# OOP (2CS302) Innovative Assignment

Group Member 1 : 20 BCE 012 - Amin Meet

Group Member 2 : 20 BCE 070 - Dwij Bavisi

## Project Title: ToDo List

## Project Description:

An app where you can add your daily task list.

* Add, Delete, Edit task
* Give description to a task
* Set Due date, mark a task as important or not
* Sorting all tasks based on priority.
* All task stored in file(File handling concept)

## Project Details:

### Concepts used

* Packages
* Inheritance (extends)
* Interface (implements)
* File handling
* Exception handling
* G.U.I. (Swing Library)
* Linked List

##### Project Link : [Meet91721/ToDo (github.com)](https://github.com/Meet91721/ToDo)

## Project Files:

### /Source/Main.java

package Source;

import Source.linkedList.List;

import Source.Frame.ListFrame;

import javax.swing.\*;

public class Main {

public static JFrame mainFrame;

public static List taskList;

public static ListFrame taskListFrame;

public static FileHandling fh;

public static void main(String[] args) throws Exception {

mainFrame = new JFrame();

mainFrame.setSize(1000, 800);

fh = new FileHandling();

taskList = fh.ScanFile();

taskListFrame = new ListFrame(taskList);

taskListFrame.Print();

}

}

### /Source/FileHandling.java

package Source;

import java.io.BufferedReader;

import Source.linkedList.List;

import java.io.File;

import Source.linkedList.Node;

import java.io.FileWriter;

import java.io.IOException;

import java.io.FileReader;

import java.text.ParseException;

import java.text.SimpleDateFormat;

public class FileHandling {

public void printinFILE(Node curr, String FileName) {

// Node curr = new Node();/

try {

FileWriter f = new FileWriter(FileName);

while (curr != null) {

// f.write(curr.Value + "\t" + curr.Due + "\t" + curr.Important + "\n");

f.write(curr.Title + "\t" + curr.Description + "\t" + curr.Due + "\t" + curr.Important + "\n");

curr = curr.Next;

}

f.close();

} catch (IOException exception) {

System.out.println("Error occured");

exception.printStackTrace();

}

}

public List ScanFile() throws Exception {

List taskList = new List();

try {

File f = new File("./Source/Data.txt");

BufferedReader br = new BufferedReader(new FileReader(f));

String st;

while ((st = br.readLine()) != null) {

// Print the string

// System.out.println(st);

String words[] = st.split("\t");

SimpleDateFormat formatter = new SimpleDateFormat("E MMM DD HH:mm:ss z yyyy");

Node temp = new Node();

temp.Title = words[0];

temp.Description = words[1];

temp.Due = formatter.parse(words[2]);

try {

temp.Due = formatter.parse(words[2]);

} catch (ParseException e) {

System.out.println("Error occurred");

e.printStackTrace();

}

temp.Important = Boolean.parseBoolean(words[3]);

taskList.Insert(temp);

}

br.close();

// f.close();

} catch (IOException exception) {

System.out.println("An unexpected error is occurred.");

exception.printStackTrace();

}

return taskList;

}

}

### /Source/linkedList/Node.java

package Source.linkedList;

import java.util.Date;

import java.util.Calendar;

public class Node {

public Node Prev;

public Node Next;

public String Title;

public String Description;

public Date Due;

public Boolean Important;

public Boolean Urgent() {

return dueIn() < 7;

}

public Boolean OverDue() {

return dueIn() < 0;

}

public long dueIn() {

if (Due != null) {

Date Today = Calendar.getInstance().getTime();

long DiffMS = Due.getTime() - Today.getTime();

long DiffDAYS = DiffMS / (1000 \* 60 \* 60 \* 24);

return DiffDAYS;

} else {

return 8;

}

}

}

### /Source/linkedList/List.java

package Source.linkedList;

public class List {

public Node Head;

public void Insert(Node N) {

// Node N to be inserted in the list;

// Linked list structure

// U = Urgent, I = Important

// [UI]->[U?]->[?I]->[??]

// Insertion in [??] - Begining of section

// Insertion in [?I] - Begining of section

// Insertion in [U?] - According to time left

// Insertion in [UI] - According to time left

if (Head == null) {

// Linked List is null,

// N should be at Head

Head = N;

Head.Prev = null;

Head.Next = null;

} else {

// Check where N belongs

Node temp = Head;

if (N.Urgent()) {

// N belongs to [UI] or [U?]

if (N.Important) {

// N belongs to [UI]

// Skip all Nodes where Nodes.dueIn() is < N.dueIn();

while (temp.Next != null && temp.Next.dueIn() < N.dueIn()) {

temp = temp.Next;

}

} else {

// N belongs to [U?]

// Skip all Nodes in [UI], and in [U?] where Nodes.dueIn() is < N.dueIn();

while (temp.Next != null && (temp.Next.Urgent() && temp.Next.Important)) {

temp = temp.Next;

}

// Skipped all Nodes in [UI]

// Now skip all Nodes where Nodes.dueIn() is < N.dueIn();

while (temp.Next != null && temp.Next.dueIn() < N.dueIn()) {

temp = temp.Next;

}

}

} else if (N.Important) {

// N belongs to [?I]

// Skip all Nodes which are [UI] or [U?] and insert before [?I]

while (temp.Next != null && temp.Next.Urgent()) {

temp = temp.Next;

}

} else {

// N belongs to [??]

