Library Management Application - Group Report

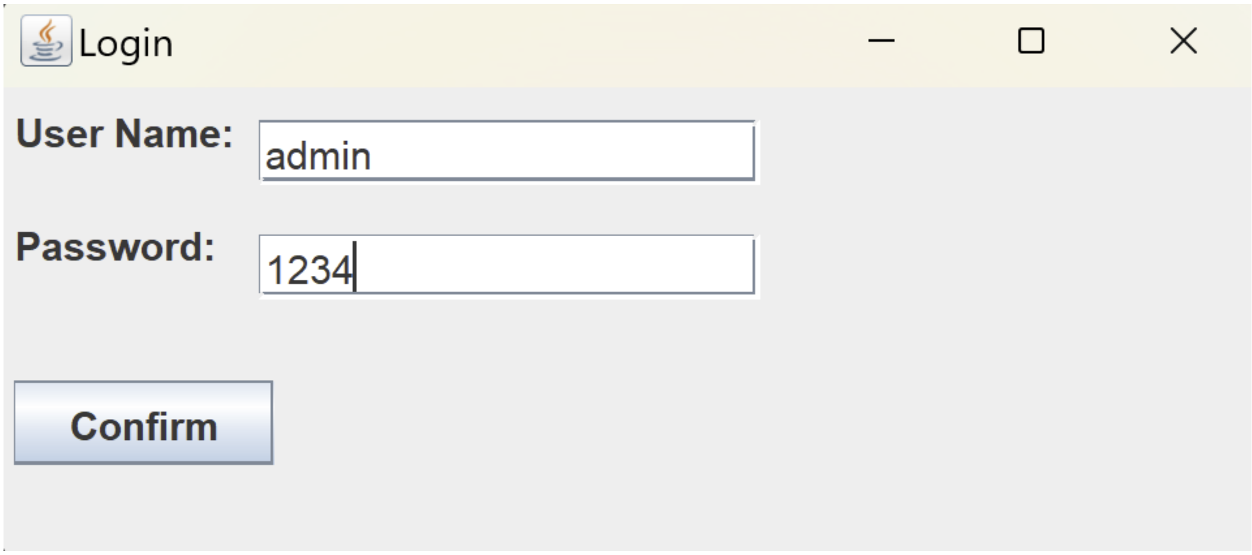
Section 1

Features	Implementation
a)	Fully implemented
b)	Fully implemented
c)	Fully implemented
d)	Fully implemented
e)	Fully implemented
f)	Fully implemented
g)	Fully implemented

Section 2

Login Window Functionality

The Menu window is secured with a "login" credential window.
The login details are “admin” as user name and “1234” as password.

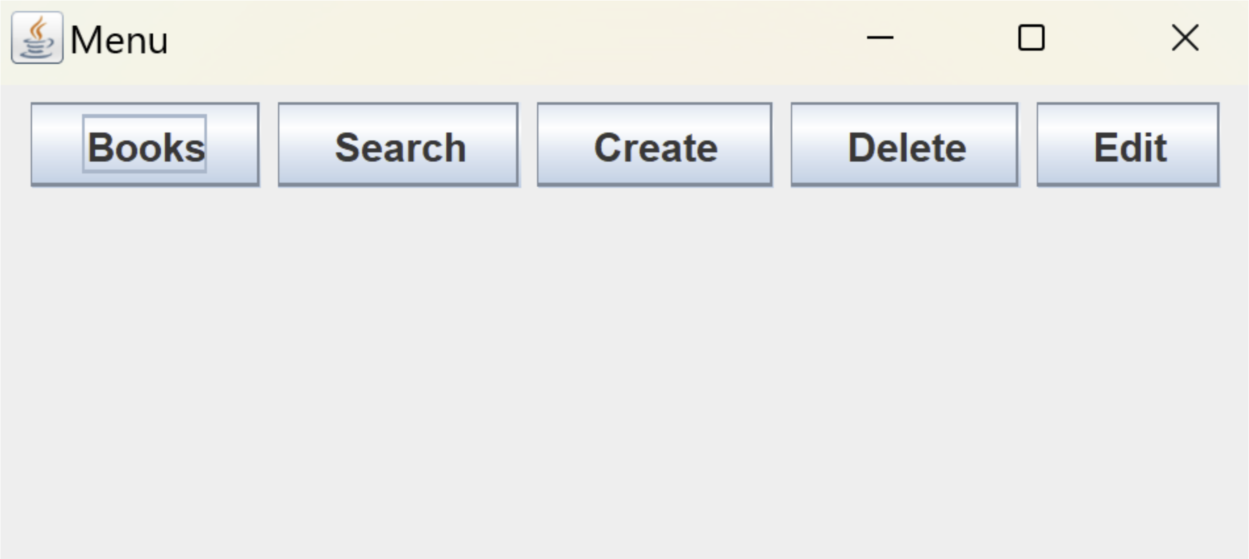


User Name: admin

Password: 1234

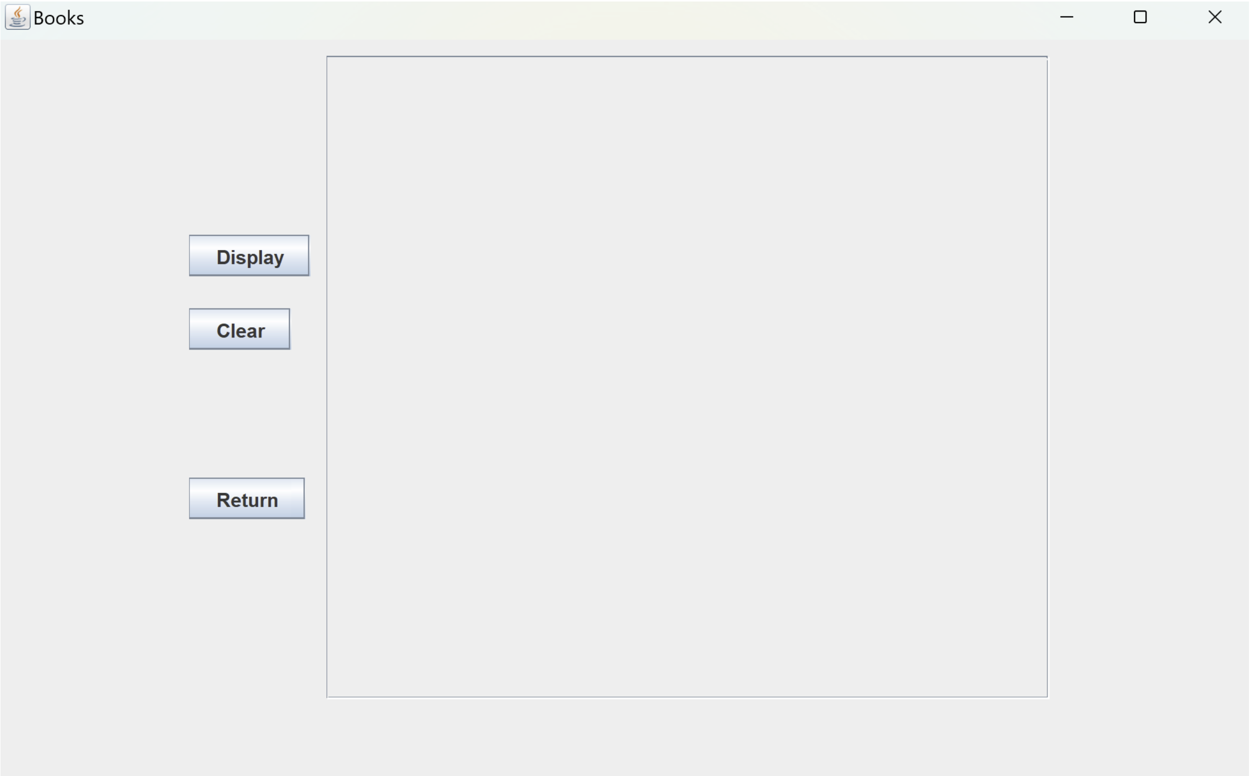
Menu Window Functionality

Main Menu Window after the login screen.

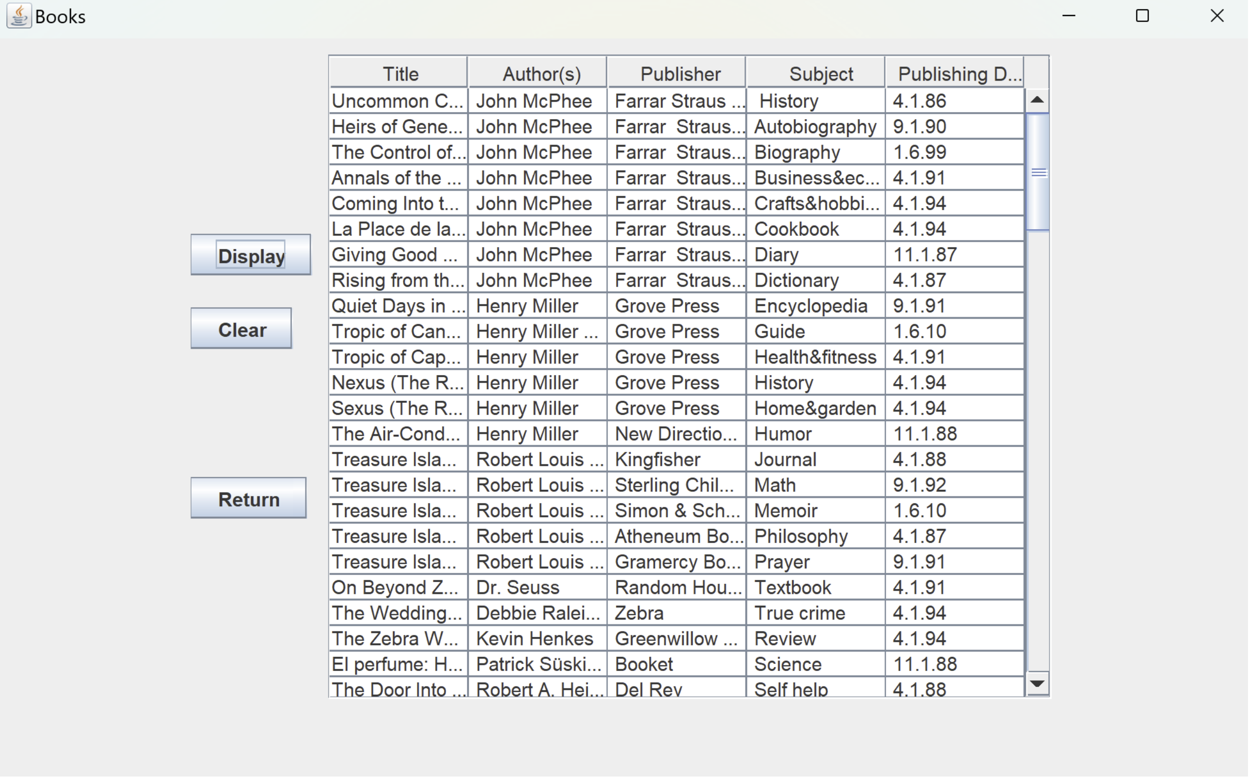


‘Books’ Window Functionality

‘Books’ button in the Menu opens a new window called ’Books’.

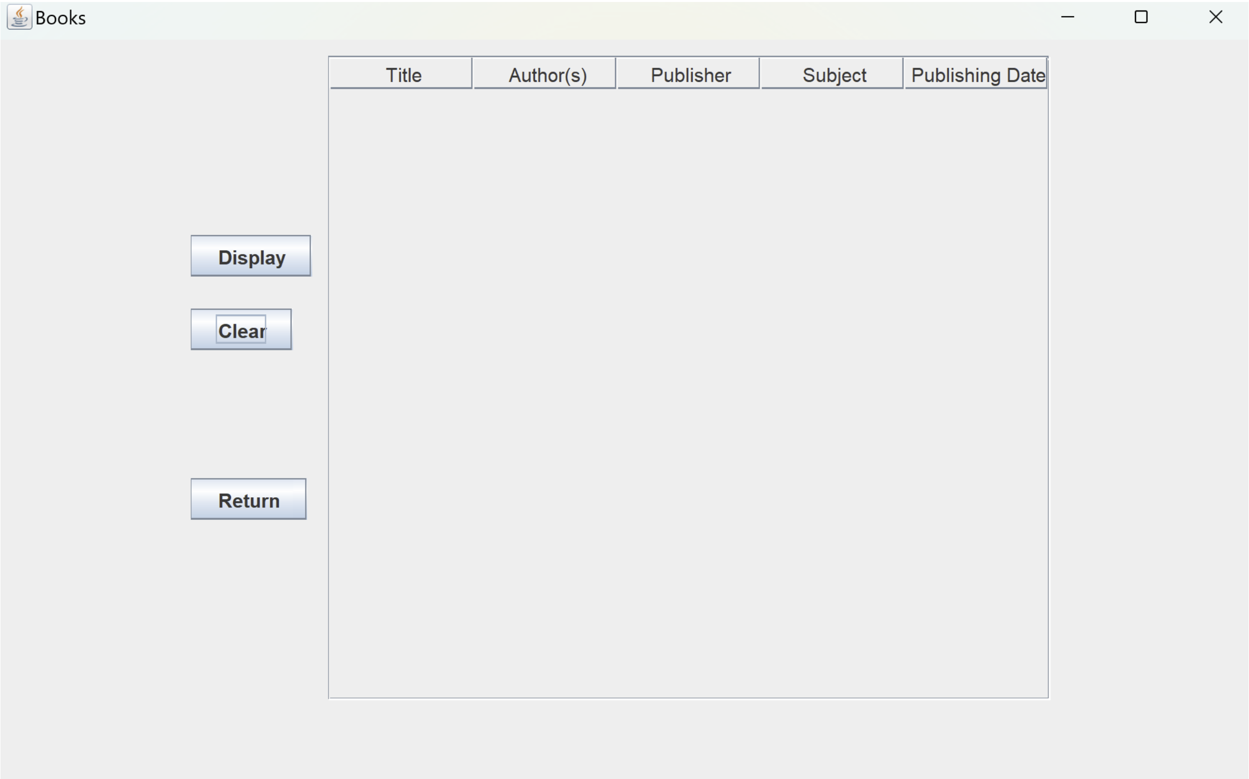


‘Display’ button in the ‘Books’ window displays the list of books from the ‘data.txt’ file.



