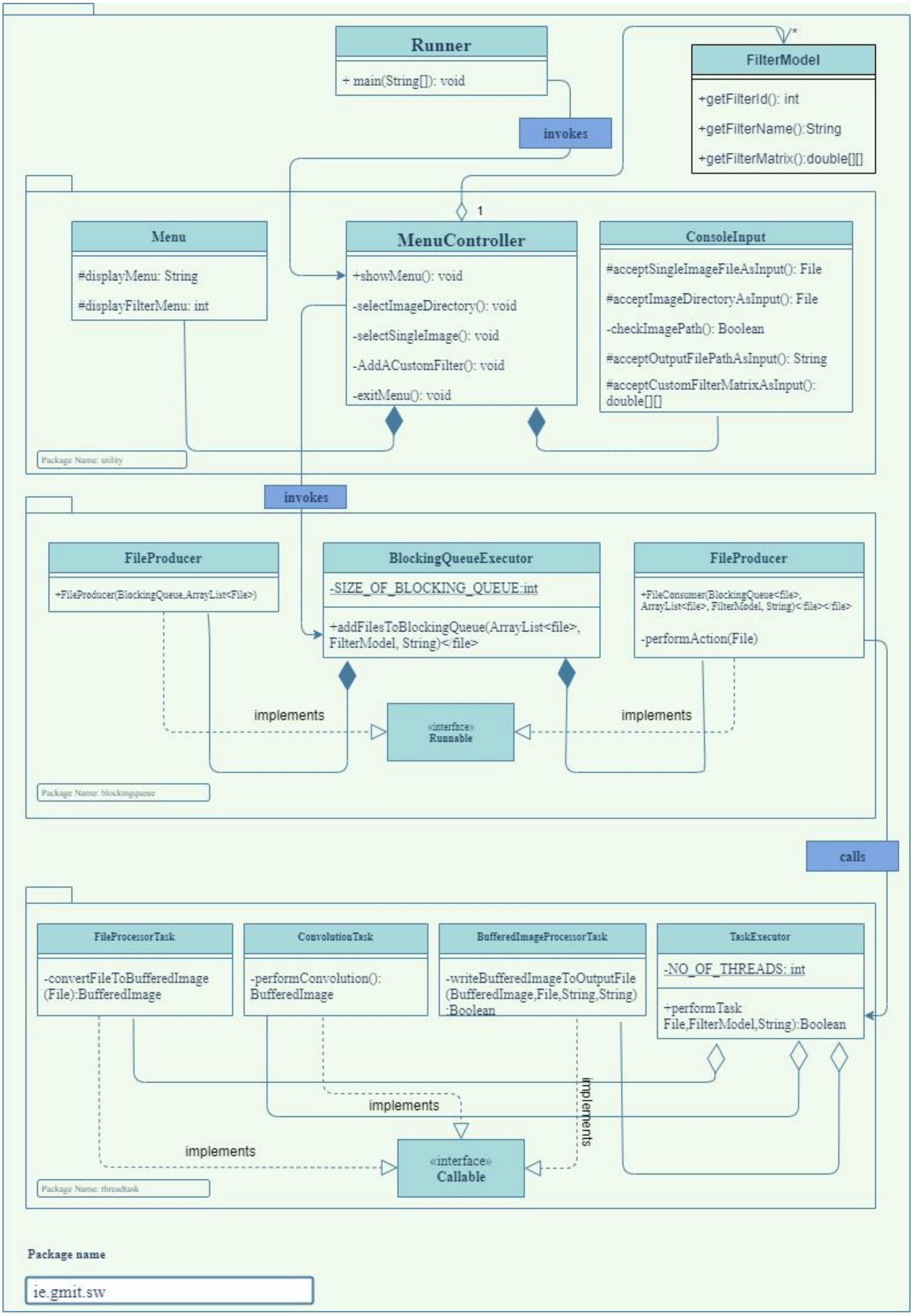
Image Filtering System

Data Structure/Types used in the application

Types/Data structure	Description
Scanner class	To accept input from the user.
ArrayList	To store file path.
ArrayBlockingQueue	Implements a queue which is thread safe, put elements into, and take elements out of it.
Two Dimensional array	To store kernel matrix of different filters.
Pojo Model Class	To store FilterId, FilterName, FilterMatrix.
Runnable interface	Implemented by the class whose Task to be executed by thread.
Callable<V> interface	an asynchronous task which can be executed by a separate thread.
Future<?>	to check the status of a Callable and then retrieve the result from the Callable once the thread is done.
ExecutorService	Submit and execute the asynchronous task.
Switch statement	For conditional execution of code block according to the user input.