// Skip all Nodes which are [UI], [U?] or [?I] and insert before [??]

while (temp.Next != null && (temp.Next.Urgent() || temp.Next.Important)) {

temp = temp.Next;

}

}

if (temp == Head) {

// temp is Head

if (N.Urgent() && N.Important) {

if (Head.Urgent() && Head.Important && Head.dueIn() < N.dueIn()) {

N.Next = Head.Next;

N.Prev = Head;

Head.Next = N;

} else {

N.Next = Head;

N.Prev = null;

Head = N;

}

} else if (N.Urgent() && !N.Important) {

if (Head.Urgent() && (Head.Important || Head.dueIn() < N.dueIn())) {

N.Next = Head.Next;

N.Prev = Head;

Head.Next = N;

} else {

N.Next = Head;

N.Prev = null;

Head = N;

}

} else if (!N.Urgent() && N.Important) {

if (Head.Urgent()) {

N.Next = Head.Next;

N.Prev = Head;

Head.Next = N;

} else {

N.Next = Head;

N.Prev = null;

Head = N;

}

} else {

if (Head.Urgent() || Head.Important) {

N.Next = Head.Next;

N.Prev = Head;

Head.Next = N;

} else {

N.Next = Head;

N.Prev = null;

Head = N;

}

}

} else {

N.Next = temp.Next;

N.Prev = temp;

temp.Next = N;

}

if (N.Next != null) {

N.Next.Prev = N;

}

}

return;

} // Insert

public Node Delete(Node N) {

if (N == Head) {

// deletion at beginning

if (N.Next != null) {

Head = N.Next;

Head.Prev = null;

} else {

Head = null;

}

} else {

N.Prev.Next = N.Next;

if (N.Next != null) {

N.Next.Prev = N.Prev;

}

}

N.Next = null;

N.Prev = null;

return N;

} // Delete

public void ReInsert(Node N) {

Insert(Delete(N));

} // ReInsert

}

### /Source/Frame/Frame.java

package Source.Frame;

public interface Frame {

public void Print();

}

### /Source/Frame/NodeFrame.java

package Source.Frame;

import Source.Main;

import Source.linkedList.Node;

import java.awt.event.\*;

import javax.swing.\*;

import java.text.SimpleDateFormat;

public class NodeFrame extends Node implements Frame {

Node node;

public NodeFrame(Node N) {

super();

this.node = N;

this.Title = N.Title;

this.Description = N.Description;

this.Next = N.Next;

this.Prev = N.Prev;

this.Important = N.Important;

this.Due = N.Due;

}

public void Print() {

Main.mainFrame.setVisible(false);

Main.mainFrame = new JFrame();

Main.mainFrame.setSize(1000, 800);

Main.taskListFrame.Print();

JTextField title = new JTextField(this.Title);

title.setBounds(400, 50, 400, 40);// x, y, w, h

JTextField description = new JTextField(this.Description);

description.setBounds(400, 120, 400, 40);

JTextField due = new JTextField(String.valueOf(this.Due));

due.setBounds(400, 180, 400, 40);

JTextField imp = new JTextField(String.valueOf(this.Important));

imp.setBounds(400, 240, 400, 40);

Main.mainFrame.add(title);

Main.mainFrame.add(description);

Main.mainFrame.add(due);

Main.mainFrame.add(imp);

JButton delete = new JButton("Delete Task");

delete.setBounds(800, 90, 200, 40);

JButton edit = new JButton("Edit Task");

edit.setBounds(800, 230, 200, 40);

delete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Main.taskList.Delete(node);

Main.taskListFrame = new ListFrame(Main.taskList);

Main.taskListFrame.Print();

Main.fh.printinFILE(Main.taskList.Head, "./Source/Data.txt");

}

});

edit.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

// Node te = new Node();

node.Description = description.getText();

try {

node.Due = new SimpleDateFormat("dd/MM/yyyy").parse(due.getText());

} catch (Exception E) {

E.printStackTrace();

}

node.Important = Boolean.parseBoolean(imp.getText());

node.Title = title.getText();

Main.taskList.ReInsert(node);

Main.taskListFrame = new ListFrame(Main.taskList);

Main.taskListFrame.Print();

Main.fh.printinFILE(Main.taskList.Head, "./Source/Data.txt");

}

});

Main.mainFrame.add(delete);

Main.mainFrame.add(edit);

JLabel extra = new JLabel();

extra.setBounds(400, 240, 600, 40);

extra.setVisible(false);

Main.mainFrame.add(extra);

Main.mainFrame.setVisible(true);

}

}

### /Source/Frame/ListFrame.java

package Source.Frame;

import Source.Main;

import Source.linkedList.List;

import Source.linkedList.Node;

import java.awt.event.\*;

import javax.swing.\*;

import Source.Frame.ListFrame;

import java.text.SimpleDateFormat;

public class ListFrame extends List implements Frame {

public ListFrame(List L) {

super();

this.Head = L.Head;

}

public void Print() {

Main.mainFrame.setVisible(false);

Main.mainFrame = new JFrame();

Main.mainFrame.setSize(1000, 800);

Node Temp = this.Head;

int Counter = 0;

while (Temp != null) {

final Node t = Temp;

System.out.println("Hello" + Counter);

JLabel Label = new JLabel("<html><h1>" + Temp.Title + "</h1></html>");

Label.setBounds(21, 17 + 37 \* Counter, 200, 35);// x, y, w, h

Label.addMouseListener(new MouseListener() {

public void mouseClicked(MouseEvent e) {

// Label.setText("Mouse Clicked");

NodeFrame current = new NodeFrame(t);

current.Print();

}

public void mouseEntered(MouseEvent e) {

// Label.setText("Mouse Entered");

}

public void mouseExited(MouseEvent e) {

// Label.setText("Mouse Exited");

}

public void mousePressed(MouseEvent e) {

// Label.setText("Mouse Pressed");

}

public void mouseReleased(MouseEvent e) {

// Label.setText("Mouse Released");

}

});

Main.mainFrame.add(Label);

Temp = Temp.Next;

Counter++;

}

JTextField tit = new JTextField();

JTextField des = new JTextField();

