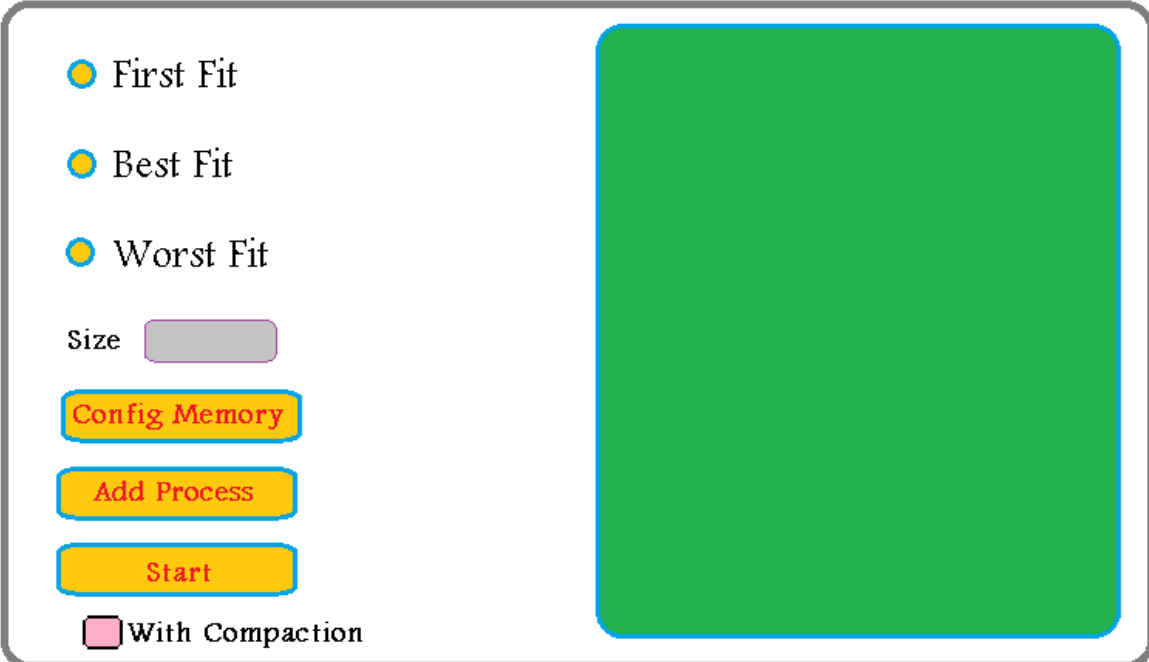


# OS Project 2 prototype

When the user first opens the program he see this window:

*Main\_Window*



The image shows a UI prototype for a memory management application. It features a title bar labeled 'Main\_Window' in a brown, italicized font. The window has a light gray border and contains several elements on the left side: three radio buttons for 'First Fit', 'Best Fit', and 'Worst Fit'; a 'Size' label followed by a gray input field; three yellow buttons with orange borders labeled 'Config Memory', 'Add Process', and 'Start'; and a checkbox labeled 'With Compaction'. On the right side of the window is a large green rectangular area, which is intended to display a list of segments.

☐ First Fit

☐ Best Fit

☐ Worst Fit

Size

☐ With Compaction

- The user should choose the allocation algorithm from the three radio buttons
- The green box on the left will show the list of segments.
- The size label shows the size of the memory
- The user can if memory compaction is needed by checking “With Compaction”

When the user clicks Config Memory he will see this:

### ***Config\_memory\_Dialog***



The Config\_memory\_Dialog interface is a rounded rectangle with a grey border. On the left side, there is a label "Size" followed by a yellow input field. Below this, there are two yellow buttons with blue borders: "Add Hole" and "Done". On the right side, there is a large green square with a blue border.

-The user can determine the size of memory.