UML Class Diagram


Runner
+ main(String[]): void
invokes
FilterModel
+getFilterId(): int
+getFilterName():String
+getFilterMatrix():double[][]
*
1
Menu
#displayMenu: String
#displayFilterMenu: int
MenuController
+showMenu(): void
-selectImageDirectory(): void
-selectSingleImage(): void
-AddACustomFilter(): void
-exitMenu(): void
ConsoleInput
#acceptSingleImageFileAsInput(): File
#acceptImageDirectoryAsInput(): File
-checkImagePath(): Boolean
#acceptOutputFilePathAsInput(): String
#acceptCustomFilterMatrixAsInput(): double[][]
Package Name: utility
invokes
FileProducer
+FileProducer(BlockingQueue,ArrayList<File>)
BlockingQueueExecutor
-SIZE_OF_BLOCKING_QUEUE:int
+addFilesToBlockingQueue(ArrayList<file>, FilterModel, String)</file>
FileProducer
+FileConsumer(BlockingQueue<file>, ArrayList<file>, FilterModel, String)</file></file>
-performAction(File)
implements
«interface» Runnable
implements
Package Name: blockingqueue
calls
FileProcessorTask
-convertFileToBufferedImage (File):BufferedImage
ConvolutionTask
-performConvolution(): BufferedImage
BufferedImageProcessorTask
-writeBufferedImageToOutputFile (BufferedImage,File,String,String) :Boolean
TaskExecutor
-NO_OF_THREADS: int
+performTask File,FilterModel,String):Boolean
implements
implements
implements
«interface» Callable
Package Name: threadtask
Package name
ie.gmit.sw

Summary

There few packages under the ie.gmit.sw package

- utility
- blockingqueque
- threadTask
- model

Runner: is the application starter class which has the main method it invokes showMenu method of MenuController.

MenuController: is the Controller class , it controls the main menu and perform action based on the user selection, it creates object of Menu and ConsoleInput. It is under the Utility package

Menu: this class has two methods
displayMenu: it display the main menu and return the choice from the user using the scanner class
displayFilterMenu: it display and return the filter selected by the user

ConsoleInput: it responsible for taking input image, input image directory, output path and validates if the path exist or not

BlockingqueueExecutor: this class is responsible for implementing a ArrayBlockingQueue of fixed size 3 and invokes object of FileProducer class and FileConsumer class, it is under blockingqueue package

FileProducer: this class implements runnable interface and insert image file into the ArrayBlockingqueue

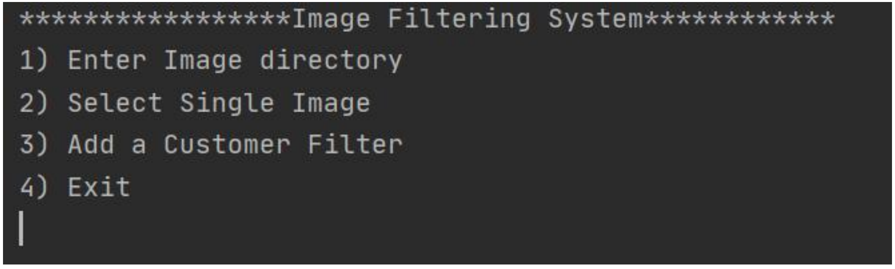
FileConsumer: this class implements runnable interface, calls performTask method of TaskExecutor and remove image file from the ArrayBlockingqueue

TaskExecutor: this class is responsibility for executing different task using ExecutorService with threadPool of size 3. it is under threadtask package
FileProcessorTask: this class implements callable interface, convert Image File to bufferedImage and returns BufferedImage Object.
ConvolutionTask: this class implements callable interface, performs the convolution of the input image pixel using the kernal matrix and returns a bufferedImage object.
BufferedImageTask: this class implements callable interface, write bufferedImage to output file
and returns the output file

model Package contains two classes which are
FilterModel: a pojo class with atrributes filterid, filtername, filtermatrix along with getter and setter methods
FilterMatrix: this class contains all the matrix values of filter in two dimensional array.

Features

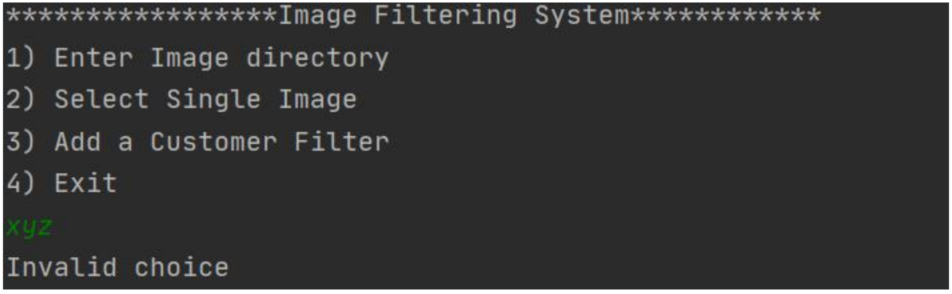
- A Persistent Main Menu, which consist of 4 options



```
*****************Image Filtering System************
1) Enter Image directory
2) Select Single Image
3) Add a Customer Filter
4) Exit
|
```

Accepts choice from the user

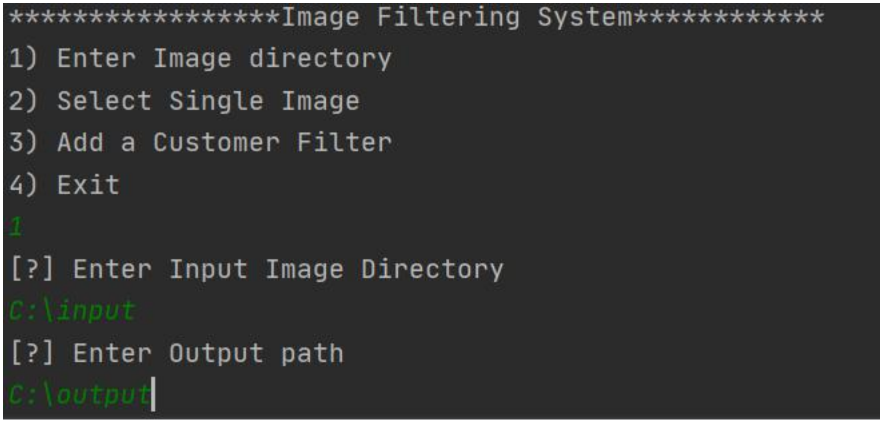
- Handles InputMismatchException in main menu,



```
*****************Image Filtering System************
1) Enter Image directory
2) Select Single Image
3) Add a Customer Filter
4) Exit
xyz
Invalid choice
```