JTextField du = new JTextField();

JTextField im = new JTextField();

JLabel l1 = new JLabel("Title");

JLabel l2 = new JLabel("Description");

JLabel l3 = new JLabel("Due Date");

JLabel l4 = new JLabel("Important");

l1.setBounds(21, 470, 150, 30);

l2.setBounds(21, 570, 150, 30);

l3.setBounds(300, 470, 150, 30);

l4.setBounds(300, 570, 150, 30);

tit.setBounds(21, 500, 150, 30);

des.setBounds(21, 600, 150, 30);

du.setBounds(300, 500, 150, 30);

im.setBounds(300, 600, 150, 30);

Main.mainFrame.add(l1);

Main.mainFrame.add(l2);

Main.mainFrame.add(l3);

Main.mainFrame.add(l4);

Main.mainFrame.add(tit);

Main.mainFrame.add(des);

Main.mainFrame.add(du);

Main.mainFrame.add(im);

JButton Binsert = new JButton("Add task");

Binsert.setBounds(170, 700, 200, 40);

Main.mainFrame.add(Binsert);

Binsert.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Node te = new Node();

te.Description = des.getText();

try {

te.Due = new SimpleDateFormat("dd/MM/yyyy").parse(du.getText());

} catch (Exception E) {

E.printStackTrace();

}

te.Important = Boolean.parseBoolean(im.getText());

te.Title = tit.getText();

Main.taskList.Insert(te);

Main.taskListFrame = new ListFrame(Main.taskList);

Main.taskListFrame.Print();

Main.fh.printinFILE(Main.taskList.Head, "./Source/Data.txt");

}

});

JLabel Label = new JLabel();

Label.setVisible(false);

Main.mainFrame.add(Label);

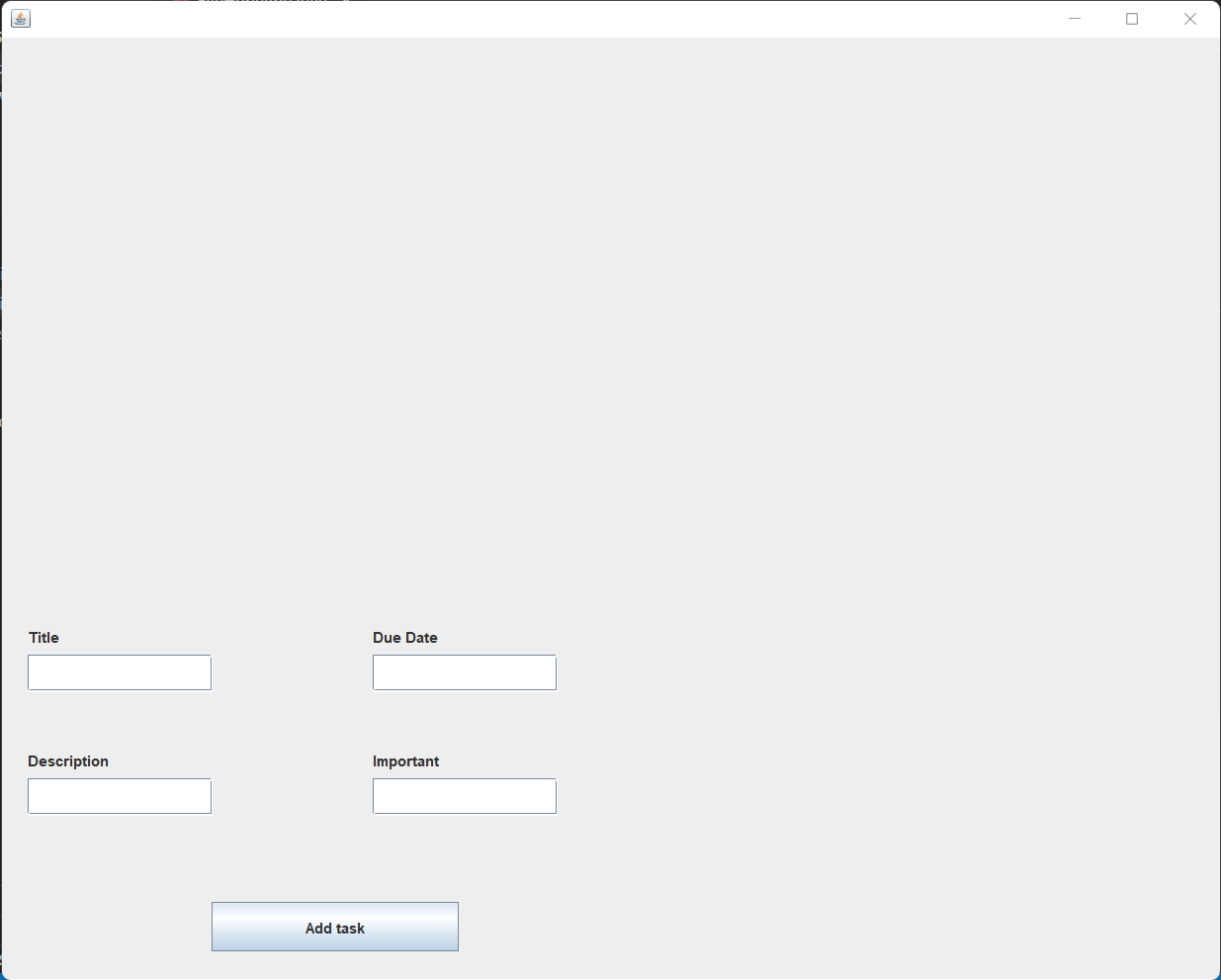
Main.mainFrame.setVisible(true);