Title	Author(s)	Publisher	Subject	Publishing D...
Uncommon C...	John McPhee	Farrar Straus ...	History	4.1.86
Heirs of Gene...	John McPhee	Farrar Straus...	Autobiography	9.1.90
The Control of...	John McPhee	Farrar Straus...	Biography	1.6.99
Annals of the ...	John McPhee	Farrar Straus...	Business&ec...	4.1.91
Coming Into t...	John McPhee	Farrar Straus...	Crafts&hobbi...	4.1.94
La Place de la...	John McPhee	Farrar Straus...	Cookbook	4.1.94
Giving Good ...	John McPhee	Farrar Straus...	Diary	11.1.87
Rising from th...	John McPhee	Farrar Straus...	Dictionary	4.1.87
Quiet Days in ...	Henry Miller	Grove Press	Encyclopedia	9.1.91
Tropic of Can...	Henry Miller ...	Grove Press	Guide	1.6.10
Tropic of Cap...	Henry Miller	Grove Press	Health&fitness	4.1.91
Nexus (The R...	Henry Miller	Grove Press	History	4.1.94
Sexus (The R...	Henry Miller	Grove Press	Home&garden	4.1.94
The Air-Cond...	Henry Miller	New Directio...	Humor	11.1.88
Treasure Isla...	Robert Louis ...	Kingfisher	Journal	4.1.88
Treasure Isla...	Robert Louis ...	Sterling Chil...	Math	9.1.92
Treasure Isla...	Robert Louis ...	Simon & Sch...	Memoir	1.6.10
Treasure Isla...	Robert Louis ...	Atheneum Bo...	Philosophy	4.1.87
Treasure Isla...	Robert Louis ...	Gramercy Bo...	Prayer	9.1.91
On Beyond Z...	Dr. Seuss	Random Hou...	Textbook	4.1.91
The Wedding...	Debbie Ralei...	Zebra	True crime	4.1.94
The Zebra W...	Kevin Henkes	Greenwillow ...	Review	4.1.94
El perfume: H...	Patrick Süski...	Booket	Science	11.1.88
The Door Into ...	Robert A. Hei...	Del Rey	Self help	4.1.88

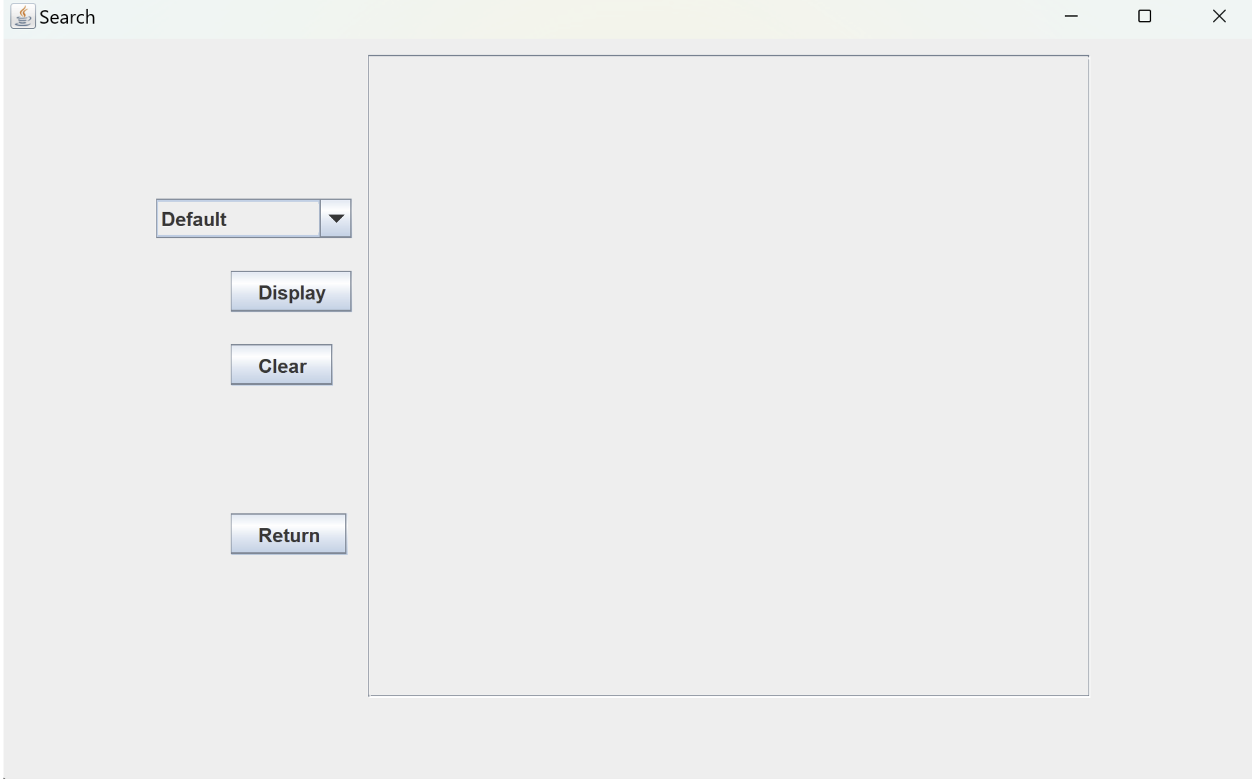
‘Clear’ button in the ‘Books’ window clears the displayed data in the table.
‘Return’ button reopens the ‘Menu’ window while closing the ‘Books’ window.



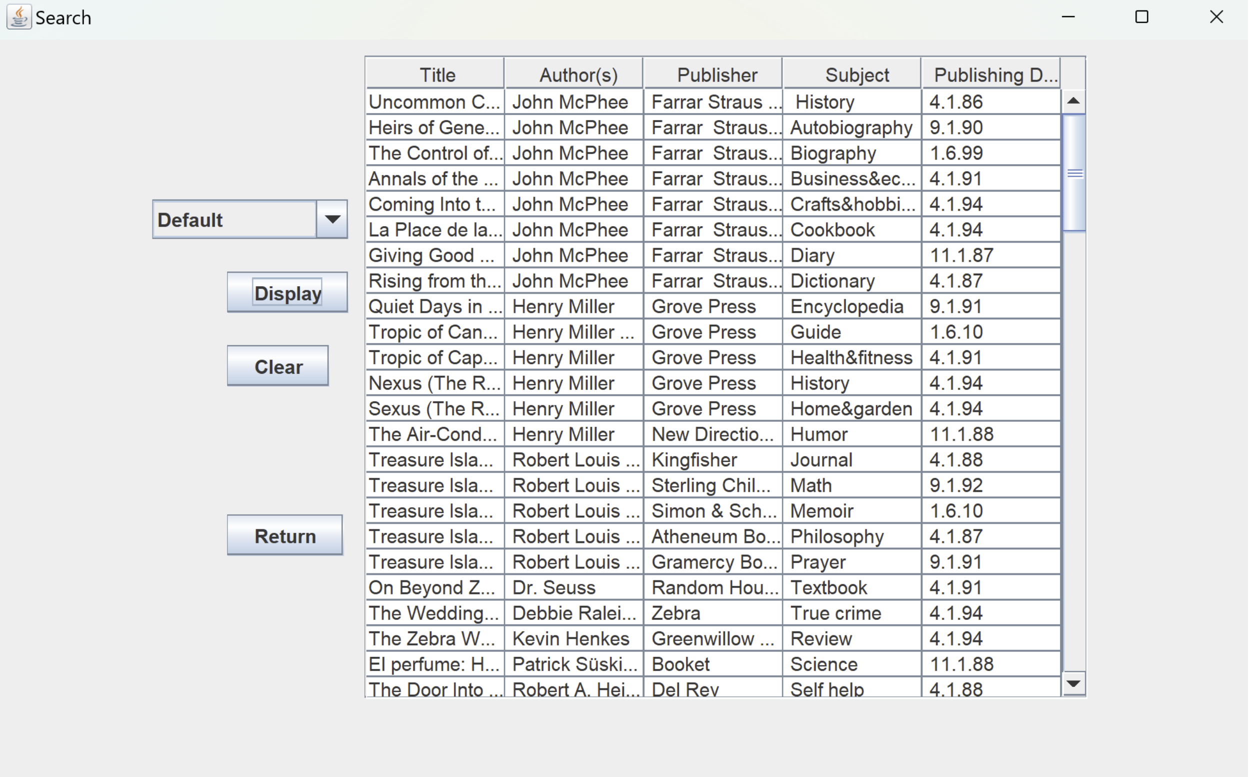
Title	Author(s)	Publisher	Subject	Publishing Date

“Search” Window Functionality

‘Search’ button in the Menu opens a new window called ’Search’.

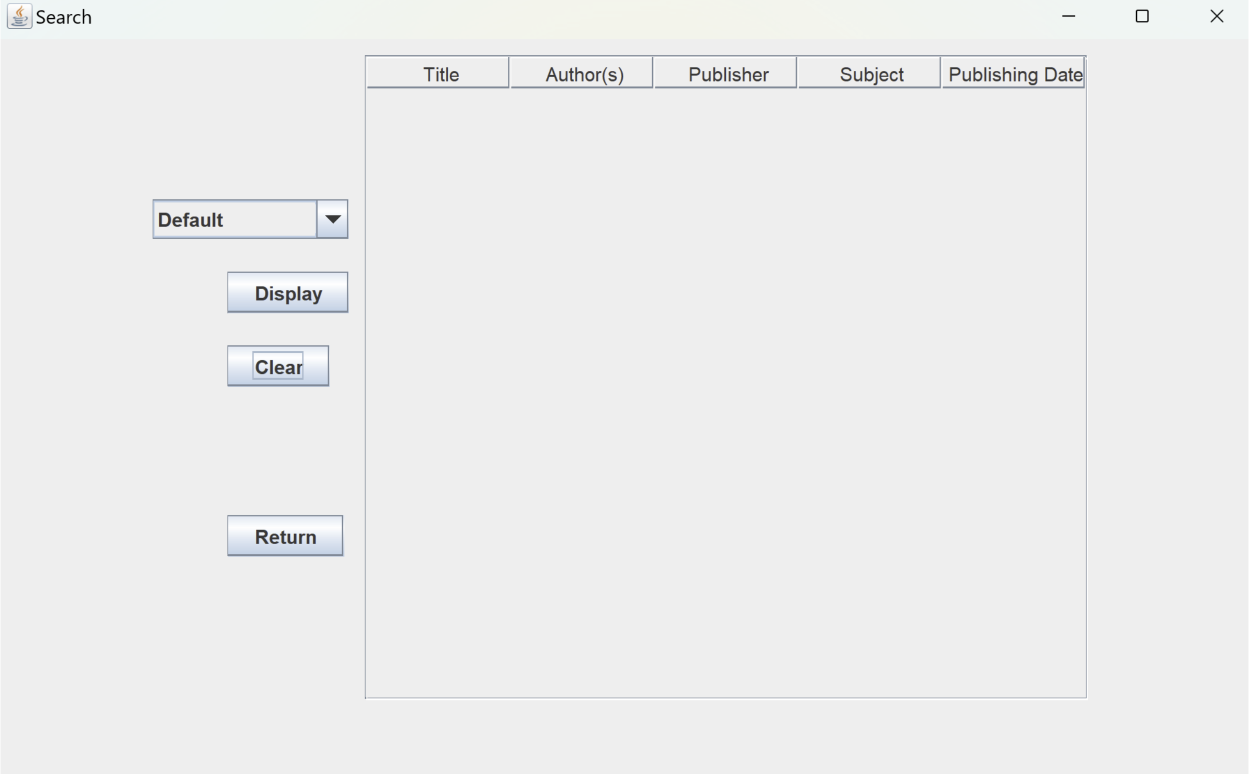


‘Display’ button in the ‘Search’ window displays the list of books from the ‘data.txt’ file. The displayed data must be cleared before displaying the default or sorted options.



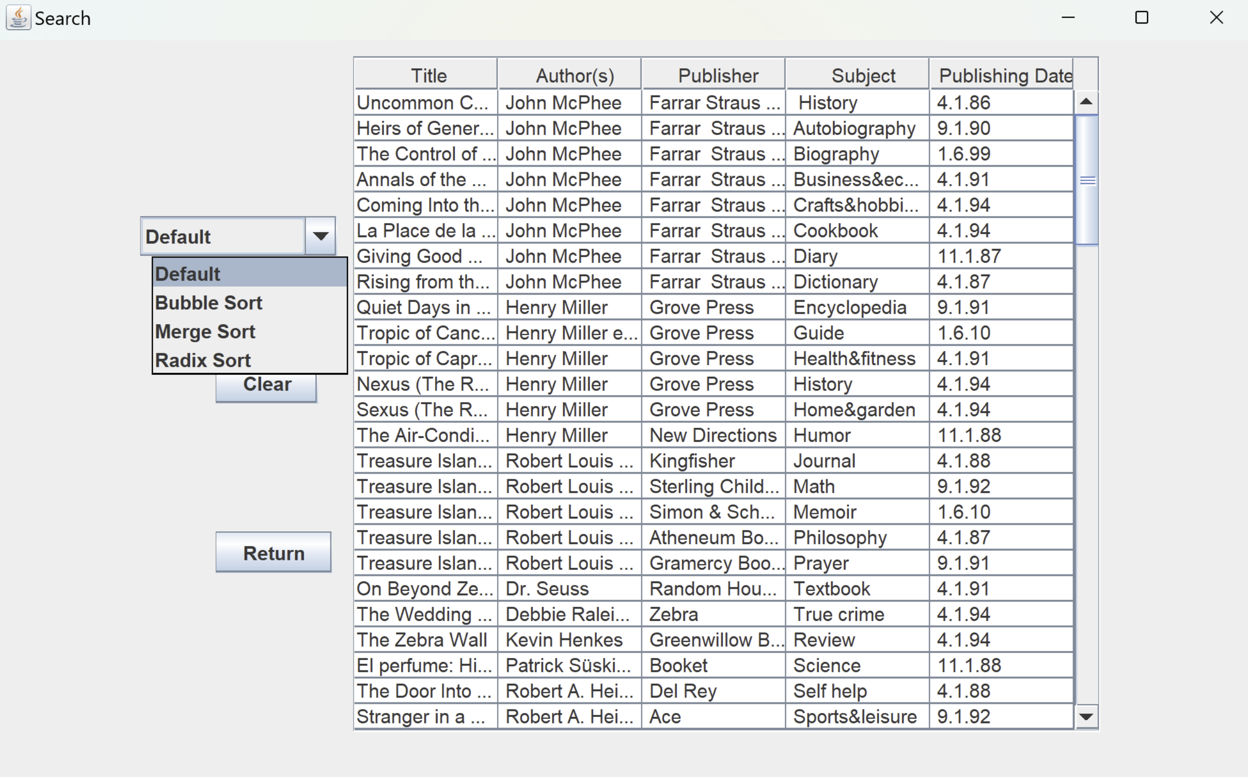
Title	Author(s)	Publisher	Subject	Publishing D...
Uncommon C...	John McPhee	Farrar Straus ...	History	4.1.86
Heirs of Gene...	John McPhee	Farrar Straus...	Autobiography	9.1.90
The Control of...	John McPhee	Farrar Straus...	Biography	1.6.99
Annals of the ...	John McPhee	Farrar Straus...	Business&ec...	4.1.91
Coming Into t...	John McPhee	Farrar Straus...	Crafts&hobbi...	4.1.94
La Place de la...	John McPhee	Farrar Straus...	Cookbook	4.1.94
Giving Good ...	John McPhee	Farrar Straus...	Diary	11.1.87
Rising from th...	John McPhee	Farrar Straus...	Dictionary	4.1.87
Quiet Days in ...	Henry Miller	Grove Press	Encyclopedia	9.1.91
Tropic of Can...	Henry Miller ...	Grove Press	Guide	1.6.10
Tropic of Cap...	Henry Miller	Grove Press	Health&fitness	4.1.91
Nexus (The R...	Henry Miller	Grove Press	History	4.1.94
Sexus (The R...	Henry Miller	Grove Press	Home&garden	4.1.94
The Air-Cond...	Henry Miller	New Directio...	Humor	11.1.88
Treasure Isla...	Robert Louis ...	Kingfisher	Journal	4.1.88
Treasure Isla...	Robert Louis ...	Sterling Chil...	Math	9.1.92
Treasure Isla...	Robert Louis ...	Simon & Sch...	Memoir	1.6.10
Treasure Isla...	Robert Louis ...	Atheneum Bo...	Philosophy	4.1.87
Treasure Isla...	Robert Louis ...	Gramercy Bo...	Prayer	9.1.91
On Beyond Z...	Dr. Seuss	Random Hou...	Textbook	4.1.91
The Wedding...	Debbie Ralei...	Zebra	True crime	4.1.94
The Zebra W...	Kevin Henkes	Greenwillow ...	Review	4.1.94
El perfume: H...	Patrick Süski...	Booket	Science	11.1.88
The Door Into ...	Robert A. Hei...	Del Rey	Self help	4.1.88

‘Clear’ button in the ‘Search’ window clears the displayed data in the table.