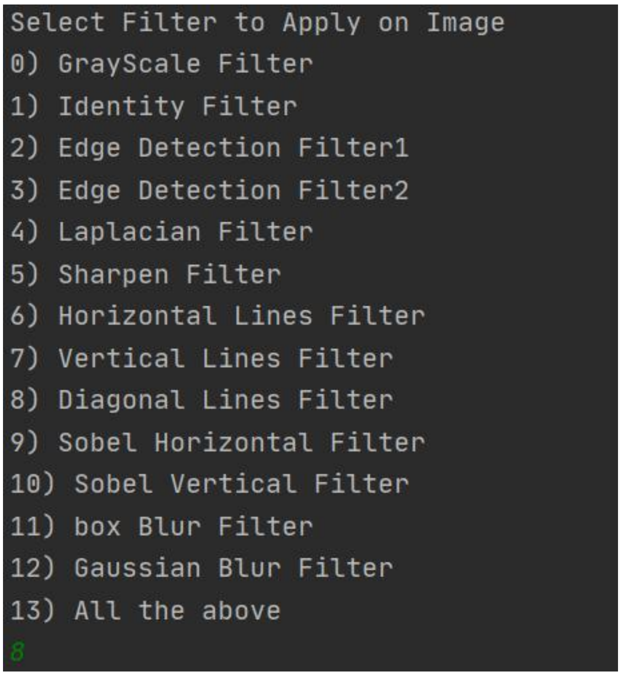
It display a message “invalid choice”. If a user enter any other character instead of number from 1-4

- **Option 1: Select Image Directory**



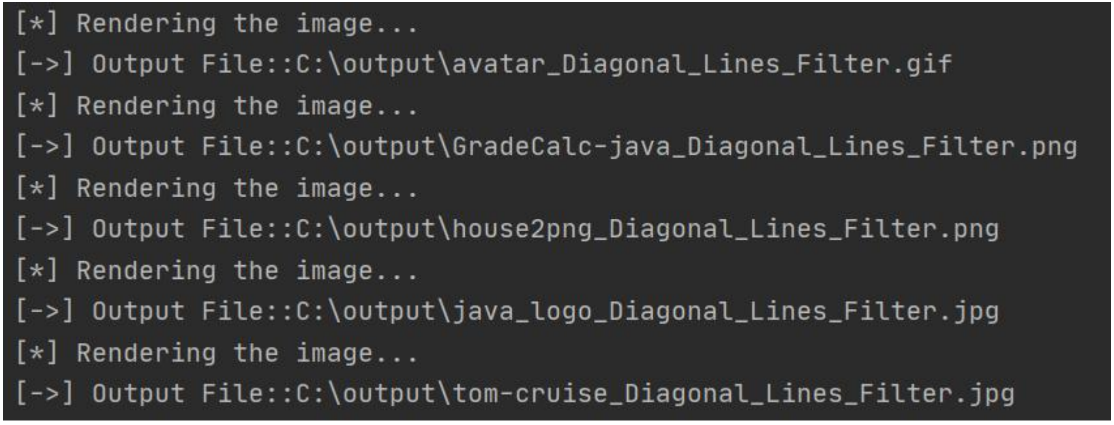
```
*****************Image Filtering System************
1) Enter Image directory
2) Select Single Image
3) Add a Customer Filter
4) Exit
1
[?] Enter Input Image Directory
C:\input
[?] Enter Output path
C:\output
```

Accepts the input image directory and output path



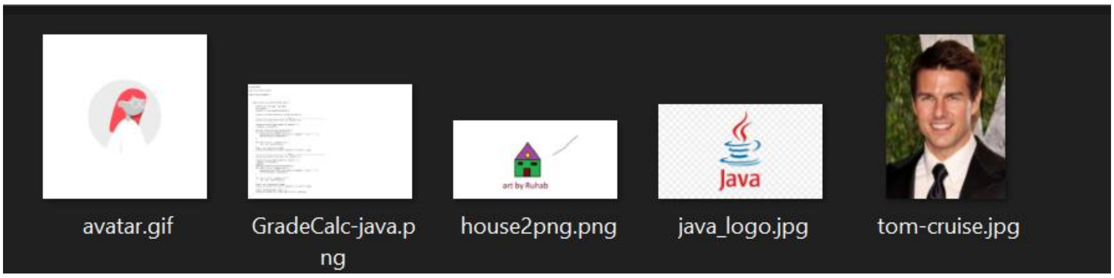
```
Select Filter to Apply on Image
0) GrayScale Filter
1) Identity Filter
2) Edge Detection Filter1
3) Edge Detection Filter2
4) Laplacian Filter
5) Sharpen Filter
6) Horizontal Lines Filter
7) Vertical Lines Filter
8) Diagonal Lines Filter
9) Sobel Horizontal Filter
10) Sobel Vertical Filter
11) box Blur Filter
12) Gaussian Blur Filter
13) All the above
8
```