When the user clicks Add Hole he will see this:

### ***Add\_Hole\_Dialog***

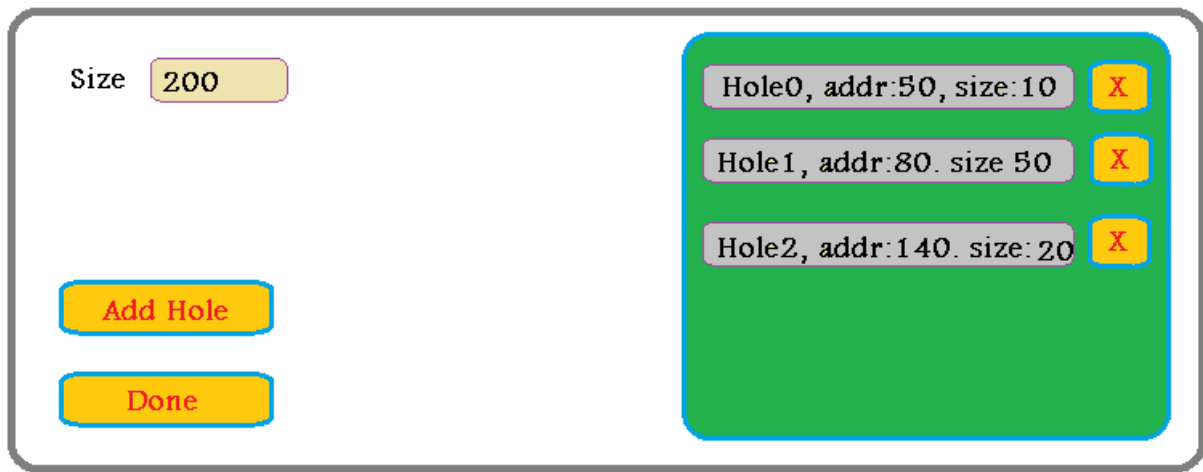


The Add\_Hole\_Dialog interface is a rounded rectangle with a grey border. It contains two labels: "Start Address" and "Hole Size". Each label is followed by a yellow input field. The "Start Address" field contains the value "140" and the "Hole Size" field contains the value "20". Below these fields is a yellow button with a blue border labeled "Add Hole".

-The user can specify the start address and size of the hole.

The user will return to Config Memory Dialog after he clicks "Add Hole".

## Config\_memory\_Dialog Example

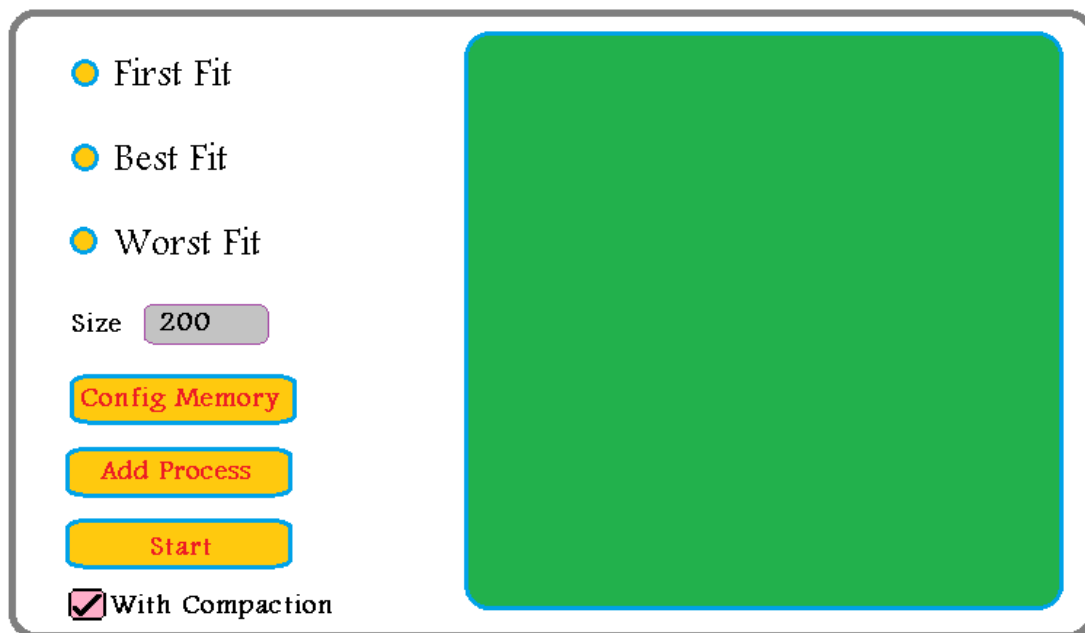


The dialog box has a title bar. On the left, there is a 'Size' label followed by a text input field containing '200'. Below this are two buttons: 'Add Hole' and 'Done'. On the right, there is a green rectangular area containing three entries, each with a delete button (a yellow square with a red 'X'):

Hole	addr	size	Delete
Hole0	50	10	X
Hole1	80	50	X
Hole2	140	20	X

-After the user has configured the memory ,he can return to the Main Window by clicking “Done”.