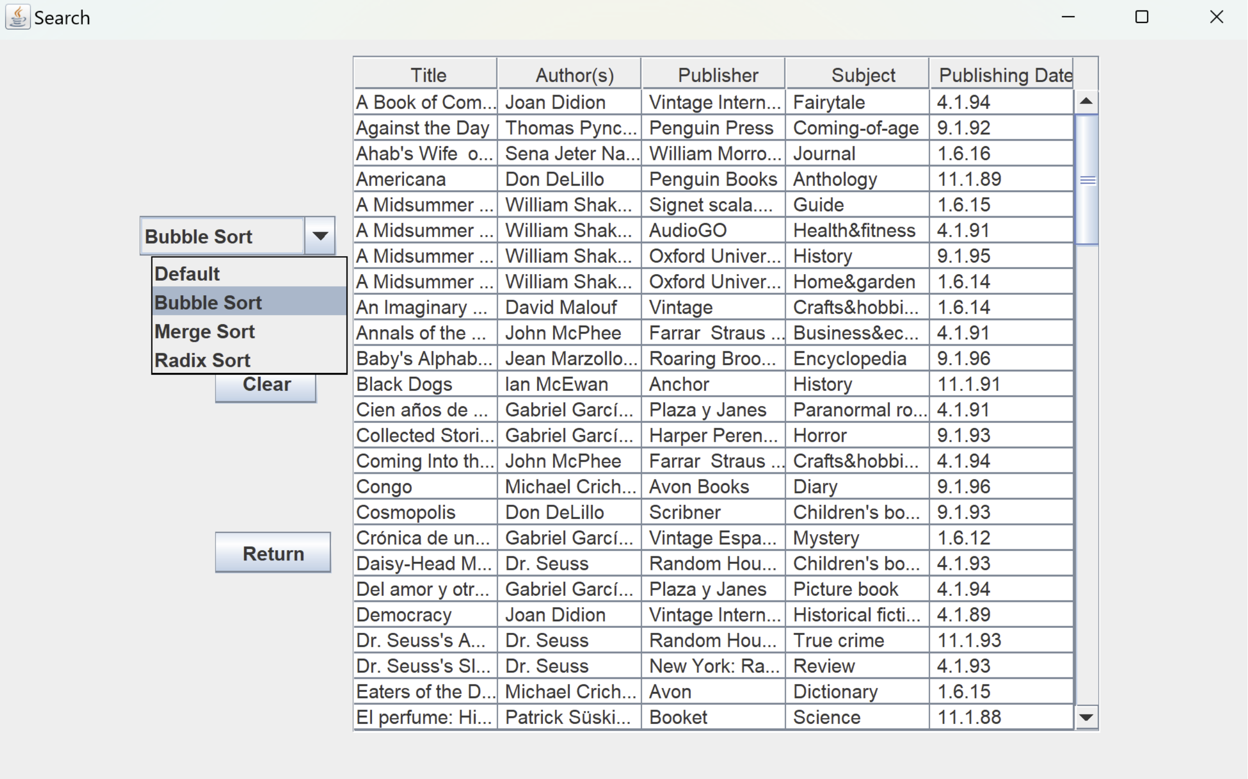
Title	Author(s)	Publisher	Subject	Publishing Date

‘Default’ choice in the drop-down menu displays the unsorted data from the ‘data.txt’ file.
The displayed data must be cleared before displaying the default or sorted options.



Title	Author(s)	Publisher	Subject	Publishing Date
Uncommon C...	John McPhee	Farrar Straus ...	History	4.1.86
Heirs of Gener...	John McPhee	Farrar Straus ...	Autobiography	9.1.90
The Control of ...	John McPhee	Farrar Straus ...	Biography	1.6.99
Annals of the ...	John McPhee	Farrar Straus ...	Business&ec...	4.1.91
Coming Into th...	John McPhee	Farrar Straus ...	Crafts&hobbi...	4.1.94
La Place de la ...	John McPhee	Farrar Straus ...	Cookbook	4.1.94
Giving Good ...	John McPhee	Farrar Straus ...	Diary	11.1.87
Rising from th...	John McPhee	Farrar Straus ...	Dictionary	4.1.87
Quiet Days in ...	Henry Miller	Grove Press	Encyclopedia	9.1.91
Tropic of Canc...	Henry Miller e...	Grove Press	Guide	1.6.10
Tropic of Capr...	Henry Miller	Grove Press	Health&fitness	4.1.91
Nexus (The R...	Henry Miller	Grove Press	History	4.1.94
Sexus (The R...	Henry Miller	Grove Press	Home&garden	4.1.94
The Air-Condi...	Henry Miller	New Directions	Humor	11.1.88
Treasure Islan...	Robert Louis ...	Kingfisher	Journal	4.1.88
Treasure Islan...	Robert Louis ...	Sterling Child...	Math	9.1.92
Treasure Islan...	Robert Louis ...	Simon & Sch...	Memoir	1.6.10
Treasure Islan...	Robert Louis ...	Atheneum Bo...	Philosophy	4.1.87
Treasure Islan...	Robert Louis ...	Gramercy Boo...	Prayer	9.1.91
On Beyond Ze...	Dr. Seuss	Random Hou...	Textbook	4.1.91
The Wedding ...	Debbie Ralei...	Zebra	True crime	4.1.94
The Zebra Wall	Kevin Henkes	Greenwillow B...	Review	4.1.94
El perfume: Hi...	Patrick Süski...	Booket	Science	11.1.88
The Door Into ...	Robert A. Hei...	Del Rey	Self help	4.1.88
Stranger in a ...	Robert A. Hei...	Ace	Sports&leisure	9.1.92

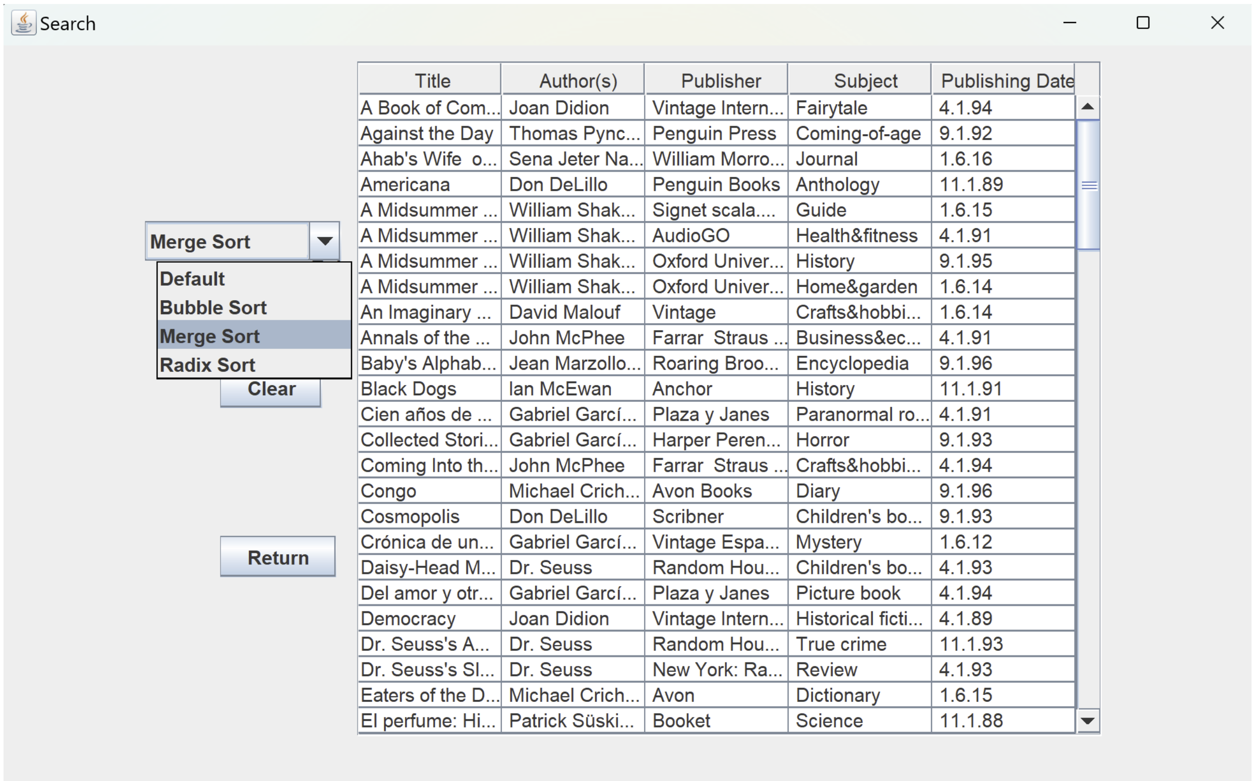
‘Bubble Sort’ choice in the drop-down menu displays the alphabetically sorted data from the ‘sorted.txt’ file.
The displayed data must be cleared before displaying the default or sorted options.



Title	Author(s)	Publisher	Subject	Publishing Date
A Book of Com...	Joan Didion	Vintage Intern...	Fairytale	4.1.94
Against the Day	Thomas Pync...	Penguin Press	Coming-of-age	9.1.92
Ahab's Wife o...	Sena Jeter Na...	William Morro...	Journal	1.6.16
Americana	Don DeLillo	Penguin Books	Anthology	11.1.89
A Midsummer ...	William Shak...	Signet scala....	Guide	1.6.15
A Midsummer ...	William Shak...	AudioGO	Health&fitness	4.1.91
A Midsummer ...	William Shak...	Oxford Univer...	History	9.1.95
A Midsummer ...	William Shak...	Oxford Univer...	Home&garden	1.6.14
An Imaginary ...	David Malouf	Vintage	Crafts&hobbi...	1.6.14
Annals of the ...	John McPhee	Farrar Straus ...	Business&ec...	4.1.91
Baby's Alphab...	Jean Marzollo...	Roaring Broo...	Encyclopedia	9.1.96
Black Dogs	Ian McEwan	Anchor	History	11.1.91
Cien años de ...	Gabriel Garcí...	Plaza y Janes	Paranormal ro...	4.1.91
Collected Stori...	Gabriel Garcí...	Harper Peren...	Horror	9.1.93
Coming Into th...	John McPhee	Farrar Straus ...	Crafts&hobbi...	4.1.94
Congo	Michael Crich...	Avon Books	Diary	9.1.96
Cosmopolis	Don DeLillo	Scribner	Children's bo...	9.1.93
Crónica de un...	Gabriel Garcí...	Vintage Espa...	Mystery	1.6.12
Daisy-Head M...	Dr. Seuss	Random Hou...	Children's bo...	4.1.93
Del amor y otr...	Gabriel Garcí...	Plaza y Janes	Picture book	4.1.94
Democracy	Joan Didion	Vintage Intern...	Historical ficti...	4.1.89
Dr. Seuss's A...	Dr. Seuss	Random Hou...	True crime	11.1.93
Dr. Seuss's Sl...	Dr. Seuss	New York: Ra...	Review	4.1.93
Eaters of the D...	Michael Crich...	Avon	Dictionary	1.6.15
El perfume: Hi...	Patrick Süski...	Booket	Science	11.1.88

‘Merge Sort’ choice in the drop-down menu displays the alphabetically sorted data from the ‘sorted.txt’ file.

The displayed data must be cleared before displaying the default or sorted options.

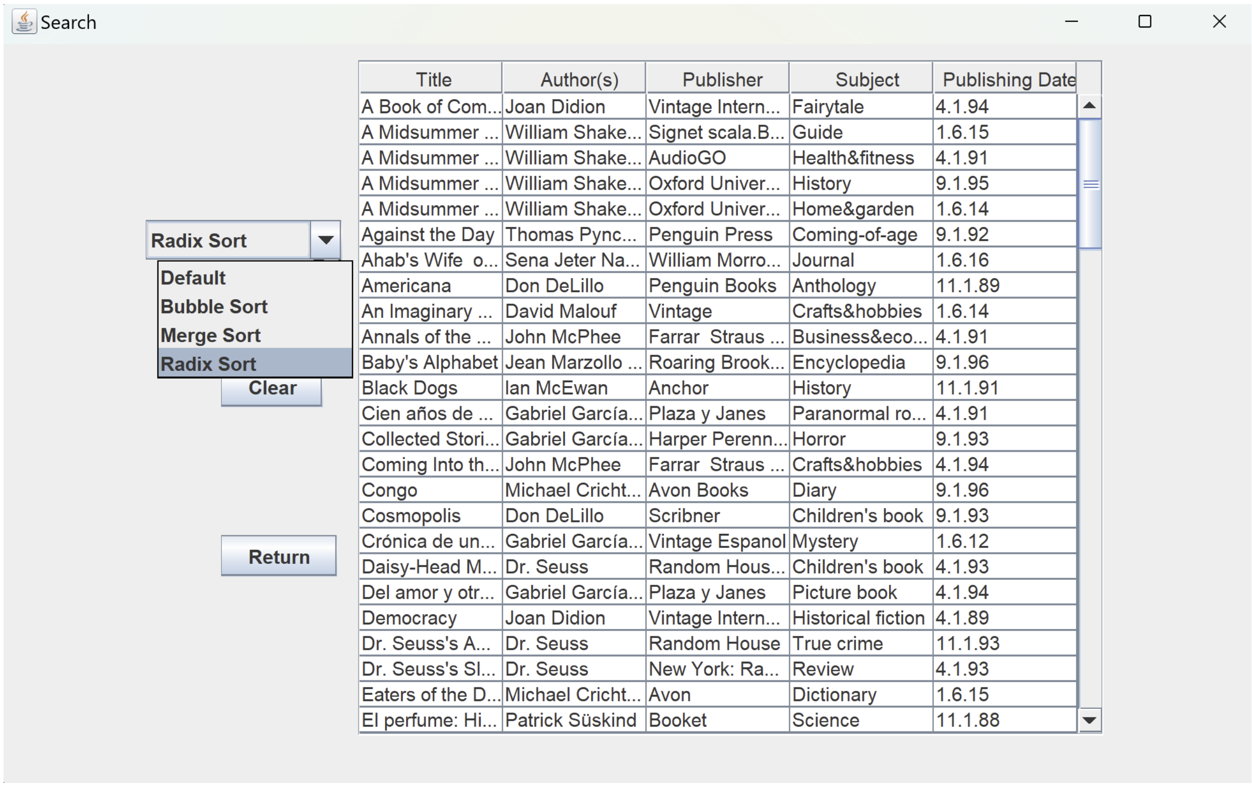


Title	Author(s)	Publisher	Subject	Publishing Date
A Book of Com...	Joan Didion	Vintage Intern...	Fairytale	4.1.94
Against the Day	Thomas Pync...	Penguin Press	Coming-of-age	9.1.92
Ahab's Wife o...	Sena Jeter Na...	William Morro...	Journal	1.6.16
Americana	Don DeLillo	Penguin Books	Anthology	11.1.89
A Midsummer ...	William Shak...	Signet scala....	Guide	1.6.15
A Midsummer ...	William Shak...	AudioGO	Health&fitness	4.1.91
A Midsummer ...	William Shak...	Oxford Univer...	History	9.1.95
A Midsummer ...	William Shak...	Oxford Univer...	Home&garden	1.6.14
An Imaginary ...	David Malouf	Vintage	Crafts&hobbi...	1.6.14
Annals of the ...	John McPhee	Farrar Straus ...	Business&ec...	4.1.91
Baby's Alphab...	Jean Marzollo...	Roaring Broo...	Encyclopedia	9.1.96
Black Dogs	Ian McEwan	Anchor	History	11.1.91
Cien años de ...	Gabriel Garcí...	Plaza y Janes	Paranormal ro...	4.1.91
Collected Stori...	Gabriel Garcí...	Harper Peren...	Horror	9.1.93
Coming Into th...	John McPhee	Farrar Straus ...	Crafts&hobbi...	4.1.94
Congo	Michael Crich...	Avon Books	Diary	9.1.96
Cosmopolis	Don DeLillo	Scribner	Children's bo...	9.1.93
Crónica de un...	Gabriel Garcí...	Vintage Espa...	Mystery	1.6.12
Daisy-Head M...	Dr. Seuss	Random Hou...	Children's bo...	4.1.93
Del amor y otr...	Gabriel Garcí...	Plaza y Janes	Picture book	4.1.94
Democracy	Joan Didion	Vintage Intern...	Historical ficti...	4.1.89
Dr. Seuss's A...	Dr. Seuss	Random Hou...	True crime	11.1.93
Dr. Seuss's Sl...	Dr. Seuss	New York: Ra...	Review	4.1.93
Eaters of the D...	Michael Crich...	Avon	Dictionary	1.6.15
El perfume: Hi...	Patrick Süski...	Booket	Science	11.1.88

‘Radix Sort’ choice in the drop-down menu displays the alphabetically sorted data from the ‘sorted.txt’ file.