Display a menu of filter, accept the choice from the user

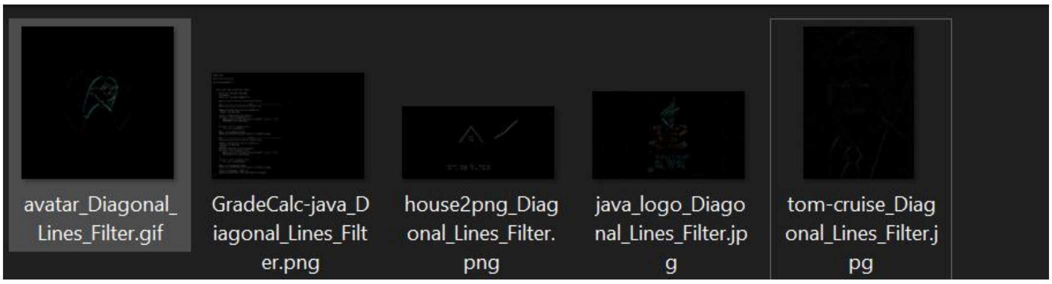


```
[*] Rendering the image...
[->] Output File::C:\output\avatar_Diagonal_Lines_Filter.gif
[*] Rendering the image...
[->] Output File::C:\output\GradeCalc-java_Diagonal_Lines_Filter.png
[*] Rendering the image...
[->] Output File::C:\output\house2png_Diagonal_Lines_Filter.png
[*] Rendering the image...
[->] Output File::C:\output\java_logo_Diagonal_Lines_Filter.jpg
[*] Rendering the image...
[->] Output File::C:\output\tom-cruise_Diagonal_Lines_Filter.jpg
```

Convolution operation is performed then Image is rendered and saved in the output directory



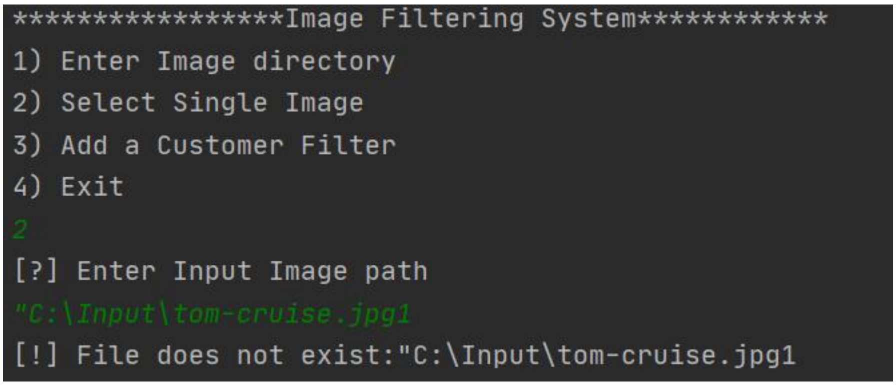
List of images in the input directory


avatar_Diagonal_Lines_Filter.gif
GradeCalc-java_Diagonal_Lines_Filter.png
house2png_Diagonal_Lines_Filter.png
java_logo_Diagonal_Lines_Filter.jpg
tom-cruise_Diagonal_Lines_Filter.jpg

List of files in the output directory on which the diagonal filter is applied

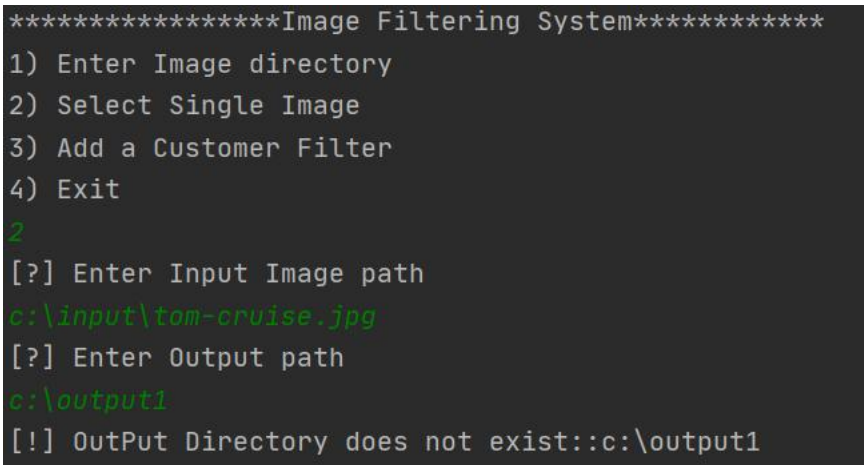
Option 2: Select Single Image

- Handles FileNotFoundException, with an appropriate message



```
*****************Image Filtering System************
1) Enter Image directory
2) Select Single Image
3) Add a Customer Filter
4) Exit
2
[?] Enter Input Image path
"C:\Input\tom-cruise.jpg1
[!] File does not exist:"C:\Input\tom-cruise.jpg1
```