}

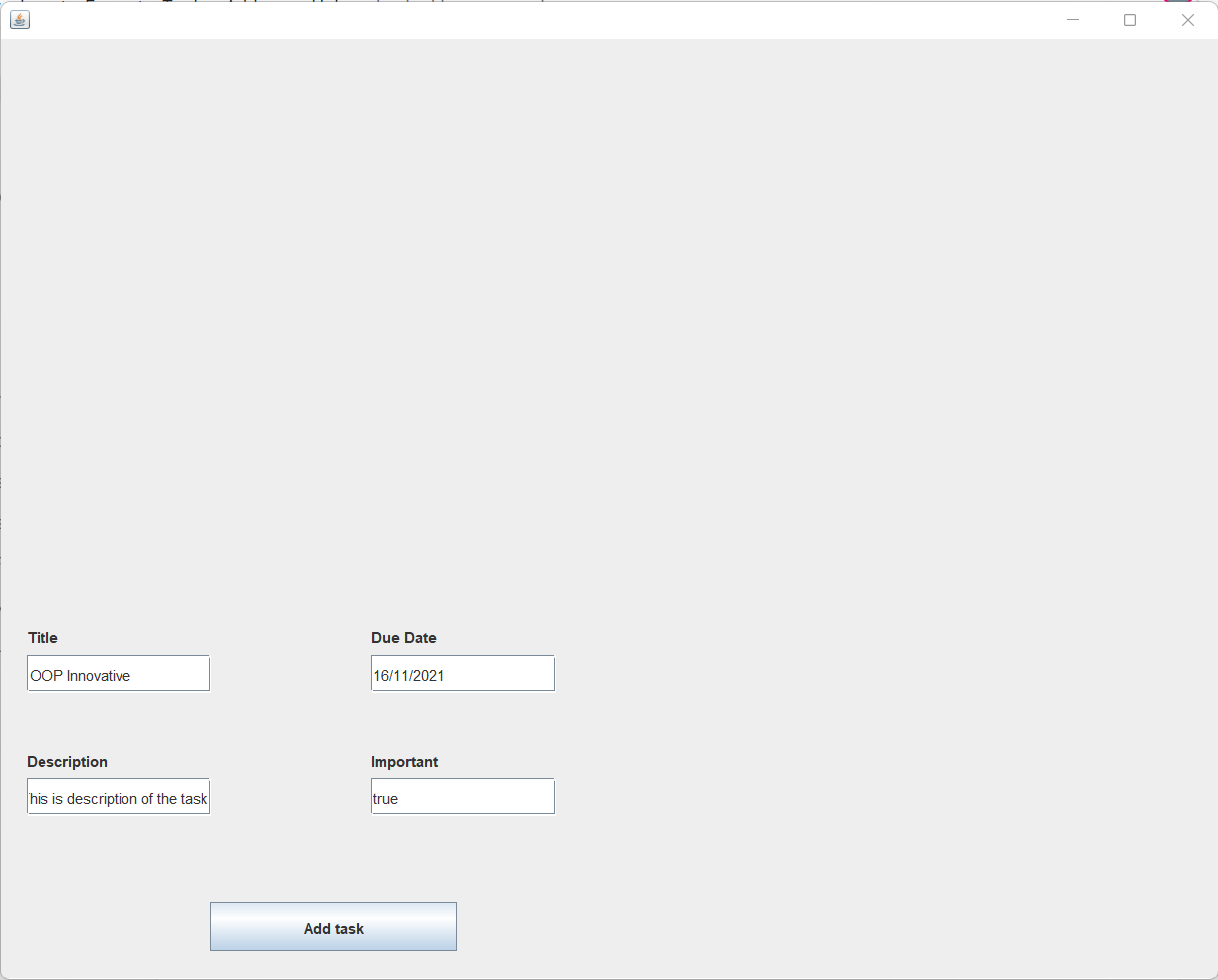
}

## Output:

### 1) Startup Screen

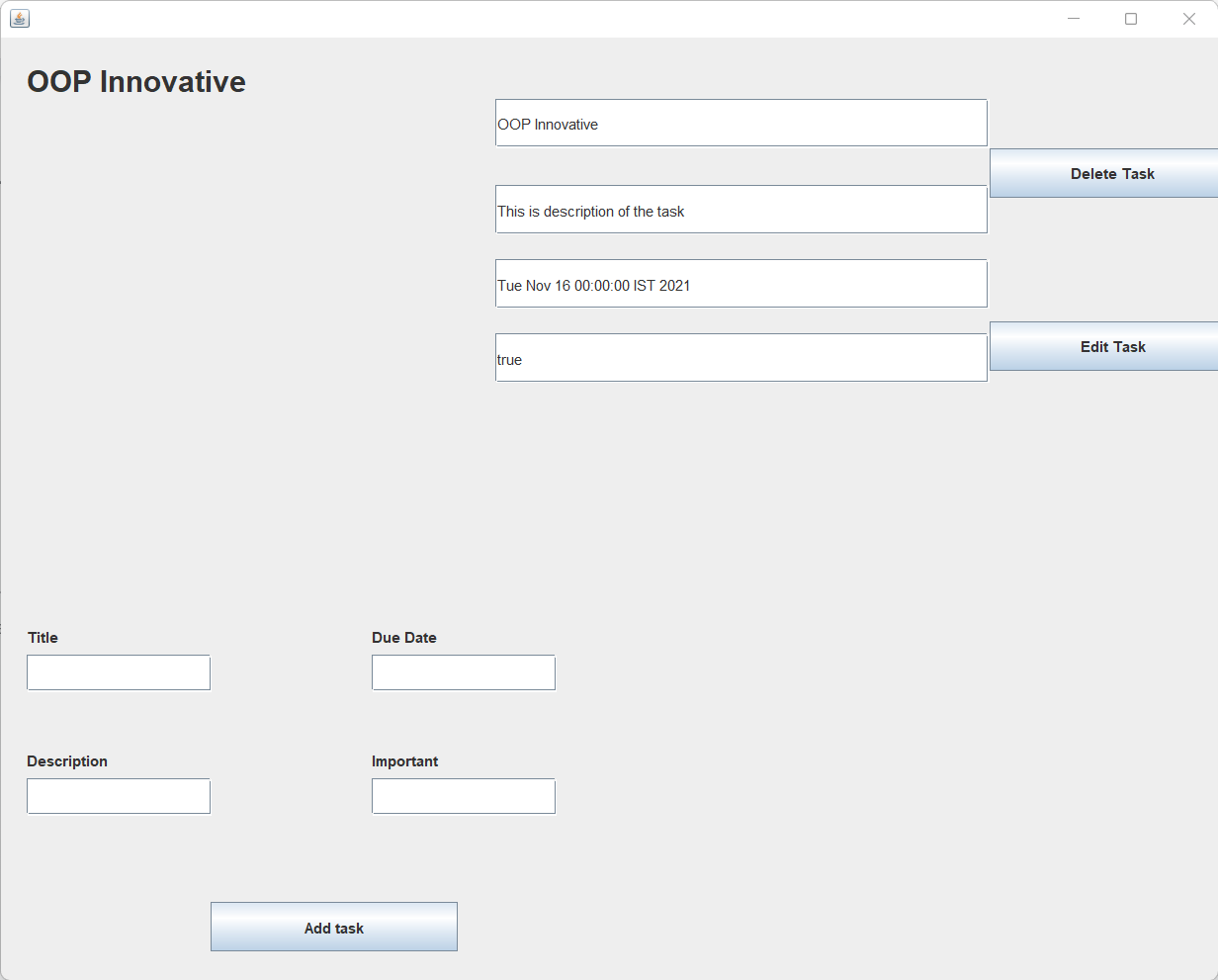