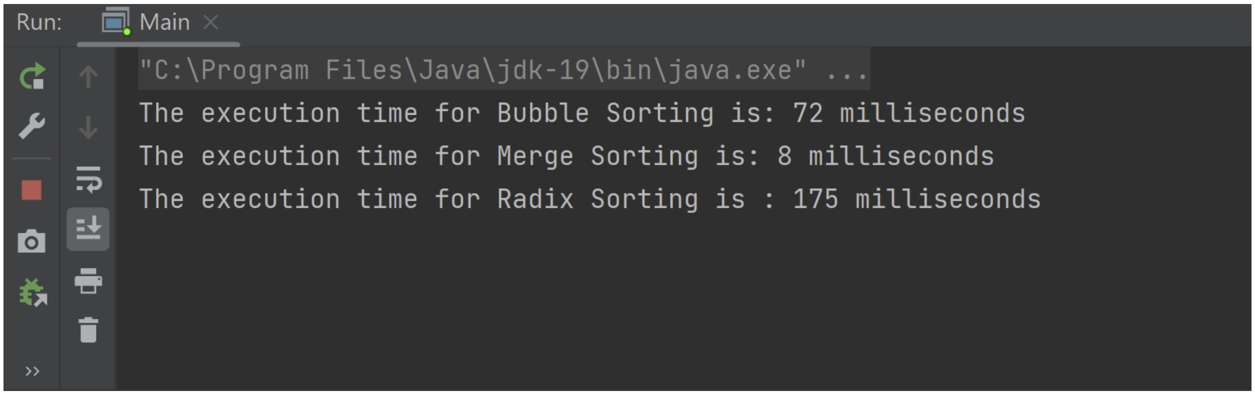
The displayed data must be cleared before displaying the default or sorted options.

‘Return’ button reopens the ‘Menu’ window while closing the ‘Search’ window.



Title	Author(s)	Publisher	Subject	Publishing Date
A Book of Com...	Joan Didion	Vintage Intern...	Fairytale	4.1.94
A Midsummer ...	William Shake...	Signet scala.B...	Guide	1.6.15
A Midsummer ...	William Shake...	AudioGO	Health&fitness	4.1.91
A Midsummer ...	William Shake...	Oxford Univer...	History	9.1.95
A Midsummer ...	William Shake...	Oxford Univer...	Home&garden	1.6.14
Against the Day	Thomas Pync...	Penguin Press	Coming-of-age	9.1.92
Ahab's Wife o...	Sena Jeter Na...	William Morro...	Journal	1.6.16
Americana	Don DeLillo	Penguin Books	Anthology	11.1.89
An Imaginary ...	David Malouf	Vintage	Crafts&hobbies	1.6.14
Annals of the ...	John McPhee	Farrar Straus ...	Business&eco...	4.1.91
Baby's Alphabet	Jean Marzollo ...	Roaring Brook...	Encyclopedia	9.1.96
Black Dogs	Ian McEwan	Anchor	History	11.1.91
Cien años de ...	Gabriel García...	Plaza y Janes	Paranormal ro...	4.1.91
Collected Stori...	Gabriel García...	Harper Perenn...	Horror	9.1.93
Coming Into th...	John McPhee	Farrar Straus ...	Crafts&hobbies	4.1.94
Congo	Michael Cricht...	Avon Books	Diary	9.1.96
Cosmopolis	Don DeLillo	Scribner	Children's book	9.1.93
Crónica de un...	Gabriel García...	Vintage Espanol	Mystery	1.6.12
Daisy-Head M...	Dr. Seuss	Random Hous...	Children's book	4.1.93
Del amor y otr...	Gabriel García...	Plaza y Janes	Picture book	4.1.94
Democracy	Joan Didion	Vintage Intern...	Historical fiction	4.1.89
Dr. Seuss's A...	Dr. Seuss	Random House	True crime	11.1.93
Dr. Seuss's Sl...	Dr. Seuss	New York: Ra...	Review	4.1.93
Eaters of the D...	Michael Cricht...	Avon	Dictionary	1.6.15
El perfume: Hi...	Patrick Süskind	Booket	Science	11.1.88

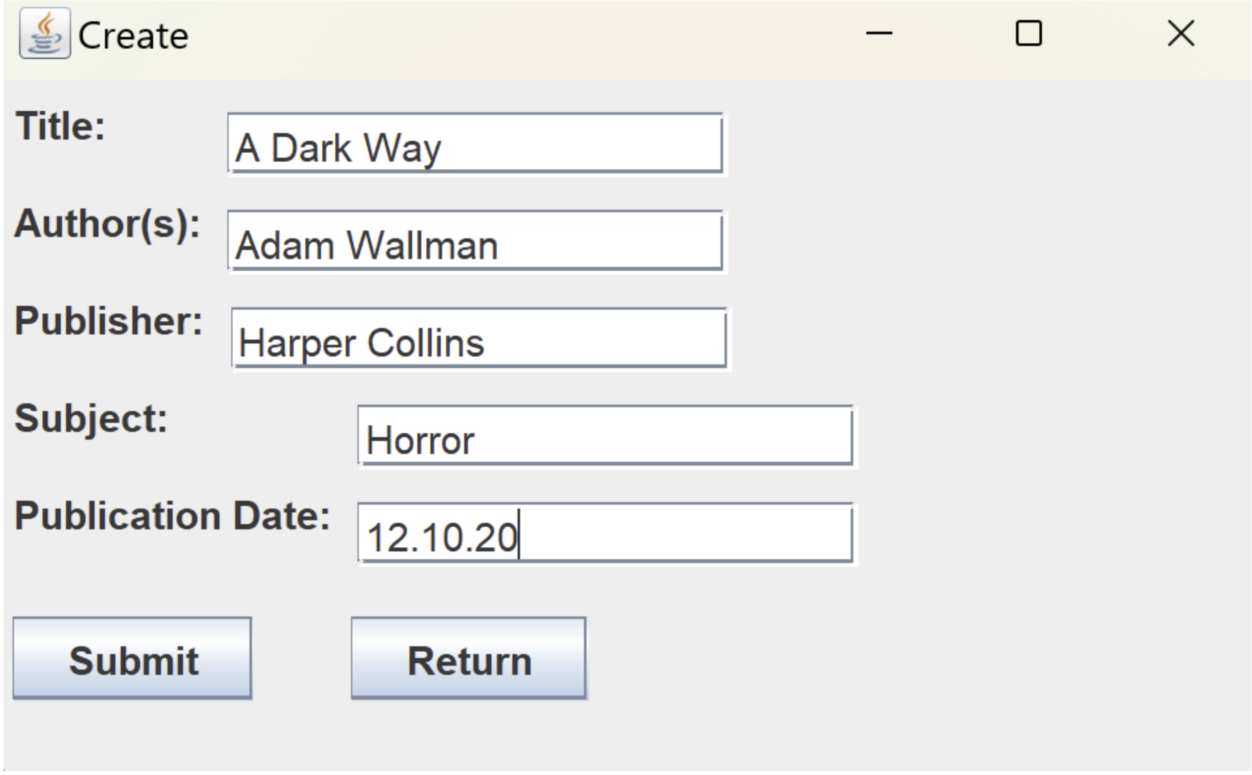
The execution time of the different sorting algorithms are shown below in the terminal.



```
"C:\Program Files\Java\jdk-19\bin\java.exe" ...
The execution time for Bubble Sorting is: 72 milliseconds
The execution time for Merge Sorting is: 8 milliseconds
The execution time for Radix Sorting is : 175 milliseconds
```

“Create” Window Functionality

‘Search’ button in the Menu opens a new window called ’Search’.
The details of the book can be entered into the relevant spaces.
‘Return’ button reopens the ‘Menu’ window while closing the ‘Create’ window.



Title: A Dark Way

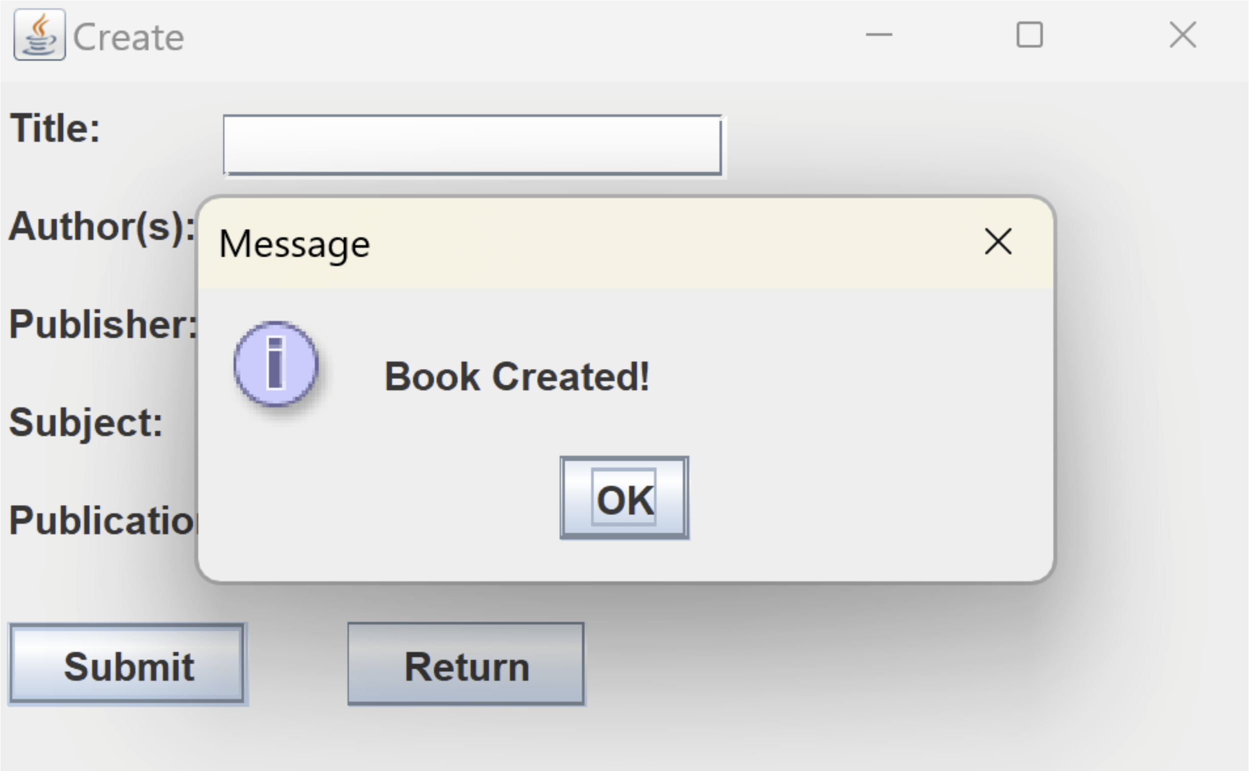
Author(s): Adam Wallman

Publisher: Harper Collins

Subject: Horror

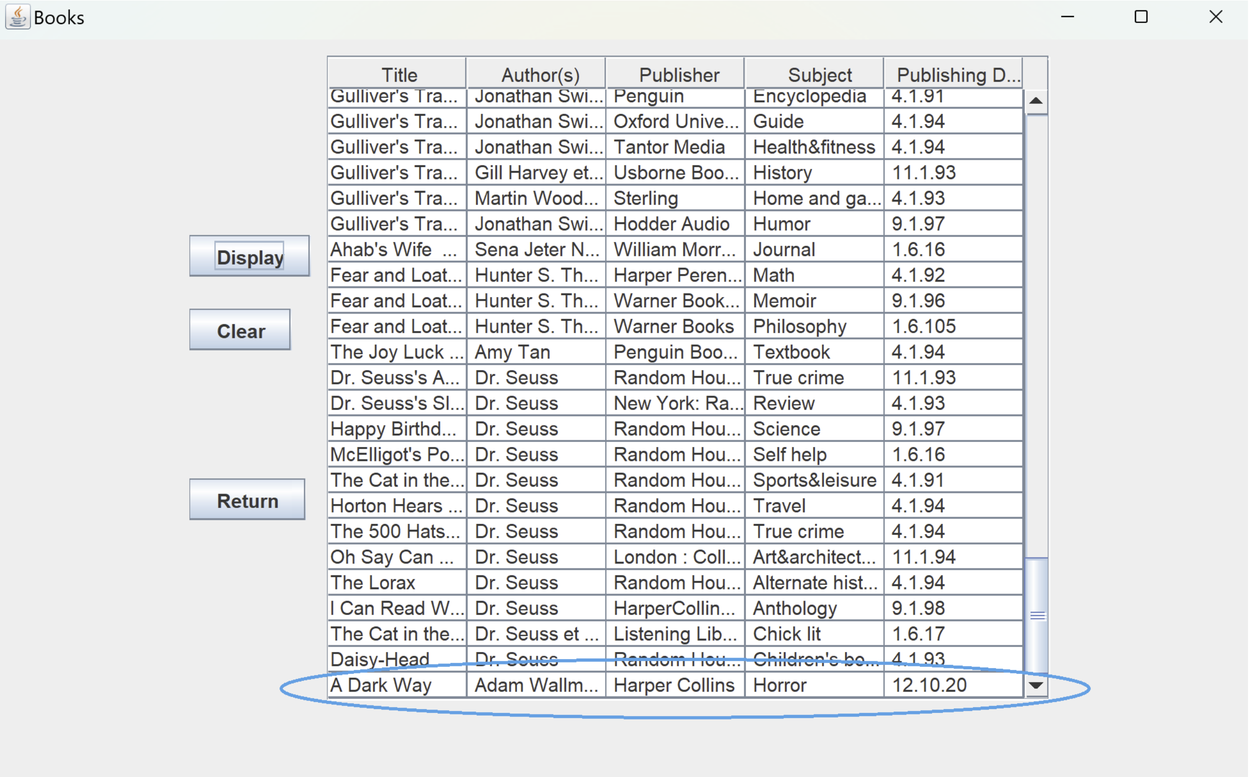
Publication Date: 12.10.20

A pop-up window will be shown upon the successful addition of the book into the ‘data.txt’ file.



Book Created!