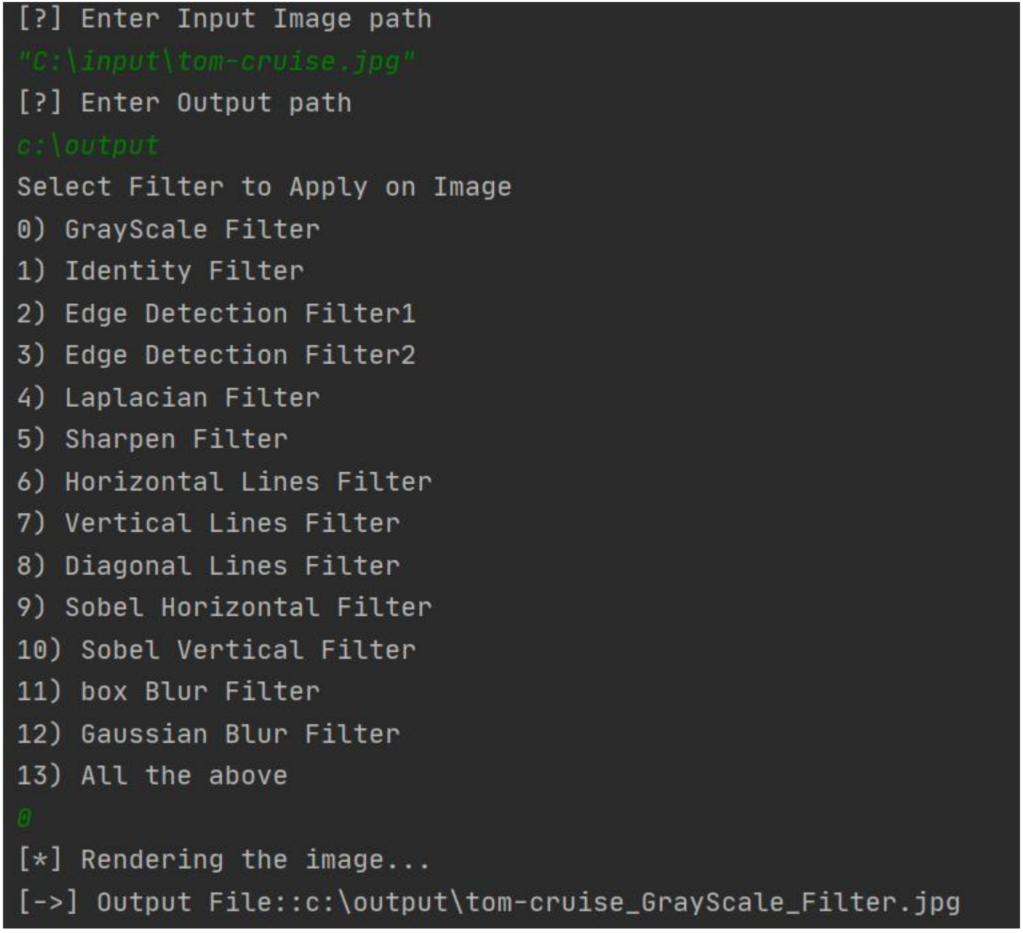
Display a message, if the input file doesn't exist



```
*****************Image Filtering System************
1) Enter Image directory
2) Select Single Image
3) Add a Customer Filter
4) Exit
2
[?] Enter Input Image path
c:\input\tom-cruise.jpg
[?] Enter Output path
c:\output1
[!] OutPut Directory does not exist::c:\output1
```

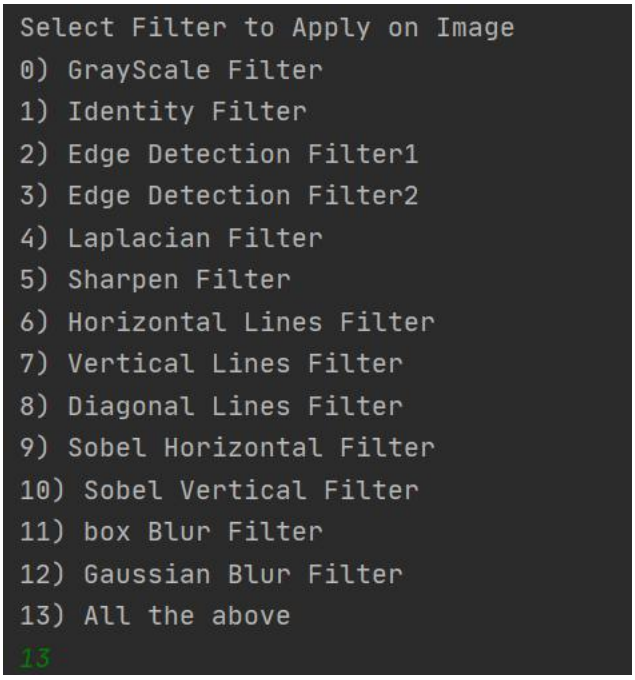
Display a message, if the output file doesn't exist

