to choose the algorithms then enter their data .

```
if(aalog.equals("Best_Fit")||aalog.equals("First_Fit")){  
    nuofseg=Integer.parseInt(t4.getText().toString());  
  
    pnu=Integer.parseInt(t6.getText().toString());  
  
    sizeofseg=t5.getText().toString();  
    nam_segmmention=t8.getText().toString();  
}
```

- after initialize memory and choose the algorithm [best or first fit] enter that data:
- notice that: this if statements is in the page after the list page in which this page is still runnable until you close program and either list page is runnable
  - 1-number of segment in nuofseg variable by gettext() through t4 textarea .
  - 2- size of segment in sizeofseg variable by gettext() through t5 textarea .
  - 3-process number in pnu variable by gettext() through t6 textarea .
  - 4- name of segmentation number in nam\_segmmention variable by gettext() through t8 textarea .

```

473
474         int flag_=0;
475
476         String []p_Of_seg=new String[100];
477         float [] t11 = new float[100]; //seg 1 start
478         float[] t22 = new float[100]; //seg 2 end
479         Map< String,Float> t_1 =
480             new HashMap< String,Float>();
481         Map< String,Float> t_2 =
482             new HashMap< String,Float>();
483
484
485
486         int cont=0;
487
488         /*
489         String []p_Of_seg=new String[100];
490         float [] t11 = new float[100]; //seg 1 start
491         float[] t22 =
492
493         */
494         int num_p=3;
495         float cc3=0;
496         int flagt1=0;

```

- string array that is name p\_of\_seg store name of each segment or hole or process reversed .
- float array that is name t11 store start addres of each segment or hole or process reversed .
- float array that is name t22 store end addres of each segment or hole or process reversed .

```

508         float[] segmentSize = stringtofloat(nuofseg , sizeofseg);
509         String[] segmentName = stringArraytoString(nuofseg, nam_segmmention);
510         String handlingCheck="";
511         for(int i = 0 ; i<nuofseg ; i++){
512             System.out.println("segmentSize["+i+"] = "+segmentSize[i]);
513             System.out.println("nam_Segmmention["+i+"] = "+segmentName[i]);
514         }
515         mmu.setAllocationMethodology(alog);
516         for(int i=0 ; i<nuofseg ; i++){
517             handlingCheck = mmu.handling(new SegmentOrHole(pnu, segmentName[i] , segmentSize[i]));
518             if(handlingCheck!="done") {
519                 mmu.deallocate(pnu);
520                 break;
521             }
522         }

```

- Here fist I define array name is segmentsize to get set segment size in order by function stringtofloat() but in float not string
- Here fist I define array name is segmentname to get set segment name in order by function stringtoatring() in string but divided or cut across the array
- Mmu class is responsible for handling the algorithm



- Setallocationmethodology() is function in MMU class is responsible for set algorithm which is  
Store in alog variable that get algorithm through list page by get text() function
- For loop is responsible for adding segment data to handling() for each segment
- In for loop if one of segment isn't fit then handling() don't return done in variable Handlingckcek so I deallocate the process.  
Else continue handling normally

```
523  
524     int iteratorMemory=0;  
525     for (SegmentOrHole s:mmu.getSegmentOrHoles()) {  
526         if (s.getType().equals("hole")) {  
527             t11[iteratorMemory]= s.getHoleStartingAddress();  
528             t22[iteratorMemory]= s.getEndAddress();  
529             p_of_seg[iteratorMemory]= s.getType();  
530  
531             System.out.println( t11[iteratorMemory]+": " +t22[iteratorMemory]+": "+p_of_seg[iteratorMemory]);  
532         } else {  
533             t11[iteratorMemory]= s.getSegmentStartingAddress();  
534             t22[iteratorMemory]= s.getEndAddress();  
535             p_of_seg[iteratorMemory]= "Process _"+s.getProcessNumber()+", Segment: "+s.getSegmentsName();  
536             System.out.println( t11[iteratorMemory]+": " +t22[iteratorMemory]+": "+p_of_seg[iteratorMemory]);  
537         }  
538         iteratorMemory++;  
539     }  
540 }
```