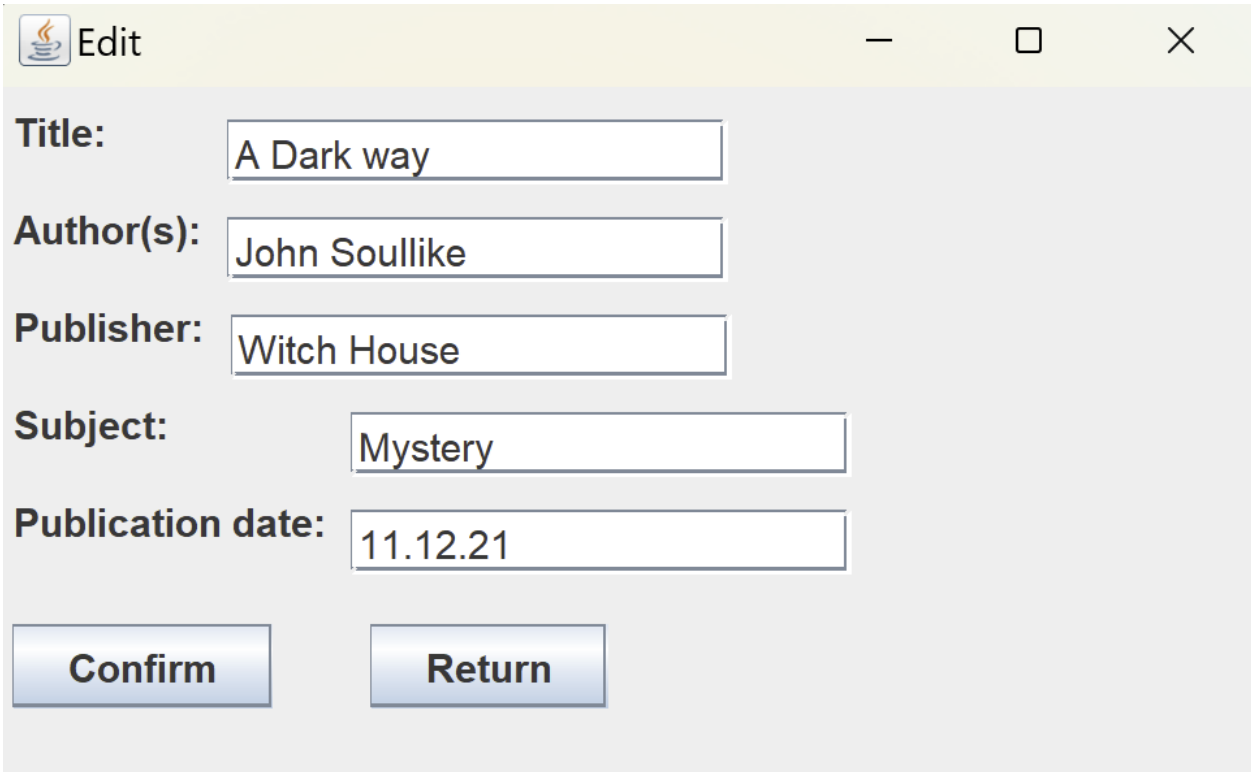
The newly added book will be shown at the end of the table and can be accessed from the ‘Books’ window.



Title	Author(s)	Publisher	Subject	Publishing D...
Gulliver's Tra...	Jonathan Swi...	Penguin	Encyclopedia	4.1.91
Gulliver's Tra...	Jonathan Swi...	Oxford Unive...	Guide	4.1.94
Gulliver's Tra...	Jonathan Swi...	Tantor Media	Health&fitness	4.1.94
Gulliver's Tra...	Gill Harvey et...	Usborne Boo...	History	11.1.93
Gulliver's Tra...	Martin Wood...	Sterling	Home and ga...	4.1.93
Gulliver's Tra...	Jonathan Swi...	Hodder Audio	Humor	9.1.97
Ahab's Wife ...	Sena Jeter N...	William Morr...	Journal	1.6.16
Fear and Loat...	Hunter S. Th...	Harper Peren...	Math	4.1.92
Fear and Loat...	Hunter S. Th...	Warner Book...	Memoir	9.1.96
Fear and Loat...	Hunter S. Th...	Warner Books	Philosophy	1.6.105
The Joy Luck ...	Amy Tan	Penguin Boo...	Textbook	4.1.94
Dr. Seuss's A...	Dr. Seuss	Random Hou...	True crime	11.1.93
Dr. Seuss's Sl...	Dr. Seuss	New York: Ra...	Review	4.1.93
Happy Birthd...	Dr. Seuss	Random Hou...	Science	9.1.97
McElligot's Po...	Dr. Seuss	Random Hou...	Self help	1.6.16
The Cat in the...	Dr. Seuss	Random Hou...	Sports&leisure	4.1.91
Horton Hears ...	Dr. Seuss	Random Hou...	Travel	4.1.94
The 500 Hats...	Dr. Seuss	Random Hou...	True crime	4.1.94
Oh Say Can ...	Dr. Seuss	London : Coll...	Art&architect...	11.1.94
The Lorax	Dr. Seuss	Random Hou...	Alternate hist...	4.1.94
I Can Read W...	Dr. Seuss	HarperCollin...	Anthology	9.1.98
The Cat in the...	Dr. Seuss et ...	Listening Lib...	Chick lit	1.6.17
Daisy-Head	Dr. Seuss	Random Hou...	Children's bo...	4.1.93
A Dark Way	Adam Wallm...	Harper Collins	Horror	12.10.20

“Edit” Window Functionality

‘Edit’ button in the Menu opens a new window called ’Edit’.
The name of the book to edit can be entered following with the new information related to the book.
‘Return’ button reopens the ‘Menu’ window while closing the ‘Create’ window.



Title: A Dark way

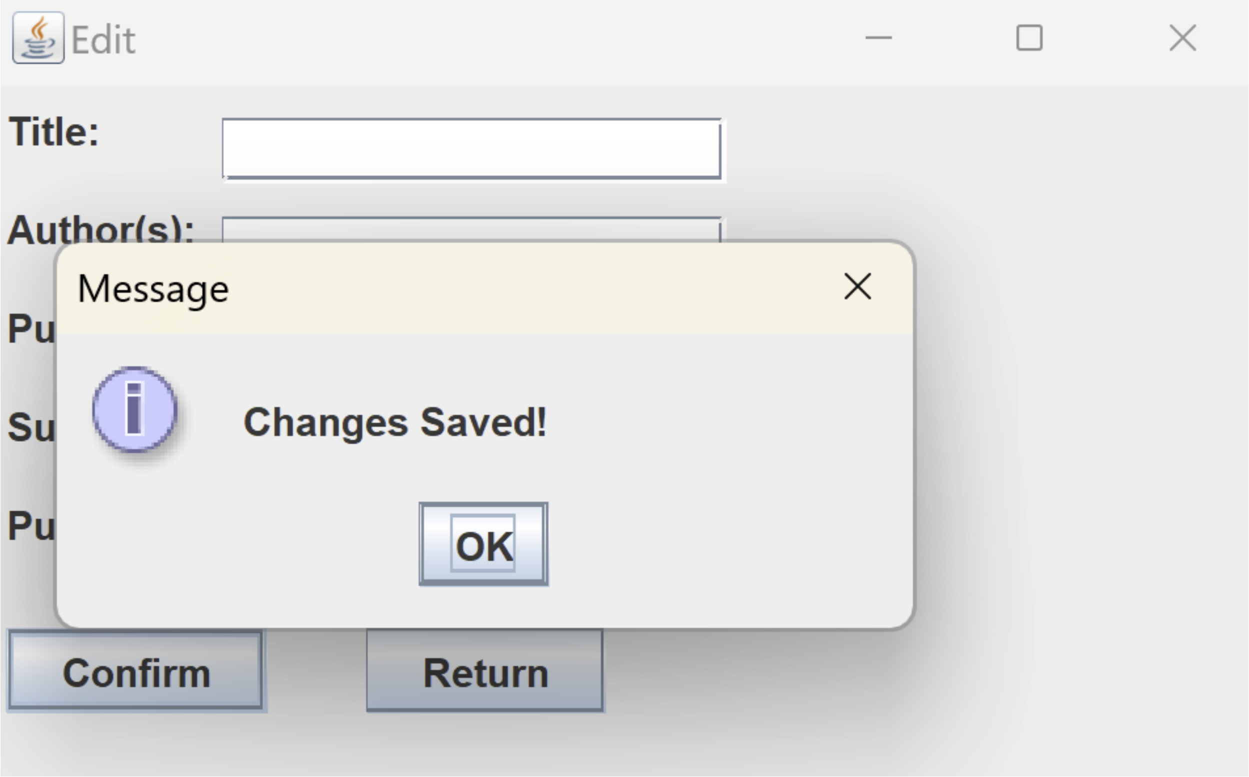
Author(s): John Soullike

Publisher: Witch House

Subject: Mystery

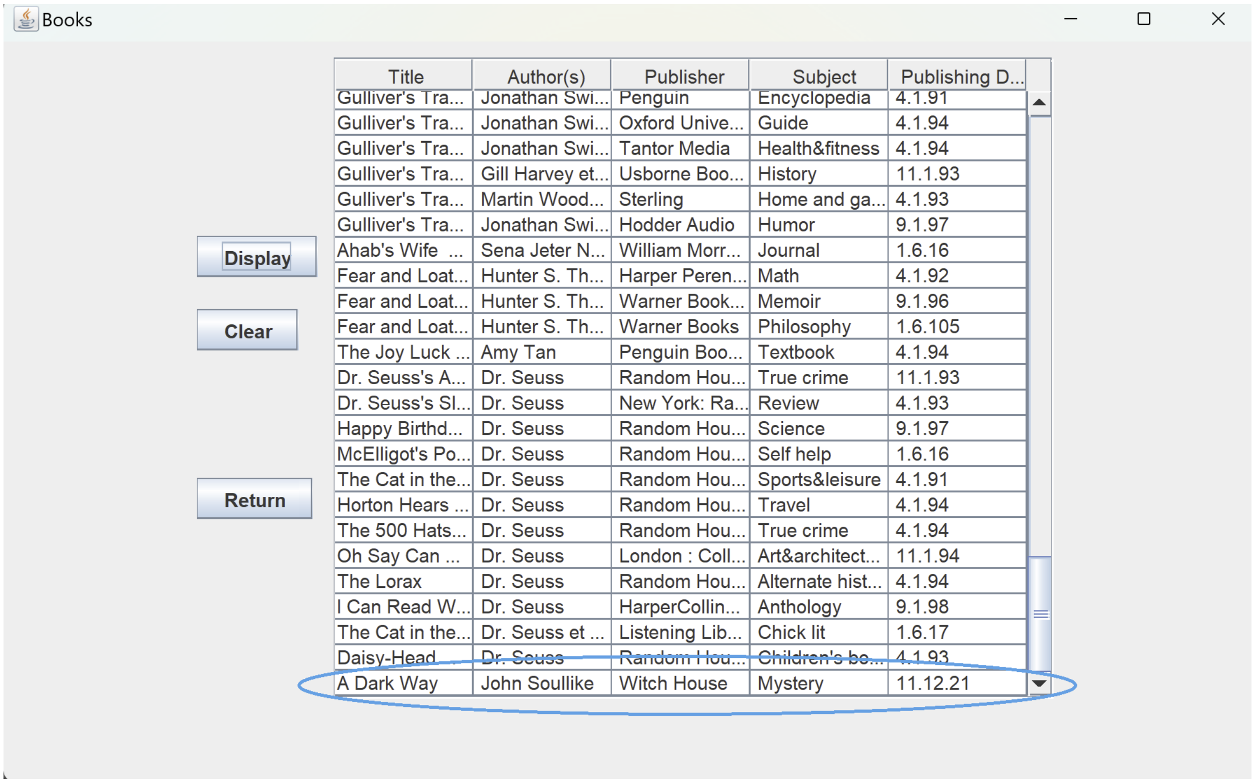
Publication date: 11.12.21

A pop-up window will be shown upon the successful edit of the book into the ‘data.txt’ file.



Changes Saved!

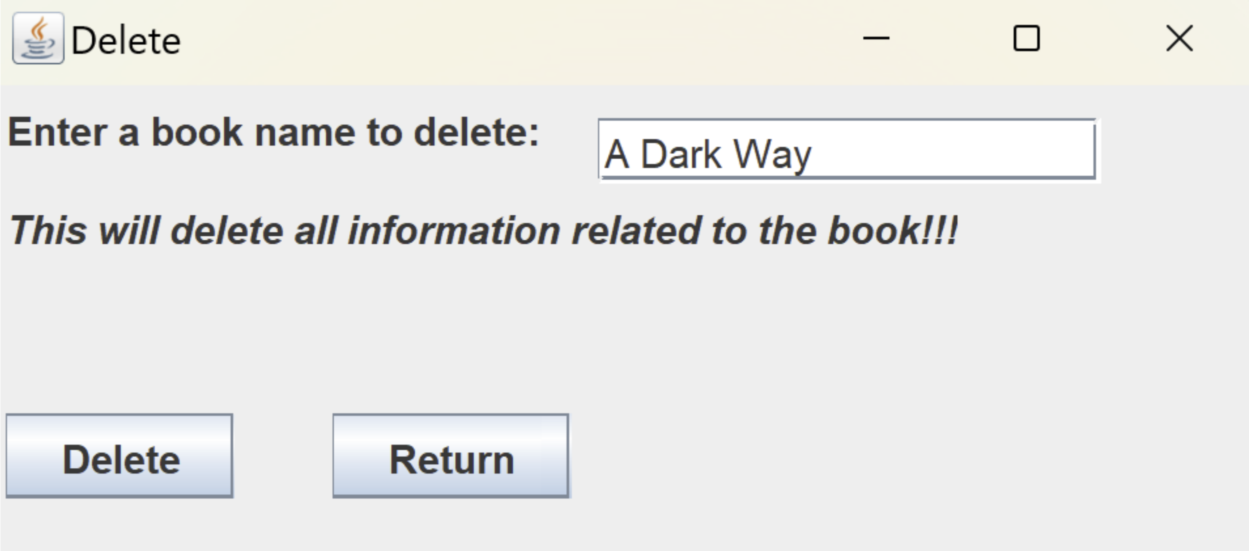
The details of the edited book will be shown and can be accessed from the ‘Books’ window.



Title	Author(s)	Publisher	Subject	Publishing D...
Gulliver's Tra...	Jonathan Swi...	Penguin	Encyclopedia	4.1.91
Gulliver's Tra...	Jonathan Swi...	Oxford Unive...	Guide	4.1.94
Gulliver's Tra...	Jonathan Swi...	Tantor Media	Health&fitness	4.1.94
Gulliver's Tra...	Gill Harvey et...	Usborne Boo...	History	11.1.93
Gulliver's Tra...	Martin Wood...	Sterling	Home and ga...	4.1.93
Gulliver's Tra...	Jonathan Swi...	Hodder Audio	Humor	9.1.97
Ahab's Wife ...	Sena Jeter N...	William Morr...	Journal	1.6.16
Fear and Loat...	Hunter S. Th...	Harper Peren...	Math	4.1.92
Fear and Loat...	Hunter S. Th...	Warner Book...	Memoir	9.1.96
Fear and Loat...	Hunter S. Th...	Warner Books	Philosophy	1.6.105
The Joy Luck ...	Amy Tan	Penguin Boo...	Textbook	4.1.94
Dr. Seuss's A...	Dr. Seuss	Random Hou...	True crime	11.1.93
Dr. Seuss's Sl...	Dr. Seuss	New York: Ra...	Review	4.1.93
Happy Birthd...	Dr. Seuss	Random Hou...	Science	9.1.97
McElligot's Po...	Dr. Seuss	Random Hou...	Self help	1.6.16
The Cat in the...	Dr. Seuss	Random Hou...	Sports&leisure	4.1.91
Horton Hears ...	Dr. Seuss	Random Hou...	Travel	4.1.94
The 500 Hats...	Dr. Seuss	Random Hou...	True crime	4.1.94
Oh Say Can ...	Dr. Seuss	London : Coll...	Art&architect...	11.1.94
The Lorax	Dr. Seuss	Random Hou...	Alternate hist...	4.1.94
I Can Read W...	Dr. Seuss	HarperCollin...	Anthology	9.1.98
The Cat in the...	Dr. Seuss et ...	Listening Lib...	Chick lit	1.6.17
Daisy-Head ...	Dr. Seuss	Random Hou...	Children's bo...	4.1.93
A Dark Way	John Soullike	Witch House	Mystery	11.12.21

“Delete” Window Functionality

‘Delete’ button in the Menu opens a new window called ’Delete’.
All the information about a book can be deleted by specifying the book name in the filed.
‘Return’ button reopens the ‘Menu’ window while closing the ‘Create’ window.