A Filter Menu which user can select and apply on the Image



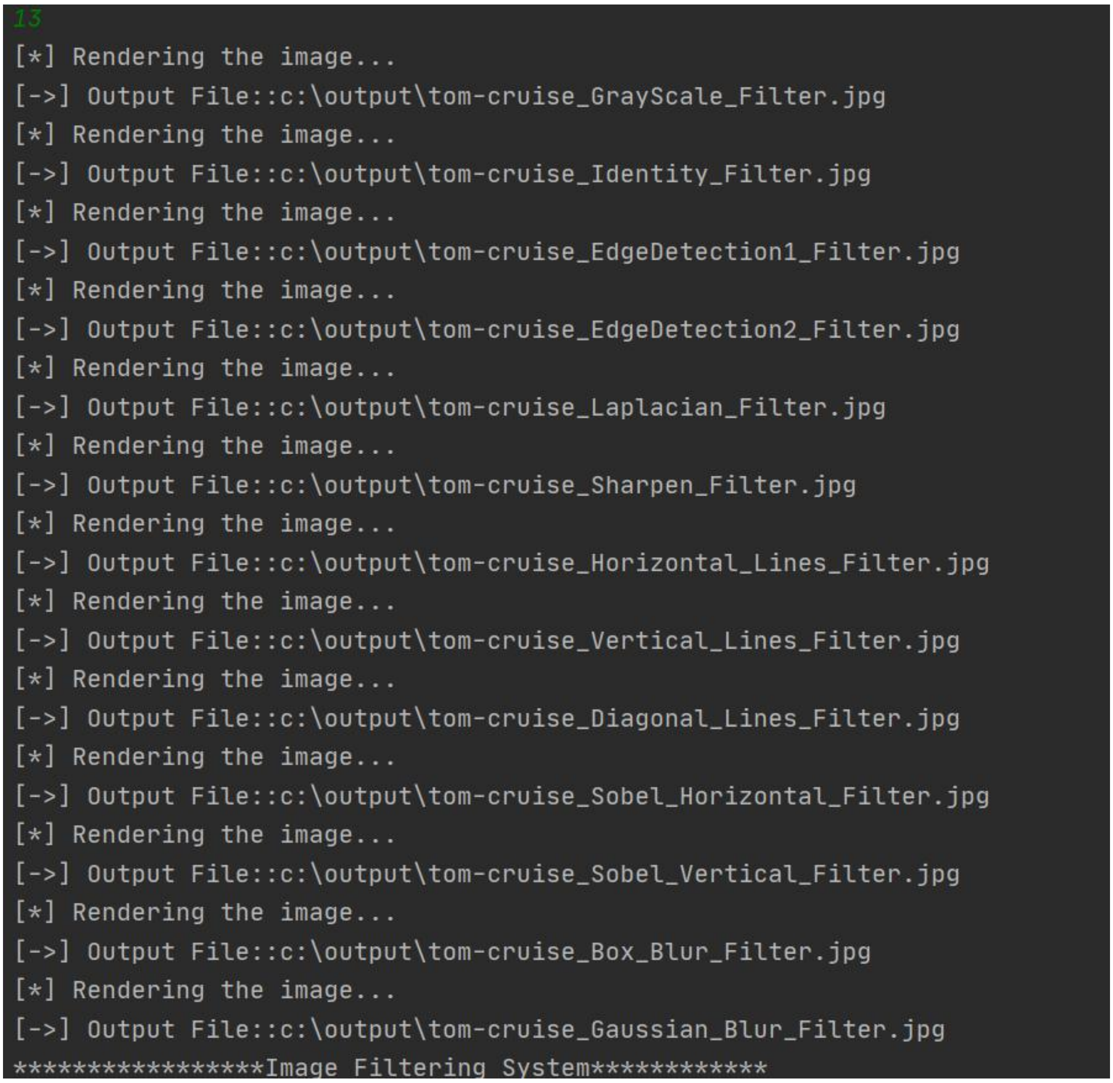
```
[?] Enter Input Image path
"C:\input\tom-cruise.jpg"
[?] Enter Output path
c:\output
Select Filter to Apply on Image
0) GrayScale Filter
1) Identity Filter
2) Edge Detection Filter1
3) Edge Detection Filter2
4) Laplacian Filter
5) Sharpen Filter
6) Horizontal Lines Filter
7) Vertical Lines Filter
8) Diagonal Lines Filter
9) Sobel Horizontal Filter
10) Sobel Vertical Filter
11) box Blur Filter
12) Gaussian Blur Filter
13) All the above
0
[*] Rendering the image...
[->] Output File::c:\output\tom-cruise_GrayScale_Filter.jpg
```

Applied Grayscale filter on the image and rendered image saved in the output



```
Select Filter to Apply on Image
0) GrayScale Filter
1) Identity Filter
2) Edge Detection Filter1
3) Edge Detection Filter2
4) Laplacian Filter
5) Sharpen Filter
6) Horizontal Lines Filter
7) Vertical Lines Filter
8) Diagonal Lines Filter
9) Sobel Horizontal Filter
10) Sobel Vertical Filter
11) box Blur Filter
12) Gaussian Blur Filter
13) All the above
13
```