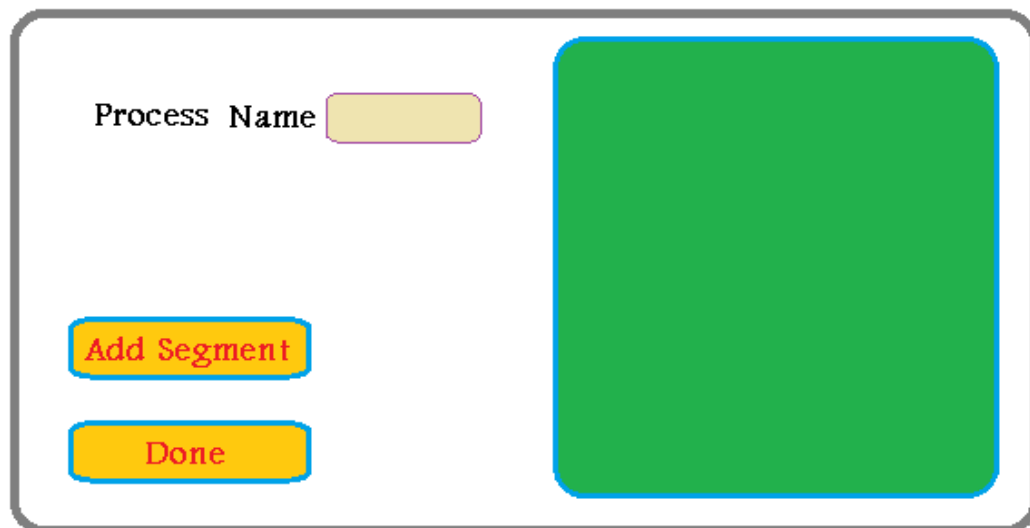
## Main\_Window Example



The main window has a title bar. On the left, there are three radio buttons for memory allocation algorithms: 'First Fit', 'Best Fit', and 'Worst Fit'. Below these is a 'Size' label followed by a text input field containing '200'. Further down are three buttons: 'Config Memory', 'Add Process', and 'Start'. At the bottom left is a checked checkbox labeled 'With Compaction'. On the right side of the window is a large green rectangular area representing the memory layout.

- The user can see that the Size label has changed to the value he had set in the Config Memory Dialog.
- The user can Add Processes by clicking “ Add Process” button and he will see this.

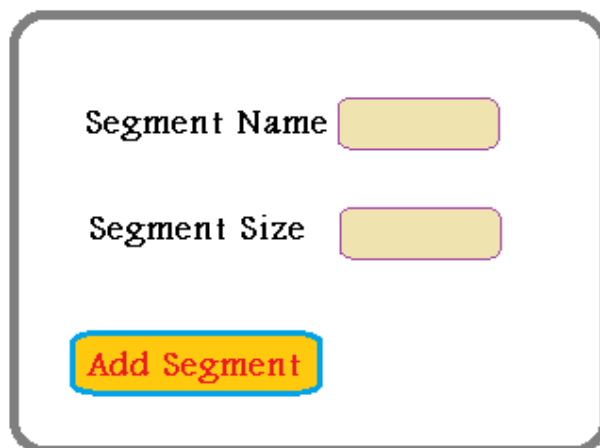
### ***Add\_Process\_Dialog***



The diagram shows a dialog box titled "Add\_Process\_Dialog". It contains a label "Process Name" followed by a text input field. Below the input field are two buttons: "Add Segment" and "Done". To the right of the input field and buttons is a large green rectangular area, likely representing a list of segments or a visual representation of memory.

- The user can define the process name.
- The user can add segments to the process by clicking “Add Segment”.