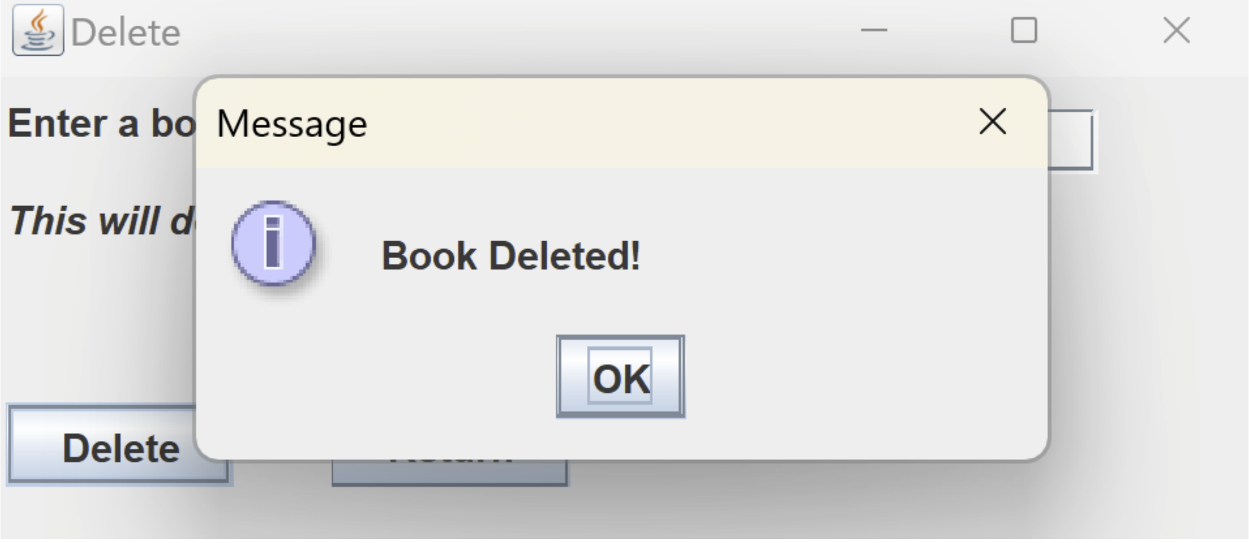


Delete

Enter a book name to delete: A Dark Way

This will delete all information related to the book!!!

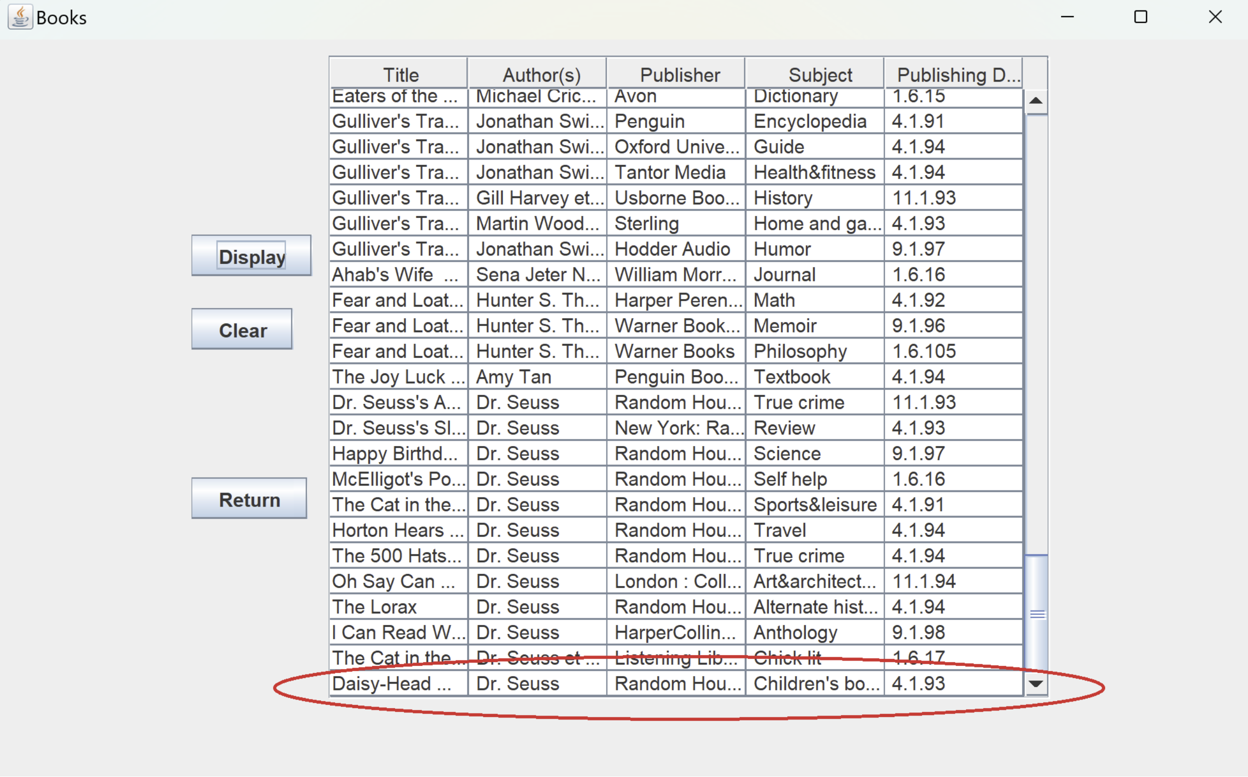
A pop-up window will be shown upon the successful deletion of the book from the ‘data.txt’ file.



Message

Book Deleted!

All the details about the book will be removed from the ‘data.txt’.



Title	Author(s)	Publisher	Subject	Publishing D...
Eaters of the ...	Michael Cric...	Avon	Dictionary	1.6.15
Gulliver's Tra...	Jonathan Swi...	Penguin	Encyclopedia	4.1.91
Gulliver's Tra...	Jonathan Swi...	Oxford Unive...	Guide	4.1.94
Gulliver's Tra...	Jonathan Swi...	Tantor Media	Health&fitness	4.1.94
Gulliver's Tra...	Gill Harvey et...	Usborne Boo...	History	11.1.93
Gulliver's Tra...	Martin Wood...	Sterling	Home and ga...	4.1.93
Gulliver's Tra...	Jonathan Swi...	Hodder Audio	Humor	9.1.97
Ahab's Wife ...	Sena Jeter N...	William Morr...	Journal	1.6.16
Fear and Loat...	Hunter S. Th...	Harper Peren...	Math	4.1.92
Fear and Loat...	Hunter S. Th...	Warner Book...	Memoir	9.1.96
Fear and Loat...	Hunter S. Th...	Warner Books	Philosophy	1.6.105
The Joy Luck ...	Amy Tan	Penguin Boo...	Textbook	4.1.94
Dr. Seuss's A...	Dr. Seuss	Random Hou...	True crime	11.1.93
Dr. Seuss's Sl...	Dr. Seuss	New York: Ra...	Review	4.1.93
Happy Birthd...	Dr. Seuss	Random Hou...	Science	9.1.97
McElligot's Po...	Dr. Seuss	Random Hou...	Self help	1.6.16
The Cat in the...	Dr. Seuss	Random Hou...	Sports&leisure	4.1.91
Horton Hears ...	Dr. Seuss	Random Hou...	Travel	4.1.94
The 500 Hats...	Dr. Seuss	Random Hou...	True crime	4.1.94
Oh Say Can ...	Dr. Seuss	London : Coll...	Art&architect...	11.1.94
The Lorax	Dr. Seuss	Random Hou...	Alternate hist...	4.1.94
I Can Read W...	Dr. Seuss	HarperCollin...	Anthology	9.1.98
The Cat in the...	Dr. Seuss et ...	Listening Lib...	Chick lit	1.6.17
Daisy-Head ...	Dr. Seuss	Random Hou...	Children's bo...	4.1.93

Section 3

Java Files:

Main.java

```
//Driver Class
public class Main {
    public static void main(String[] args) {new Login();}
}
```

Login.java

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SpringLayout;

//Establishing JFrame
public class Login extends JFrame {
    public Login() {
        //Title of the frame
        setTitle("Login");
        //Panel for contents in the frame
        JPanel panel = new JPanel();
        //Setting the layout of the panel
        SpringLayout layout = new SpringLayout();

        //Setting layout of the label
        JLabel l1=new JLabel("User Name: ");
        layout.putConstraint(SpringLayout.WEST, l1, 5, SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, l1, 5, SpringLayout.NORTH, panel);
        //Setting layout of textField
        JTextField q= new JTextField(15);
        layout.putConstraint(SpringLayout.WEST, q, 5, SpringLayout.EAST, l1);
        layout.putConstraint(SpringLayout.NORTH, q, 5, SpringLayout.NORTH, l1);
```

```java
        //Setting layout of the label
        JLabel l2=new JLabel("Password: ");
        layout.putConstraint(SpringLayout.WEST, l2, 5, SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, l2, 40, SpringLayout.NORTH, panel);
        //Setting layout of textField
        JTextField w= new JTextField(15);
        layout.putConstraint(SpringLayout.WEST, w, 10, SpringLayout.EAST, l2);
        layout.putConstraint(SpringLayout.NORTH, w, 5, SpringLayout.NORTH, l2);

        //Button declaration
        JButton a=new JButton("Confirm");
        a.addActionListener(e -> {
            //Getting the input of credentials
            String UserName = q.getText();
            String Password = w.getText();

            //Determining if the user is authorized
            if (UserName.equals("admin") && Password.equals("1234")){
                dispose();
                Menu page0 = new Menu();
                page0.setVisible(true);
            }else{
                //Fail case
                JOptionPane.showMessageDialog(null,"Invalid Credentials!");
            }
        });
        //Setting layout of button
        layout.putConstraint(SpringLayout.WEST, a, 5, SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, a, 90, SpringLayout.NORTH, panel);

        //Adding the panel to the frame
        add(panel);
        //Adding the layout to the panel
        panel.setLayout(layout);
        //Adding the contents to the panel
        panel.add(l1);
        panel.add(q);
        panel.add(l2);
        panel.add(w);
        panel.add(a);
        //Setting frame size
        setSize(400,180);
        //Setting frame location
        setLocationRelativeTo(null);
        //Setting visibility of the frame
        setVisible(true);
        //Setting closing case
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

Menu.java

```java
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.WindowConstants;
import java.awt.FlowLayout;

//Establishing JFrame
public class Menu extends JFrame {
   public Menu() {
       //Title of the frame
       setTitle("Menu");
       //Panel for contents in the frame
       JPanel panel = new JPanel();
       //Button declaration
       JButton a = new JButton("Books");
       //Go to Books frame
       a.addActionListener(e -> {
           //make the current page invisible
           dispose();

           //create instance of the NewPage
           Books page1 = new Books();

           //make page visible to the user
           page1.setVisible(true);
       });

       //Button declaration
       JButton b = new JButton("Search");
       //Go to Search frame
       b.addActionListener(e -> {
           //make the current page invisible
           dispose();

           //create instance of the NewPage
           Search page2 = new Search();

           //make page visible to the user
           page2.setVisible(true);
       });

       //Button declaration
       JButton c = new JButton("Create");
       //Go to Create frame
       c.addActionListener(e -> {
           //make the current page invisible
           dispose();

           //create instance of the NewPage
           Create page3 = new Create();
```

```java
            //make page visible to the user
            page3.setVisible(true);
        });

        //Button declaration
        JButton d = new JButton("Delete");
        //Go to Delete frame
        d.addActionListener(e -> {
            //make the current page invisible
            dispose();

            //create instance of the NewPage
            Delete page4 = new Delete();

            //make page visible to the user
            page4.setVisible(true);
        });

        //Button declaration
        JButton f = new JButton("Edit");
        //Go to Edit frame
        f.addActionListener(e -> {
            //make the current page invisible
            dispose();

            //create instance of the NewPage
            Edit page5 = new Edit();

            //make page visible to the user
            page5.setVisible(true);
        });

        //Adding the panel to the frame
        add(panel);
        //Setting the layout of the panel
        panel.setLayout(new FlowLayout(FlowLayout.CENTER));
        //Adding the contents to the panel
        panel.add(a);
        panel.add(b);
        panel.add(c);
        panel.add(d);
        panel.add(f);
        //Setting frame size
        setSize(400, 180);
        //Setting frame location
        setLocationRelativeTo(null);
        //Setting visibility of the frame
        setVisible(true);
        //Setting closing case
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }
}
```

Books.java

```java
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.logging.Level;
import java.util.logging.Logger;

//Establishing JFrame
public class Books extends JFrame {
   public Books(){
       //Title of the frame
       setTitle("Books");
       //Panel for contents in the frame
       JPanel panel1 = new JPanel();
       //Panel for contents in the frame
       JPanel panel2 = new JPanel();
       //Setting the layout of the panel
       BoxLayout layout1 = new BoxLayout(panel1, BoxLayout.Y_AXIS);

       //Setting table to display the database
       JTable x = new JTable();
       //Adding scrollPane to the table
       JScrollPane sp = new JScrollPane(x);

       //Button declaration
       JButton a=new JButton("Display");
       //Displaying database in the table
       a.addActionListener(e -> {
           //Instantiating the file
           File file = new File("data.txt");

           try {
               BufferedReader br = new BufferedReader(new FileReader(file));
               //Get the first line
               //Get the column name from the first line
               //Set column names to the JTable model
               String firstLine = br.readLine().trim();
               String[] columnsName = firstLine.split(",");
               DefaultTableModel model = (DefaultTableModel)x.getModel();
               model.setColumnIdentifiers(columnsName);

               //Get the line from data.txt file
```

```
                Object[] tableLines = br.lines().toArray();

                //Extract data from lines
                //Set data to JTable model
                for (Object tableLine : tableLines) {
                    String line = tableLine.toString().trim();
                    String[] dataRow = line.split("/");
                    model.addRow(dataRow);
                }

            } catch(Exception ex) {
                //Fail case
                Logger.getLogger(Books.class.getName()).log(Level.SEVERE,null,ex);
            }
        });

        //Button declaration
        JButton b=new JButton("Clear");
        //Clearing the table
        b.addActionListener(e -> x.setModel(new DefaultTableModel(null, new String[]{
"Title", "Author(s)", "Publisher", "Subject", "Publishing Date" })));

        //Button declaration
        JButton c=new JButton("Return");
        //Return to Menu
        c.addActionListener(e -> {
            //make the current page invisible
            dispose();

            //create instance of the Menu
            Menu page0 = new Menu();

            //make page visible to the user
            page0.setVisible(true);
        });

        //Adding the panel to the frame
        add(panel1);
        //Adding the layout to the panel
        panel1.setLayout(layout1);
        //Adding the panel to the frame
        add(panel2);
        //Adding the contents to the panel
        panel1.add(a);
        panel1.add(Box.createRigidArea(new Dimension(0, 20)));
        panel1.add(b);
        panel1.add(Box.createRigidArea(new Dimension(0, 80)));
        panel1.add(c);
        panel2.add(sp);
        //Setting the layout of the frame
        setLayout(new FlowLayout());
        //Setting frame size
        setSize(800,500);
        //Setting frame location
        setLocationRelativeTo(null);
        //Setting closing case
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

Search.java