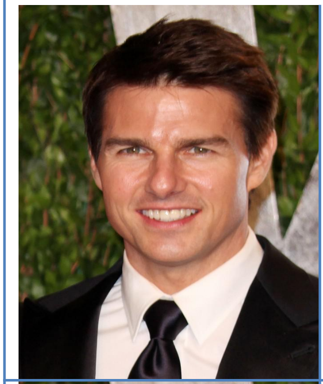
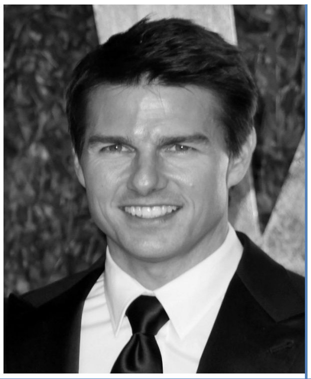
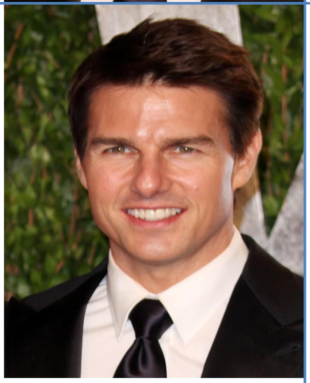
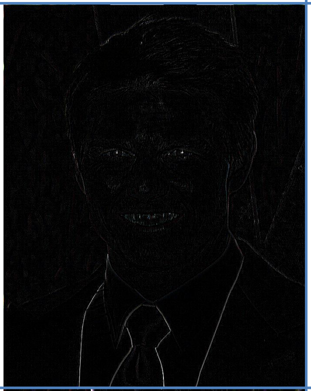
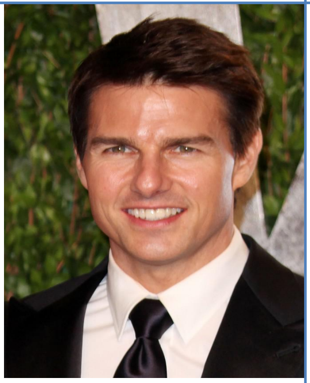
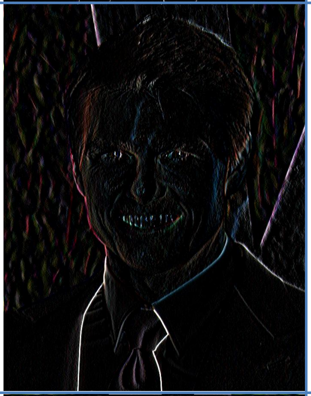
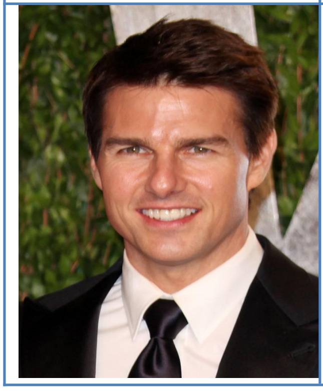
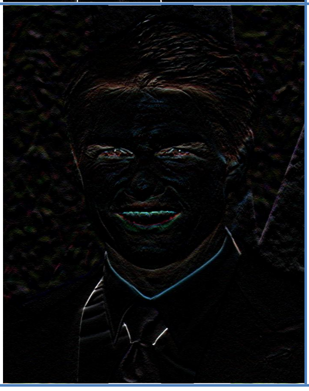
Selected option 13 , Apply all the filters on the Input Image

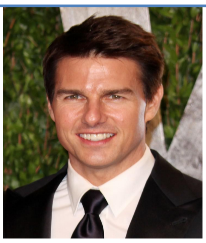
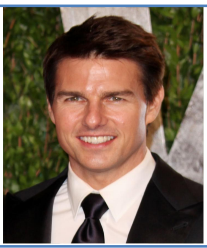


```
13
[*] Rendering the image...
[->] Output File::c:\output\tom-cruise_GrayScale_Filter.jpg
[*] Rendering the image...
[->] Output File::c:\output\tom-cruise_Identity_Filter.jpg
[*] Rendering the image...
[->] Output File::c:\output\tom-cruise_EdgeDetection1_Filter.jpg
[*] Rendering the image...
[->] Output File::c:\output\tom-cruise_EdgeDetection2_Filter.jpg
[*] Rendering the image...
[->] Output File::c:\output\tom-cruise_Laplacian_Filter.jpg
[*] Rendering the image...
[->] Output File::c:\output\tom-cruise_Sharpen_Filter.jpg
[*] Rendering the image...
[->] Output File::c:\output\tom-cruise_Horizontal_Lines_Filter.jpg
[*] Rendering the image...
[->] Output File::c:\output\tom-cruise_Vertical_Lines_Filter.jpg
[*] Rendering the image...
[->] Output File::c:\output\tom-cruise_Diagonal_Lines_Filter.jpg
[*] Rendering the image...
[->] Output File::c:\output\tom-cruise_Sobel_Horizontal_Filter.jpg
[*] Rendering the image...
[->] Output File::c:\output\tom-cruise_Sobel_Vertical_Filter.jpg
[*] Rendering the image...
[->] Output File::c:\output\tom-cruise_Box_Blur_Filter.jpg
[*] Rendering the image...
[->] Output File::c:\output\tom-cruise_Gaussian_Blur_Filter.jpg
*****************Image Filtering System************
```