### 2) Add Task

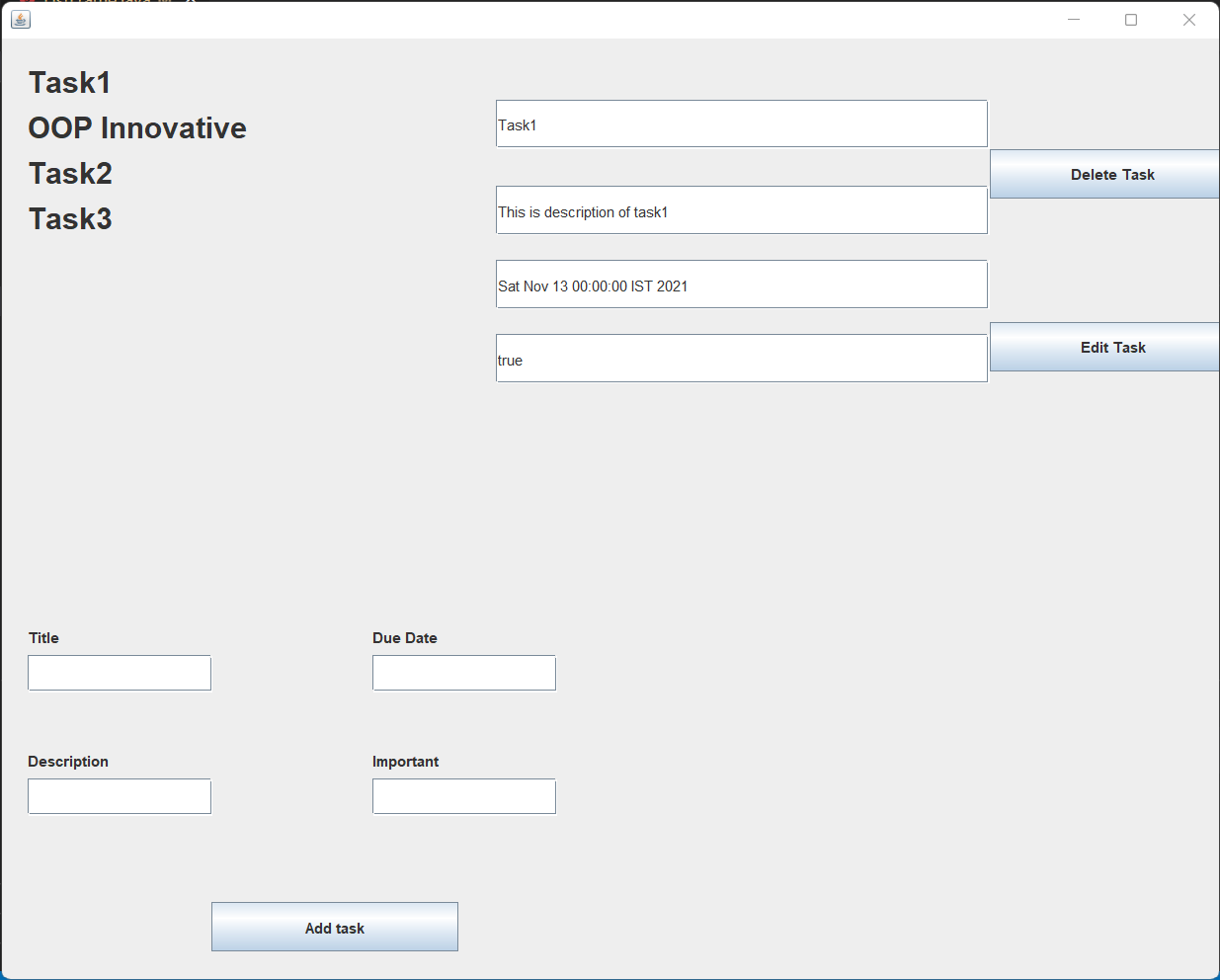


User can add title, description, due date and important.