```
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.logging.Level;
import java.util.logging.Logger;

//Establishing JFrame
public class Search extends JFrame {
   public Search(){
        //Title of the frame
        setTitle("Search");
        //Panel for contents in the frame
        JPanel panel1 = new JPanel();
        //Panel for contents in the frame
        JPanel panel2 = new JPanel();
        //Setting the layout of the panel
        BoxLayout layout1 = new BoxLayout(panel1, BoxLayout.Y_AXIS);

        //Setting table to display the database
        JTable x = new JTable();
        //Adding scrollPane to the table
        JScrollPane sp = new JScrollPane(x);
        //Drop-down menu for sorting algorithm options
        String[] choices = { "Default", "Bubble Sort","Merge Sort", "Radix Sort" };

        //Adding actions to the drop-down menu
        final JComboBox<String> c = new JComboBox<>(choices);
        c.addActionListener(e -> {
            JComboBox<String> comboBox = (JComboBox<String>)e.getSource();
            Object o = comboBox.getSelectedItem();
            //Instantiating the file
            File file = new File("data.txt");
            //Instantiating the sorted file
            File fileSorted = new File("sorted.txt");

            try {
                //Unsorted database
                if(o != null && o.equals("Default")) {
                    BufferedReader br = new BufferedReader(new FileReader(file));
                    //Get the first line
                    //Get the column name from the first line
                    //Set column names to the JTable model
                    String firstLine = br.readLine().trim();
                    String[] columnsName = firstLine.split(",");
                    DefaultTableModel model = (DefaultTableModel) x.getModel();
                    model.setColumnIdentifiers(columnsName);
                    //Get the line from data.txt file
                    Object[] tableLines = br.lines().toArray();

                    //Extract data from lines
                    //Set data to JTable model
                    for (Object tableLine : tableLines) {
                        String line = tableLine.toString().trim();
                        String[] dataRow = line.split("/");
                        model.addRow(dataRow);
                    }
                //Bubble sorted database
                } else if (o != null && o.equals("Bubble Sort")) {
                    Integration bubble = new Integration();
                    bubble.bubbleAlgo();
                    BufferedReader br = new BufferedReader(new FileReader(fileSorted));
                    //Get the first line
                    //Get the column name from the first line
                    //Set column names to the JTable model
                    String firstLine = br.readLine().trim();
                    String[] columnsName = firstLine.split(",");
                    DefaultTableModel model = (DefaultTableModel) x.getModel();
                    model.setColumnIdentifiers(columnsName);
                    //Get the line from sorted.txt file
                    Object[] tableLines = br.lines().toArray();

                    //Extract data from lines
                    //Set data to JTable model
                    for (Object tableLine : tableLines) {
                        String line = tableLine.toString().trim();
                        String[] dataRow = line.split("/");
                        model.addRow(dataRow);
                    }
                //Merge sorted database
                } else if (o != null && o.equals("Merge Sort")) {
                    Integration merge = new Integration();
                    merge.mergeAlgo();
                    BufferedReader br = new BufferedReader(new FileReader(fileSorted));
                    //Get the first line
                    //Get the column name from the first line
                    //Set column names to the JTable model
                    String firstLine = br.readLine().trim();
                    String[] columnsName = firstLine.split(",");
                    DefaultTableModel model = (DefaultTableModel) x.getModel();
                    model.setColumnIdentifiers(columnsName);
                    //Get the line from sorted.txt file
                    Object[] tableLines = br.lines().toArray();

                    //Extract data from lines
                    //Set data to JTable model
                    for (Object tableLine : tableLines) {
                        String line = tableLine.toString().trim();
                        String[] dataRow = line.split("/");
                        model.addRow(dataRow);
                    }
                } else if (o != null && o.equals("Radix Sort")) {
                    RadixSortAlgorithm.Radix();
                    BufferedReader br = new BufferedReader(new FileReader(fileSorted));
                    //Get the first line
                    //Get the column name from the first line
                    //Set column names to the JTable model
                    String firstLine = br.readLine().trim();
                    String[] columnsName = firstLine.split(",");
                    DefaultTableModel model = (DefaultTableModel) x.getModel();
                    model.setColumnIdentifiers(columnsName);
                    //Get the line from sorted.txt file
                    Object[] tableLines = br.lines().toArray();

                    //Extract data from lines
                    //Set data to JTable model
                    for (Object tableLine : tableLines) {
                        String line = tableLine.toString().trim();
                        String[] dataRow = line.split("/");
                        model.addRow(dataRow);
                    }
                }
            } catch(Exception ex) {
                //Fail case
                Logger.getLogger(Search.class.getName()).log(Level.SEVERE,null,ex);
            }
        });

        //Button declaration
        JButton a=new JButton("Display");
        //Displaying database in the table
        a.addActionListener(e -> {
            //Instantiating the file
            File file = new File("data.txt");
            //Displaying the unsorted database
            try {
                BufferedReader br = new BufferedReader(new FileReader(file));
                //Get the first line
                //Get the column name from the first line
                //Set column names to the JTable model
                String firstLine = br.readLine().trim();
                String[] columnsName = firstLine.split(",");
                DefaultTableModel model = (DefaultTableModel)x.getModel();
                model.setColumnIdentifiers(columnsName);

                //Get the line from data.txt file
                Object[] tableLines = br.lines().toArray();

                //Extract data from lines
                //Set data to JTable model
                for (Object tableLine : tableLines) {
                    String line = tableLine.toString().trim();
                    String[] dataRow = line.split("/");
                    model.addRow(dataRow);
                }

            } catch(Exception ex) {
                //Fail case
                Logger.getLogger(Search.class.getName()).log(Level.SEVERE,null,ex);
            }
        });


        //Button declaration
        JButton b=new JButton("Clear");
        //Clearing the table
        b.addActionListener(e -> x.setModel(new DefaultTableModel(null, new
String[]{"Title", "Author(s)", "Publisher", "Subject", "Publishing Date"})));

        //Button declaration
        JButton d=new JButton("Return");
        //Return to Menu
        d.addActionListener(e -> {
            //make the current page invisible
            dispose();

            //create instance of the Menu
            Menu page0 = new Menu();

            //make page visible to the user
            page0.setVisible(true);
        });
        //Adding the panel to the frame
        add(panel1);
        //Adding the layout to the panel
        panel1.setLayout(layout1);
        //Adding the panel to the frame
        add(panel2);
        //Adding the contents to the panel
        panel1.add(c);
        panel1.add(Box.createRigidArea(new Dimension(0, 20)));
        panel1.add(a);
        panel1.add(Box.createRigidArea(new Dimension(0, 20)));
        panel1.add(b);
        panel1.add(Box.createRigidArea(new Dimension(0, 80)));
        panel1.add(d);
        panel2.add(sp);
        //Setting the layout of the frame
        setLayout(new FlowLayout());
        //Setting frame size
        setSize(800,500);
        //Setting frame location
        setLocationRelativeTo(null);
        //Setting closing case
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

Create.java

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SpringLayout;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
//Establishing JFrame
public class Create extends JFrame {
   public Create(){
       //Title of the frame
       setTitle("Create");
       //Panel for contents in the frame
       JPanel panel = new JPanel();
       //Setting the layout of the panel
       SpringLayout layout = new SpringLayout();

       //Setting layout of the label
       JLabel l1=new JLabel("Title: ");
       layout.putConstraint(SpringLayout.WEST, l1, 5, SpringLayout.WEST, panel);
       layout.putConstraint(SpringLayout.NORTH, l1, 5, SpringLayout.NORTH, panel);
       //Setting layout of textField
       JTextField q= new JTextField(15);
       layout.putConstraint(SpringLayout.WEST, q, 34, SpringLayout.EAST, l1);
       layout.putConstraint(SpringLayout.NORTH, q, 5, SpringLayout.NORTH, l1);

       //Setting layout of the label
       JLabel l2=new JLabel("Author(s): ");
       layout.putConstraint(SpringLayout.WEST, l2, 5, SpringLayout.WEST, panel);
       layout.putConstraint(SpringLayout.NORTH, l2, 35, SpringLayout.NORTH, panel);
       //Setting layout of textField
       JTextField w= new JTextField(15);
       layout.putConstraint(SpringLayout.WEST, w, 5, SpringLayout.EAST, l2);
       layout.putConstraint(SpringLayout.NORTH, w, 5, SpringLayout.NORTH, l2);

       //Setting layout of the label
       JLabel l3=new JLabel("Publisher: ");
       layout.putConstraint(SpringLayout.WEST, l3, 5, SpringLayout.WEST, panel);
       layout.putConstraint(SpringLayout.NORTH, l3, 65, SpringLayout.NORTH, panel);
       //Setting layout of textField
       JTextField x= new JTextField(15);
       layout.putConstraint(SpringLayout.WEST, x, 5, SpringLayout.EAST, l3);
       layout.putConstraint(SpringLayout.NORTH, x, 5, SpringLayout.NORTH, l3);

       //Setting layout of the label
       JLabel l4=new JLabel("Subject: ");
       layout.putConstraint(SpringLayout.WEST, l4, 5, SpringLayout.WEST, panel);
       layout.putConstraint(SpringLayout.NORTH, l4, 95, SpringLayout.NORTH, panel);
       //Setting layout of textField
       JTextField y= new JTextField(15);
       layout.putConstraint(SpringLayout.WEST, y, 54, SpringLayout.EAST, l4);
       layout.putConstraint(SpringLayout.NORTH, y, 5, SpringLayout.NORTH, l4);

       //Setting layout of the label
       JLabel l5=new JLabel("Publication Date: ");
       layout.putConstraint(SpringLayout.WEST, l5, 5, SpringLayout.WEST, panel);
       layout.putConstraint(SpringLayout.NORTH, l5, 125, SpringLayout.NORTH, panel);
       //Setting layout of textField
       JTextField z= new JTextField(15);
       layout.putConstraint(SpringLayout.WEST, z, 5, SpringLayout.EAST, l5);
```

```
        layout.putConstraint(SpringLayout.NORTH, z, 5, SpringLayout.NORTH, 15);

        //Button declaration
        JButton a=new JButton("Submit");
        a.addActionListener(e -> {
            //Instantiating the file
            Path p = Paths.get("data.txt");
            //Getting input from the user
            String s = System.lineSeparator() + q.getText() + " / " + w.getText() + " / "
+ x.getText() + " / " + y.getText() + " / " + z.getText();

            try {
                //Writing the input to the file
                Files.write(p, s.getBytes(), StandardOpenOption.APPEND);
                //Clear the user input in the fields
                q.setText("");
                w.setText("");
                x.setText("");
                y.setText("");
                z.setText("");

            } catch (Exception ex) {
                //Fail case
                Logger.getLogger(Create.class.getName()).log(Level.SEVERE, null, ex);
            }
            //Confirmation message
            JOptionPane.showMessageDialog(null, "Book Created!");
        });
        //Setting layout of button
        layout.putConstraint(SpringLayout.WEST, a, 5, SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, a, 165, SpringLayout.NORTH, panel);

        //Button declaration
        JButton b=new JButton("Return");
        //Return to Menu
        b.addActionListener(e -> {
            //make the current page invisible
            dispose();

            //create instance of the Menu
            Menu page0 = new Menu();

            //make page visible to the user
            page0.setVisible(true);
        });
        //Setting layout of button
        layout.putConstraint(SpringLayout.WEST, b, 30, SpringLayout.EAST, a);
        layout.putConstraint(SpringLayout.NORTH, b, 0, SpringLayout.NORTH, a);

        //Adding the panel to the frame
        add(panel);
        //Adding the layout to the panel
        panel.setLayout(layout);
        //Adding the contents to the panel
        panel.add(l1);
        panel.add(q);
        panel.add(l2);
        panel.add(w);
        panel.add(l3);
        panel.add(x);
        panel.add(l4);
        panel.add(y);
        panel.add(l5);
        panel.add(z);
        panel.add(a);
        panel.add(b);
        //Setting frame size
        setSize(400,250);
        //Setting frame location
        setLocationRelativeTo(null);
        //Setting closing case
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

Delete.java

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SpringLayout;
import java.io.File;
import java.nio.file.Files;
import java.nio.file.StandardOpenOption;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.stream.Collectors;

//Establishing JFrame
public class Delete extends JFrame {
    public Delete(){
```

```java
        //Title of the frame
        setTitle("Delete");
        //Panel for contents in the frame
        JPanel panel = new JPanel();
        //Setting the layout of the panel
        SpringLayout layout = new SpringLayout();

        //Setting layout of the label
        JLabel l1=new JLabel("Enter a book name to delete: ");
        layout.putConstraint(SpringLayout.WEST, l1, 5, SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, l1, 5, SpringLayout.NORTH, panel);
        //Setting layout of textField
        JTextField q= new JTextField(15);
        layout.putConstraint(SpringLayout.WEST, q, 15, SpringLayout.EAST, l1);
        layout.putConstraint(SpringLayout.NORTH, q, 5, SpringLayout.NORTH, l1);

        //Setting layout of the label
        JLabel l2=new JLabel("<html><b><i>This will delete all information related to the
book!!!</i></b>");
        layout.putConstraint(SpringLayout.WEST, l2, 5, SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, l2, 35, SpringLayout.NORTH, panel);

        //Button declaration
        JButton a=new JButton("Delete");
        a.addActionListener(e -> {
            //Instantiating the file
            File file = new File("data.txt");

            try {
                //Getting the lines from the file
                List<String> out = Files.lines(file.toPath())
                        //Filtering the lines for the book name
                        .filter(line -> !line.contains(q.getText()))
                        .collect(Collectors.toList());
                Files.write(file.toPath(), out, StandardOpenOption.WRITE,
StandardOpenOption.TRUNCATE_EXISTING);
                //Clear the user input in the fields
                q.setText("");

            } catch (Exception ex) {
                //Fail case
                Logger.getLogger(Delete.class.getName()).log(Level.SEVERE, null, ex);
            }
            //Confirmation message
            JOptionPane.showMessageDialog(null, "Book Deleted!");
        });
        //Setting layout of button
        layout.putConstraint(SpringLayout.WEST, a, 5, SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, a, 100, SpringLayout.NORTH, panel);

        //Button declaration
        JButton b=new JButton("Return");
        //Return to Menu
        b.addActionListener(e -> {
            //make the current page invisible
            dispose();

            //create instance of the Menu
            Menu page0 = new Menu();

            //make page visible to the user
            page0.setVisible(true);
        });
        //Setting layout of button
        layout.putConstraint(SpringLayout.WEST, b, 30, SpringLayout.EAST, a);
        layout.putConstraint(SpringLayout.NORTH, b, 0, SpringLayout.NORTH, a);