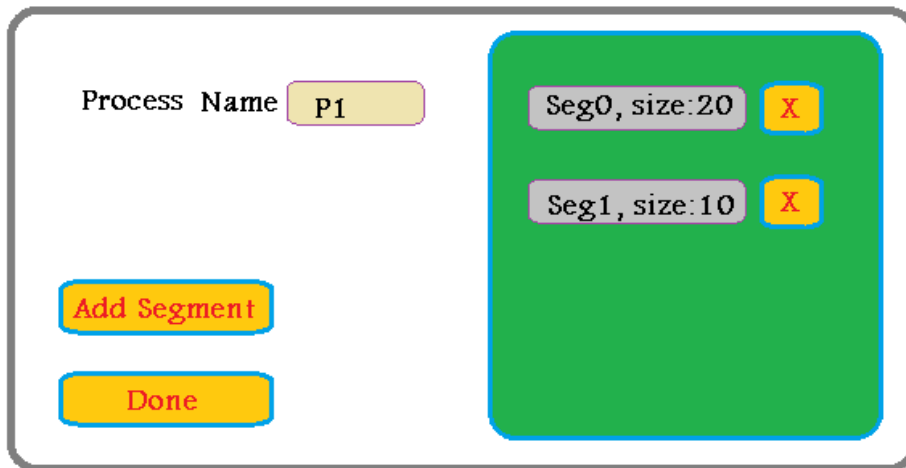
### ***Add\_Segment\_Dialog***



The diagram shows a dialog box titled "Add\_Segment\_Dialog". It contains two labels: "Segment Name" and "Segment Size", each followed by a text input field. Below the input fields is a button labeled "Add Segment".

After the user supplies the segment name and size and clicks “Add Segment” he will return to the process dialog.

### *Add\_Process\_Dialog Example*

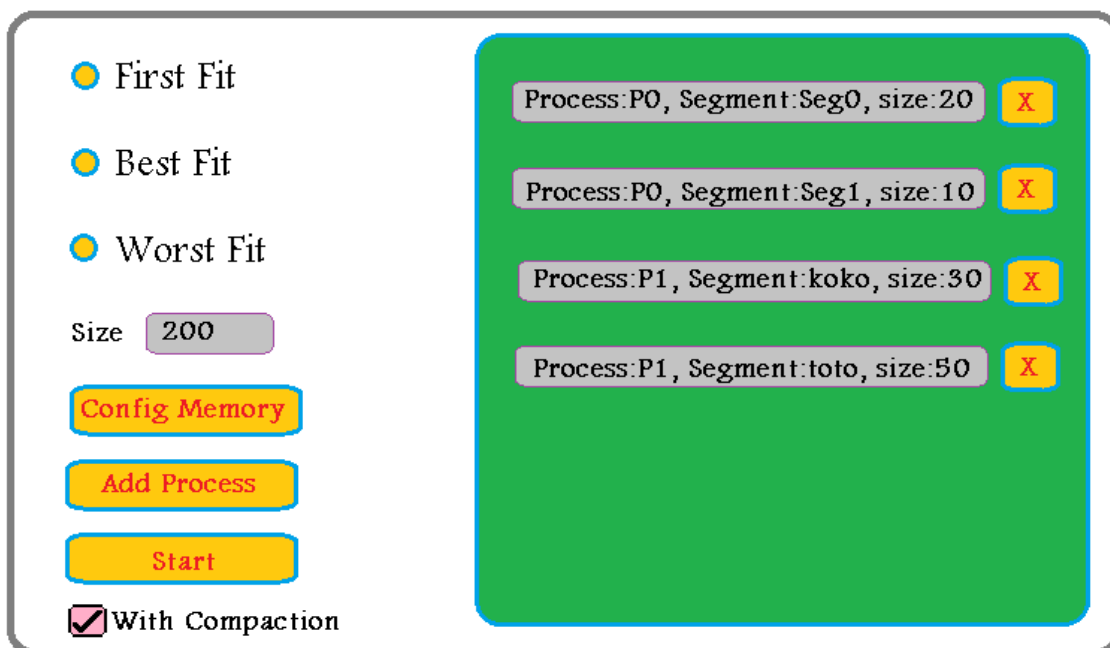


The diagram shows a dialog box titled "Add\_Process\_Dialog Example". It contains a "Process Name" field with the value "P1". Below this are two buttons: "Add Segment" and "Done". To the right of these buttons is a green rectangular area containing two entries: "Seg0, size:20" and "Seg1, size:10". Each entry has a yellow button with a red "X" next to it, indicating a delete function.