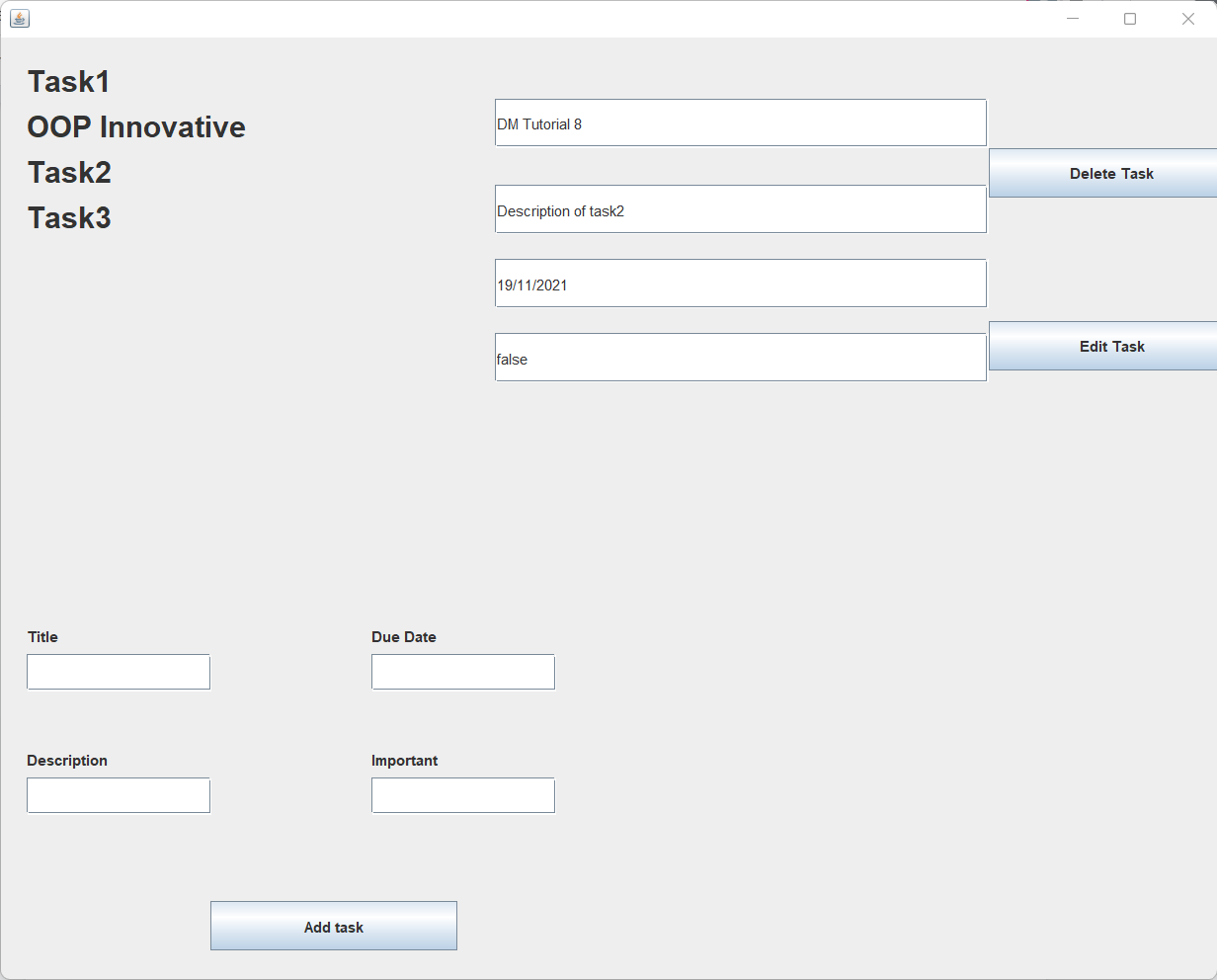
### 3) Viewing Task



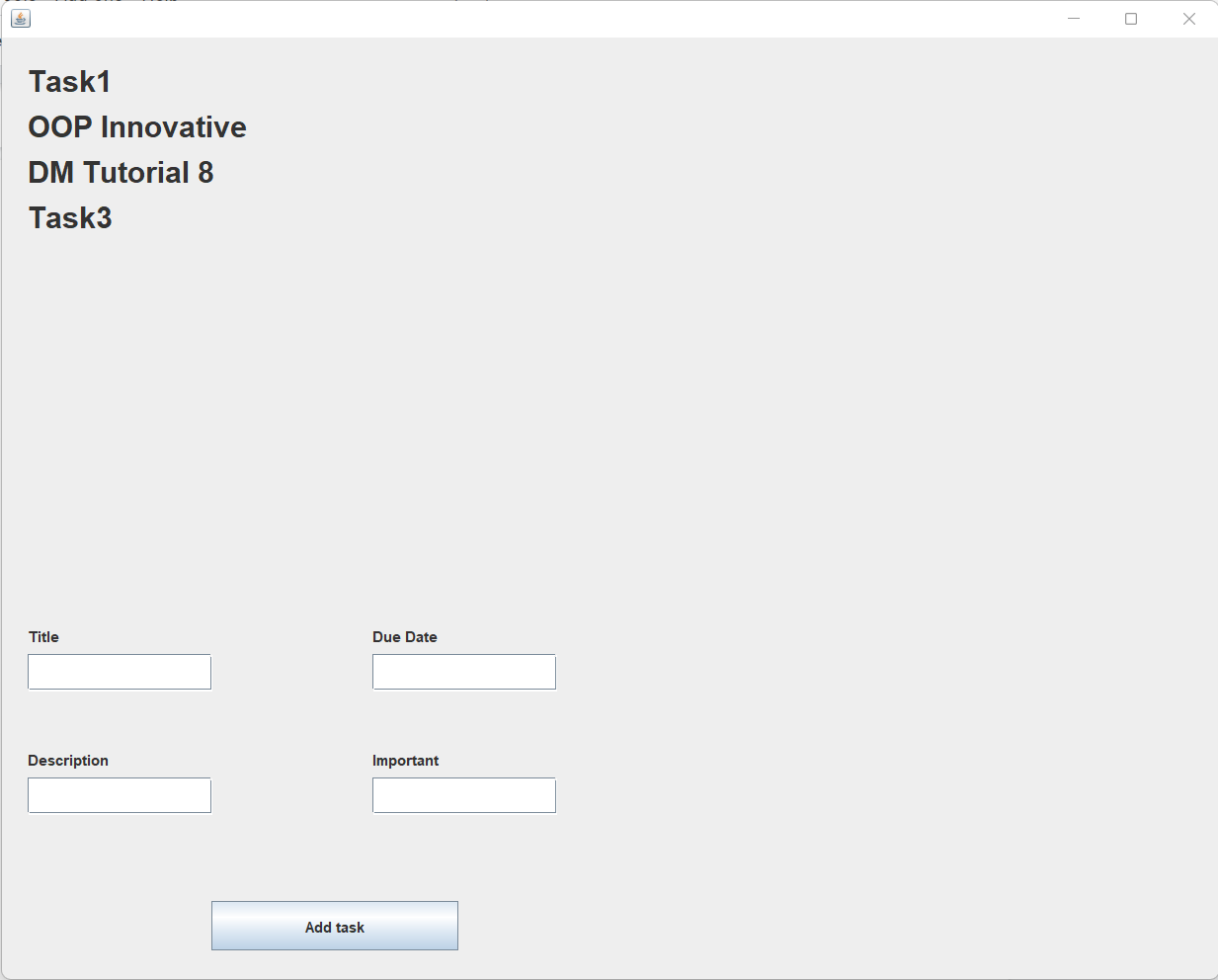
### 4) Sorting all tasks based on importance and due date