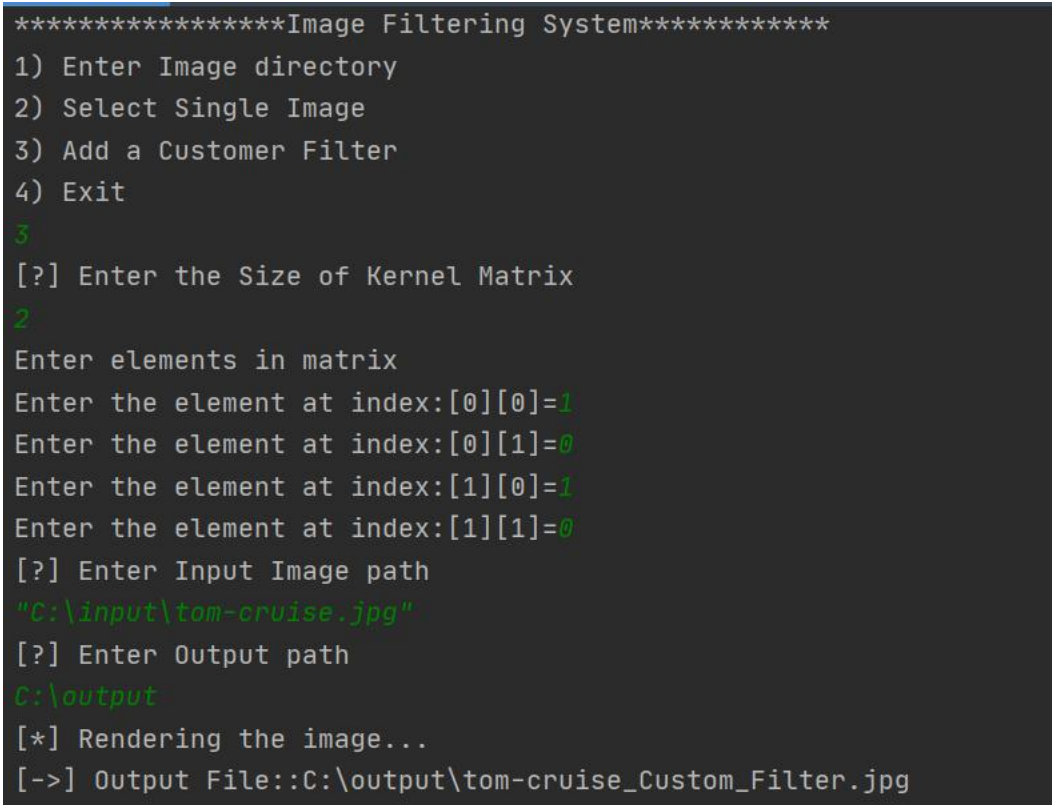
Rendered image saved in the output path

Some input and applied effect as output

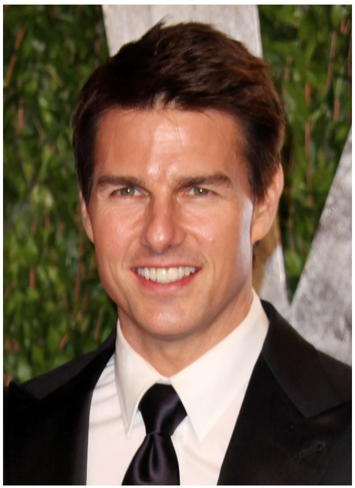
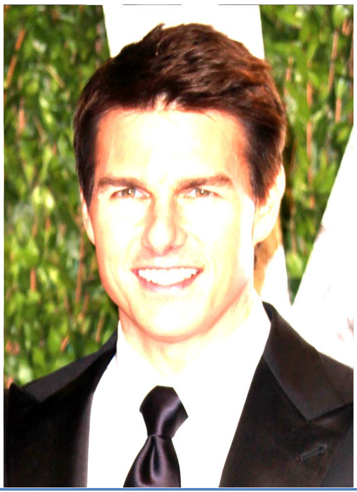
Input	output	Effect
		Grayscale
		Edge Detection
		Sobel horizontal
		Sobel vertical

		Box blur

- Option 3: Add a Custom Filter

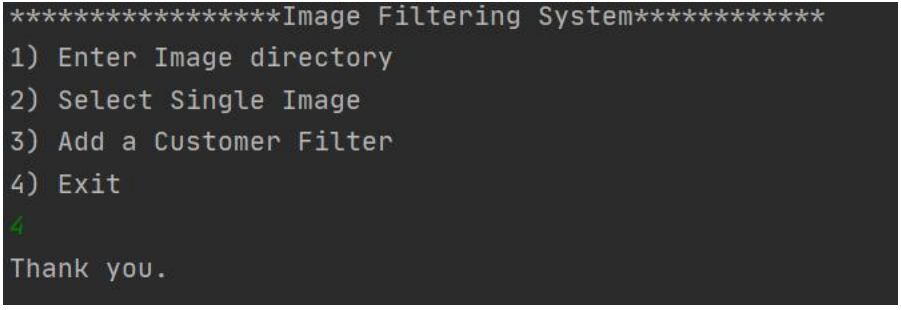


```
****************Image Filtering System************
1) Enter Image directory
2) Select Single Image
3) Add a Customer Filter
4) Exit
3
[?] Enter the Size of Kernel Matrix
2
Enter elements in matrix
Enter the element at index:[0][0]=1
Enter the element at index:[0][1]=0
Enter the element at index:[1][0]=1
Enter the element at index:[1][1]=0
[?] Enter Input Image path
"C:\input\tom-cruise.jpg"
[?] Enter Output path
C:\output
[*] Rendering the image...
[->] Output File::C:\output\tom-cruise_Custom_Filter.jpg
```

input	output	Custom filter
		{ 1,0}, {0,1},

Accepts a kernel size and elements of the kernel matrix and saved the image in the output folder

- Option 4: Exit



```
*****************Image Filtering System************
1) Enter Image directory
2) Select Single Image
3) Add a Customer Filter
4) Exit
4
Thank you.
```

Option 4 is selected and user exist from the main menu and program is terminated