- The user can delete segments by press the “X” button beside them.

After the user clicks “done” he will be returned to the Main Window.

### *Main\_Window Example*

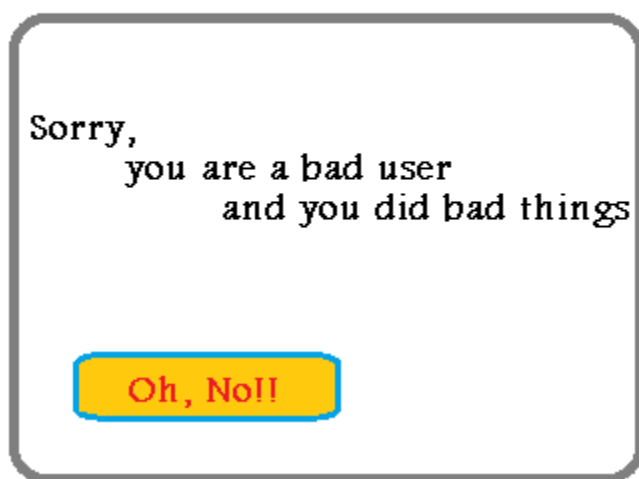


The diagram shows a main window titled "Main\_Window Example". It contains several options: "First Fit", "Best Fit", and "Worst Fit", each with a radio button. Below these is a "Size" field with the value "200". There are three buttons: "Config Memory", "Add Process", and "Start". At the bottom left is a checkbox labeled "With Compaction" which is checked. To the right of these options is a green rectangular area containing four entries: "Process:P0, Segment:Seg0, size:20", "Process:P0, Segment:Seg1, size:10", "Process:P1, Segment:koko, size:30", and "Process:P1, Segment:toto, size:50". Each entry has a yellow button with a red "X" next to it, indicating a delete function.

-The user can see the segments of every process in the green box, he can delete segments by clicking the "X" button beside it.

If any logical error happened(e.g , the user inserted a negative number as the size of memory), the user should be greeted with this error Dialog explaining the error he had done.

### *Error Dialog*



## OS Project 2 specifications

-----Classes responsible for back end-----

Hole

Segment

Process

Memory

Allocator

First\_Fit

Best\_Fit

Worst\_Fit

-----

-----Classes responsible for front end-----

Gtk.\*\*\*\*\* <-- choose whatever Gtk class you may like, replace \*\*\*\*\* with  
window / grid / dialog

- Main\_Window
- Add\_Process\_Dialog
- Add\_Segment\_Dialog
- Add\_Hole\_Dialog
- New\_Memory\_Dialog
- Output\_Grid
- Error\_Dialog

Gtk.Box <-- Gtk container class to hold multiple items (e.g text + button)

- list\_element
  - Hole\_list\_element
  - Segment\_list\_element
  - Process\_list\_element

Hole:

- vars:
  - int start\_address
  - int size

Segment:

- vars:
  - int segment\_index
  - int process\_index
  - string name
  - int start\_address
  - int size

- functions:

- \_\_init\_\_(self, String name,int segment\_index,int process\_index, int  
start\_address, int size)
  - remove\_segment()

process:

vars:

int process\_index

string name

array Segment

functions:

\_\_init\_\_(self, String name)

\_\_del\_\_(self)

add\_segment(Segment segment)

remove\_segment(int segment\_index)

Memory:

vars:

int size

array Process

functions:

\_\_init\_\_(self, int size, array hole)

add\_Process(Segment segment)

remove\_Process(int process\_index)

Allocator:

vars:

array Process

functions:

\_\_init\_\_(self, array Process)

allocate() <---- the children of this class should implement this

First\_Fit:

functions:

\_\_init\_\_(self, array Process) <--call parent's constructor

allocate()

Best\_Fit:

functions:

\_\_init\_\_(self, array Process) <--call parent's constructor

allocate()

Worst\_Fit:



functions:

```
__init__(self, array Process) <--call parent's constructor  
allocate()
```