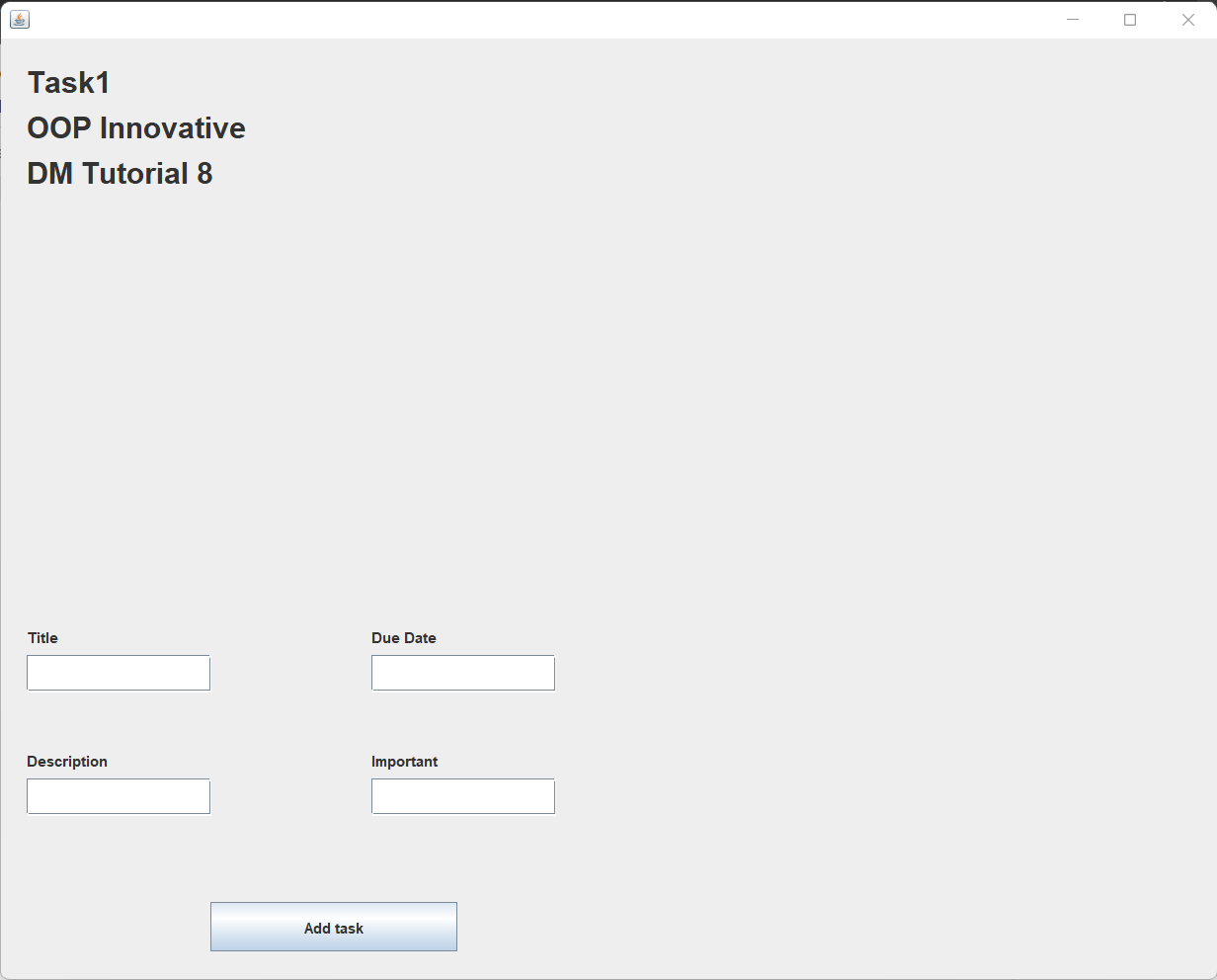
### 5) Options to Edit or Delete Task



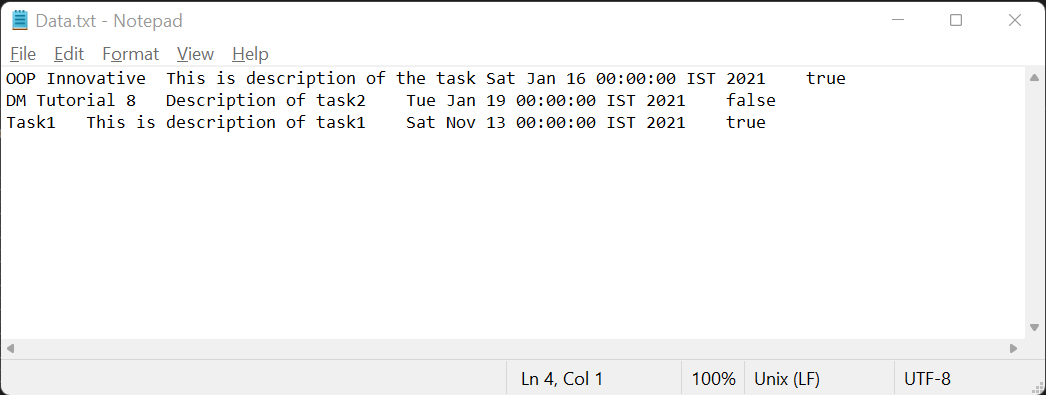
### 6) Edit Task



### 7) Delete