- Here I put all data of hole or data in for loop in arrays t11,t22,p\_of\_seg after handling so I  
Use this arrays to represent segmentation table and shape of memory in  
New stage
- If type is "hole" I set data of hole
- If type is "process"(normal or reserved ) I set data of process

```
int xx=0;
for(int i=1;i<mmu.getSegmentOrHoles().size()+1;i++){
    //cc3=cc3-20;
    String str1 = Integer.toString(i);
    String ss2="Segmention:"+str1;

    /// Text g=new Text(100,5090,defin2(cc3,ss2,t_1,t_2)+"");

    xx=i-1;
    float t111=0;
    float t222=0;
```

This is the above for loop:

- This for loop is used for represent shape of memory and segmentation table according to Segment or hole size

```
681
682     root.getChildren().add(definscale(t111,t11,xx,scale));
683     root.getChildren().add(definscale(t222,t2,xx,scale));
684     if(p_of_seg[xx].equals("hole")){
685         root.getChildren().add(defin(cc3,"hole",t_1,t_2,scale));
686     }
687     else if(p_of_seg[xx].charAt(8)=='-'){
688         root.getChildren().add(defin(cc3,"Process Reserved\nTo deallocate enter: (" +p_of_seg[xx].substring(
689     )
690     }
691     root.getChildren().add(defin(cc3,p_of_seg[xx],t_1,t_2,scale));
692     }
693     root.getChildren().add(definli(cc3,t111,t11,xx,scale));
694     root.getChildren().add(definli(cc3,t222,t2,xx,scale));
```

- First Definscale function is used to add starting address text
- second Definscale function is used to add ending address text
- And see if type is "hole" so add name of this part hole in a text through defin() function
- Else is process reserved known by "-" so add name of this part process reserved+ it's number in a text through defin() function
- Else add name of this part according to name of segment in p\_of\_seg array in a text through defin() function
- Then add line through definli() function one for starting address And other for ending address
  - Notice that: all of that is in above for loop and happen iteratively.



```
root.getChildren().addAll(raw,colum,au,down,intsize,finsize,dellocp,segtiontable,dellocp2,dellocp3,segtiontable);
ScrollPane scrollPane = new ScrollPane();
scrollPane.setFitToHeight(true);
scrollPane.setFitToWidth(true);
scrollPane.setContent(root);
int i=0;
```

- I used root class to add lines, textfield ,textarea, lables by function getchildren() and add() and addall() and add by used scrollpane class for all pages (windows) or(stages) for that project and scheduler project

#### References:

1- process scheduling algorithms\_ brief & memory allocation algorithms\_brief:

- <https://books.google.com.eg/books?id=VFV1DwAAQBAJ&printsec=frontcover&dq=Operating+System+Concepts-Wiley&hl=en&sa=X&ved=0ahUKEwiqxtOfwvLpAhVaQhUIHUV8CGYQ6AEIJjAA#v=onepage&q=Operating%20System%20Concepts-Wiley&f=false>

[ Silberschatz, A., Galvin, P., & Gagne, G. *Operating system concepts* ].

#### 2-Windows:

- <https://books.google.com.eg/books?id=WZdCAwAAQBAJ&printsec=frontcover&dq=Windows+Internals&hl=en&sa=X&ved=0ahUKEwjShK-6wflpAhWygHEKHRnfBegQ6AEIMDAB#v=onepage&q=Windows%20Internals&f=false> [Russinovich, M., Ionescu, A., & Solomon, D. (2012). *Windows internals*. Redmond, Wash.: Microsoft Press].
- <https://www.ukessays.com/essays/engineering/compare-the-memory-management.php?vref=1> [UKEssays. November 2018]



3-Unix:

- <https://books.google.com.eg/books?id=-Mq5ve5KHXQC&printsec=frontcover&dq=UNIX+Operating+System:+The+Development+Tutorial+via+UNIX+Kernel+Services&hl=en&sa=X&ved=0ahUKEwiglbSOWfLpAhXhoXEKHc6uCiwQ6AEIjAA#v=onepage&q=UNIX%20Operating%20System%3A%20The%20Development%20Tutorial%20via%20UNIX%20Kernel%20Services&f=false>

[ Liu, Y., Yue, Y., & Guo, L. (2011). *UNIX operating system*. Beijing: Higher Education Press ]