Task

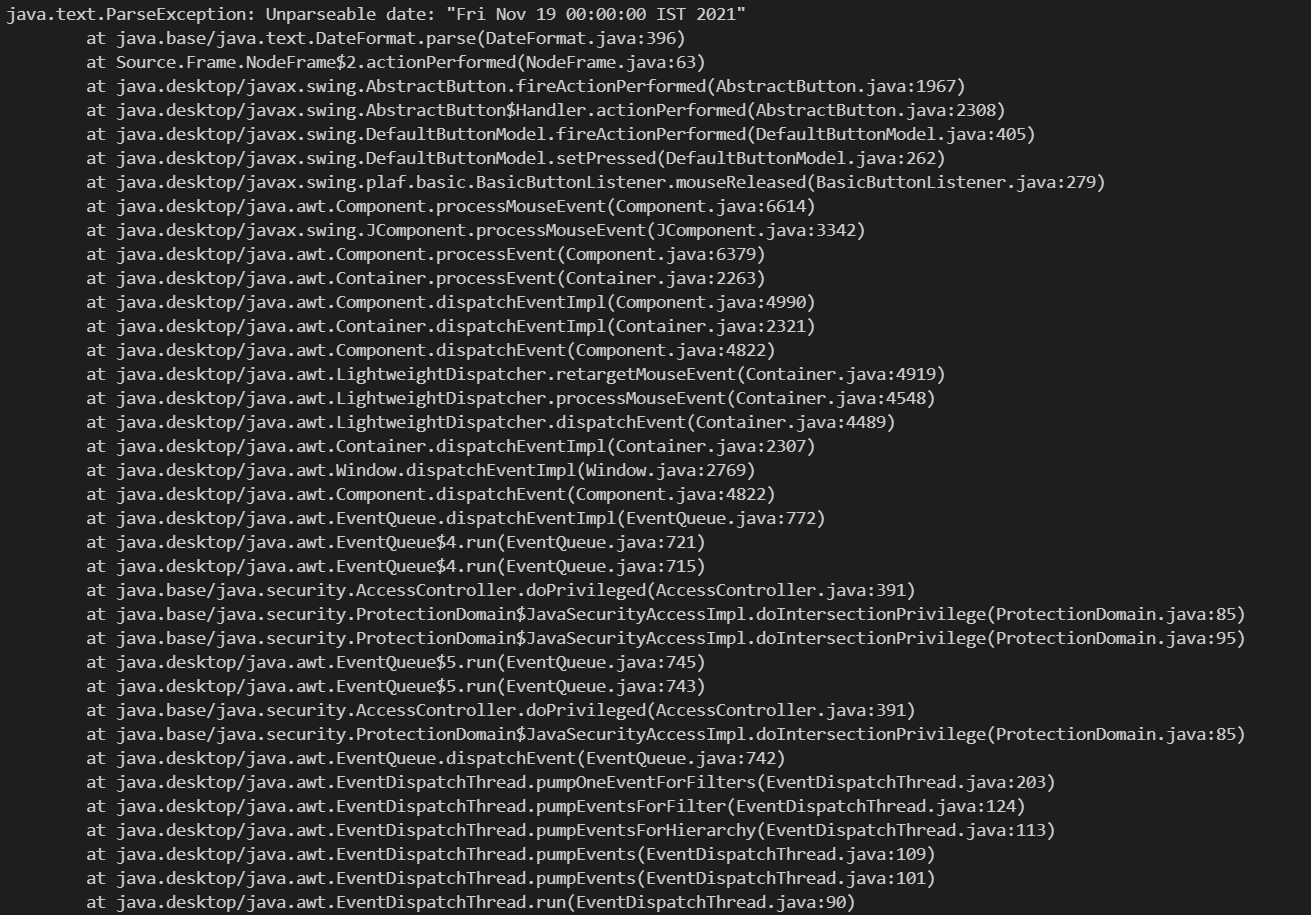


### 8) Save contents to file



### 9) Exception Handling

##### Invalid Date Format



##### File Not Found