        //Adding the panel to the frame
        add(panel);
        //Adding the layout to the panel
        panel.setLayout(layout);
        //Adding the contents to the panel
        panel.add(l1);
        panel.add(q);
        panel.add(l2);
        panel.add(a);
        panel.add(b);
        //Setting frame size
        setSize(400,180);
        //Setting frame location
        setLocationRelativeTo(null);
        //Setting closing case
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

Edit.java

```java
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SpringLayout;
import java.io.File;
import java.io.FileWriter;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

//Establishing JFrame
public class Edit extends JFrame {
    public Edit(){
        //Title of the frame
        setTitle("Edit");
        //Panel for contents in the frame
        JPanel panel = new JPanel();
        //Setting the layout of the panel
        SpringLayout layout = new SpringLayout();

        //Setting layout of the label
        JLabel l1=new JLabel("Title: ");
        layout.putConstraint(SpringLayout.WEST, l1, 5, SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, l1, 5, SpringLayout.NORTH, panel);
        //Setting layout of textField
        JTextField q= new JTextField(15);
        layout.putConstraint(SpringLayout.WEST, q, 34, SpringLayout.EAST, l1);
        layout.putConstraint(SpringLayout.NORTH, q, 5, SpringLayout.NORTH, l1);

        //Setting layout of the label
        JLabel l2=new JLabel("Author(s): ");
        layout.putConstraint(SpringLayout.WEST, l2, 5, SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, l2, 35, SpringLayout.NORTH, panel);
        //Setting layout of textField
        JTextField w= new JTextField(15);
        layout.putConstraint(SpringLayout.WEST, w, 5, SpringLayout.EAST, l2);
        layout.putConstraint(SpringLayout.NORTH, w, 5, SpringLayout.NORTH, l2);

        //Setting layout of the label
        JLabel l3=new JLabel("Publisher: ");
        layout.putConstraint(SpringLayout.WEST, l3, 5, SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, l3, 65, SpringLayout.NORTH, panel);
        //Setting layout of textField
        JTextField x= new JTextField(15);
        layout.putConstraint(SpringLayout.WEST, x, 5, SpringLayout.EAST, l3);
        layout.putConstraint(SpringLayout.NORTH, x, 5, SpringLayout.NORTH, l3);

        //Setting layout of the label
        JLabel l4=new JLabel("Subject: ");
        layout.putConstraint(SpringLayout.WEST, l4, 5, SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, l4, 95, SpringLayout.NORTH, panel);
        //Setting layout of textField
        JTextField y= new JTextField(15);
        layout.putConstraint(SpringLayout.WEST, y, 53, SpringLayout.EAST, l4);
        layout.putConstraint(SpringLayout.NORTH, y, 5, SpringLayout.NORTH, l4);

        //Setting layout of the label
        JLabel l5=new JLabel("Publication date: ");
        layout.putConstraint(SpringLayout.WEST, l5, 5, SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, l5, 125, SpringLayout.NORTH, panel);
        //Setting layout of textField
        JTextField z= new JTextField(15);
        layout.putConstraint(SpringLayout.WEST, z, 5, SpringLayout.EAST, l5);
        layout.putConstraint(SpringLayout.NORTH, z, 5, SpringLayout.NORTH, l5);

        //Button declaration
        JButton a=new JButton("Confirm");
        a.addActionListener(e -> {
            //Instantiating the file
            String filePath = "data.txt";
            try {
                //Instantiating the Scanner class to read the file
                Scanner sc = new Scanner(new File(filePath));
                //instantiating the StringBuffer class
                StringBuilder buffer = new StringBuilder();
                //Reading lines of the file and appending them to StringBuffer
                    while (sc.hasNextLine()) {
                        buffer.append(sc.nextLine()).append(System.lineSeparator());
                    }
                String fileContents = buffer.toString();
                //closing the Scanner object
                sc.close();
                String oldLine = q.getText();
                String newLine = q.getText() + " / " + w.getText() + " / " + x.getText() +
" / " + y.getText() + " / " + z.getText();
                //Replacing the old line with new line
                fileContents = fileContents.replaceAll(oldLine, newLine);
                //instantiating the FileWriter class
                FileWriter writer = new FileWriter(filePath);
                writer.append(fileContents);
                writer.flush();
                //Clear the user input in the fields
                q.setText("");
                w.setText("");
                x.setText("");
                y.setText("");
                z.setText("");

            } catch (Exception ex) {
                //Fail case
                Logger.getLogger(Edit.class.getName()).log(Level.SEVERE, null, ex);
            }
            //Confirmation message
            JOptionPane.showMessageDialog(null, "Changes Saved!");
        });
        layout.putConstraint(SpringLayout.WEST, a, 5, SpringLayout.WEST, panel);
        layout.putConstraint(SpringLayout.NORTH, a, 165, SpringLayout.NORTH, panel);

        //Button declaration
        JButton b=new JButton("Return");
        //Return to Menu
        b.addActionListener(e -> {
            //make the current page invisible
            dispose();

            //create instance of the Menu
            Menu page0 = new Menu();

            //make page visible to the user
            page0.setVisible(true);
        });
        //Setting layout of button
        layout.putConstraint(SpringLayout.WEST, b, 30, SpringLayout.EAST, a);
        layout.putConstraint(SpringLayout.NORTH, b, 0, SpringLayout.NORTH, a);

        //Adding the panel to the frame
        add(panel);
        //Adding the layout to the panel
        panel.setLayout(layout);
        //Adding the contents to the panel
        panel.add(l1);
        panel.add(q);
        panel.add(l2);
        panel.add(w);
        panel.add(l3);
        panel.add(x);
        panel.add(l4);
        panel.add(y);
        panel.add(l5);
        panel.add(z);
        panel.add(a);
        panel.add(b);
        //Setting frame size
        setSize(400,250);
        //Setting frame location
        setLocationRelativeTo(null);
        //Setting closing case
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

Kotlin Files:

BubbleSortAlgorithm.kt

```
import kotlin.collections.ArrayList

//creating class of bubble sorting
class BubbleSortAlgorithm {
   //creating function of bubble sorting

   fun bubbleSorting(lines: ArrayList<String>){

       var tem:String

       //first pass by iterating through all the index number of array
       for(j in lines.indices){

           //second pass by iterating  next current index number until the size of array
           for (i in j+1 until lines.size){

               //comparing the current value with the next  value.
               //The algorithm will be looping as long as   (next value < current value)
is true.
               //it compares two  elements' uppercase value
               if (lines[i].uppercase().filter { !it.isWhitespace()}<
lines[j].uppercase().filter { !it.isWhitespace()}){

                   //temporarily storing the current value
                   tem = lines[j]

                   //replace current value to the next current value
                   lines[j] = lines[i]

                   //next value will be stored temporarily to tem variable
                   lines[i] = tem

                   //loop again  to the  next comparison for 2nd pass
               }

               //if 2nd pass is completed, go for the next index number of first pass.
           }

           //loop again  to the 2nd pass then 3rd and so forth to the last indices of
ArrayList
       }

   }
}
```

MergeSortAlgorithm.kt

```
class MergeSortAlgorithm {
    /**
     * An implementation of merge sort procedure
     * AveragePerformance = O(n*lg(n)), where lg(n) is a logarithm of n for base 2
     * The function mergeSort() gets a list and splits at the middle into two new lists.
     * This lists will be passed to mergeSort again until there only lists with one item.
     */

    fun mergeSort(list: MutableList<String>): List<String> {

        // making sure the list is not empty
        //If a given list is empty the function will return empty list and will be
executed.
        if (list.size <= 1) {
            return list
        }
        //finding the middle index number of the list
        val middle = list.size / 2

        //Returns a view of the portion of this list between the specified fromIndex
(inclusive) and toIndex (exclusive)
        // The returned list is backed by this list
        val left = list.subList(0, middle)
        val right = list.subList(middle, list.size)

        // this is a recursive call.
        // firstly  mergeSort function call "merge" function then it will call itself
        // the algorithm will split and sort both halves before merging them.
        //finally it is going to return the sorted list

        return merge(mergeSort(left), mergeSort(right))
    }

    /**
     * Merges two sublist of initial mutableList of list
     * in ascending order.
     * */

    private fun merge(left: List<String>, right: List<String>): List<String> {
        var indexLeft = 0
        var indexRight = 0
        val newList: MutableList<String> = mutableListOf()

        // mergeSort algorithm is “divide and conquer”
        // it is dividing the element until there is only one element or none left
        while (indexLeft < left.count() && indexRight < right.count()) {

            if (left[indexLeft].uppercase().filter { !it.isWhitespace()} <=
right[indexRight].uppercase().filter { !it.isWhitespace()}) {
                newList.add(left[indexLeft])
                indexLeft++
            } else {
                newList.add(right[indexRight])
                indexRight++
            }
        }

        while (indexLeft < left.size) {
            newList.add(left[indexLeft])
            indexLeft++
        }

        while (indexRight < right.size) {
            newList.add(right[indexRight])
            indexRight++
        }
        return newList

    }
}
```

InputOutput.kt

```
import java.io.*

class InputOutput {

    //Reading  and writing  text file using java.io package of java

    fun readFile():ArrayList<String>{
        val lines = ArrayList<String>() // Creating an empty array
        val reader: BufferedReader?
        reader = BufferedReader(FileReader("data.txt")) // Read a stream of characters
from the text file
        var currentLine = reader.readLine() // Read the characters line by line

        while (currentLine != null) {
            lines.add(currentLine)         // storing  each line as an element in an array
            currentLine = reader.readLine()

        }
        reader.close()                // close the file output stream and releases all
system resources
```

```
        lines.removeAt(0)     // remove title of the array
        return lines
    }

    fun writeFile(args: List<String>){
        val writer: BufferedWriter?
        writer = BufferedWriter(FileWriter("Sorted.txt")) // Write a stream of characters from the sorted array

        writer.write("Title, Author(s), Publisher, Subject, Publishing Date\n")
        for (line in args) {
            writer.write(line) // Write each element of sorted array to a new text file
            writer.newLine()  // jump to a new  line after writing each element

        }
        try {

            writer.close()  // close output stream
        } catch (e: IOException) {
            e.printStackTrace()
        }
    }
}
```

Integration.kt

```
import kotlin.time.ExperimentalTime
import kotlin.time.measureTimedValue

class Integration {

    // Sorting a data by calling bubbleAlgo or mergeAlgo function

    @OptIn(ExperimentalTime::class)
    fun bubbleAlgo() {
        val a = InputOutput()        // Creating an object of InputOutput class
        val array = a.readFile()     // Read the Input file and store in an array by calling readFile function
        val x = BubbleSortAlgorithm() // Creating an object of BubbleSortAlgorithm class

        val time = measureTimedValue {   //measuring time
            x.bubbleSorting(array)       // sorting the array
        }
        a.writeFile(array)// storing the sorted array to a text file

        println("The execution time for Bubble Sorting is:
${time.duration.inWholeMilliseconds} milliseconds")
    }

    @OptIn(ExperimentalTime::class)
    fun mergeAlgo (){

        val a = InputOutput()            // Creating an object of InputOutput class
        val getArray = a.readFile() // Read the Input file and store in an array by
calling readFile function

        var sortedArray: ArrayList<String>
        val merge = MergeSortAlgorithm()// Creating an object of mergeSortAlgorithm  class
        val time = measureTimedValue {
            sortedArray = merge.mergeSort(getArray) as ArrayList<String> //sorting the
array using mergeSort function
        }
        a.writeFile(sortedArray)// storing the sorted array to txt file

        println("The execution time for Merge Sorting is:
${time.duration.inWholeMilliseconds} milliseconds")

    }
}
```

Scala Files:

BookExtended.scala

```
case class BookExtended(title:String, author:String, publisher:String, subject:String,
publishingDate:String)
```

RadixSortAlgorithm.scala

```
import java.io._
import scala.io._
import scala.util.{Failure, Success, Try}

object RadixSortAlgorithm {

 def Radix(): Unit = {
   // buffering the source file named "data.txt"
    val bookSource = Source.fromFile("data.txt") //U can change Books Source
   // List by parsing into rows
    val data = bookSource.getLines().toList
   // separating the list as "header" and "tail"
   val (header, tail) = (data.head, data.tail)
   //splitting tail with "/" then mapping all tail . and get List of BookExtended
    val book_List = tail.map(l => {
     val split = l.split("/").map(_.strip())
     BookExtended(split(0), split(1), split(2), split(3), split(4))
   })
   bookSource.close()

   //recording the startTime before using the radix method
    val startTime = System.currentTimeMillis()
   // sorts by titles first and then sorts by authors .
   //using nested Radix method named "sortRadixList".
    val bookSorted = sortRadixList[BookExtended](
     //inside method sorts by authors
     sortRadixList[BookExtended](book_List, _.author),
     //outside method sorts by titles
     _.title
   )
   //mapping List of BookExtended named "bookSorted" and converting to List of String as
sortBOOKasList
```

```
    val sortBOOKasList = bookSorted.map(_.productIterator.mkString("/"))
   //add the header to the list
    val outputTextFile = sortBOOKasList.+:(header)
   //recording the endTime after using the radix method
    val endTime = System.currentTimeMillis()
   //calculating the executionTime for the radix method
    val executionTime = endTime - startTime
   //Print here: time elapsed sort book
   println(s"The execution time for Radix Sorting is : $executionTime milliseconds\n")

   //using printToFile method to output file named "sorted.txt".
   printToFile(new File("sorted.txt")) { p =>
     outputTextFile.map(l => p.println(l))
   }
 }
 //a list that has elements of type T
 //  Radix method named "sortRadixList" that works any types "A"
 //  takes List input parameter that has elements of type any: List[A]  named "listRaw"
 //  and takes "A" parameter  of function returns String  named "columnFunc"
 def sortRadixList[A](listRaw: List[A], columnFunc: A => String): List[A] = {
   // find the longest String of the given column
   val wMax = listRaw.map(l => columnFunc(l).length).max

   // A method named "accLSD" is inner method of : "sortRadixList" method. it is
tailRecursive method
   @scala.annotation.tailrec
   def accLSD(listAdj: List[A], chIndex: Int): List[A] = {
     // splitting into long and short Strings in 2 lists using map function
     val listParts = listAdj.partition(l => columnFunc(l).length < chIndex + 1)

     // matching String index
     chIndex match {
       case -1 => listAdj
       case _ => accLSD(listParts._1 ::: listParts._2.map(l => columnFunc(l)(chIndex)).
         // using distinct and sorted func of scala
         distinct.sorted.
         flatMap(ch => listParts._2.filter(fStr => columnFunc(fStr)(chIndex) == ch)),
chIndex - 1)
     }
   }
   //starting inner method here, which is tail recursive
   accLSD(listRaw, wMax - 1)
 }
 // printToFile method to write a file
 def printToFile(f: java.io.File)(op: java.io.PrintWriter => Unit): Unit = {
   // create a new writer
   val p = new java.io.PrintWriter(f)
   Try(op(p)) match {
     case Success(_) => p.close()
     case Failure(exception) => println(s"Got an error $exception")
   }
 }
}
```

Section 4

COURSEWORK CONTRIBUTION FORM

Team member name	Student ID	individual overall work contribution (%)	Note
Bahadir Erkam Bakoglu	001089837	45%	Tech lead - Frontend developer
Mohammad Nazmul Islam	001083736	40%	Backend developer
Mohammad Tamjid Bin Sarwar	001086421	15%	Backend developer
		Total 100%	

Individual Report

Section 1

The Scala programming language is a general-purpose language that supports both object-oriented and functional programming. Scala is therefore an extremely flexible language in terms of functional programming. Due to its compatibility to Java bytecode, Scala can be executed on any JVM, thereby maintaining interoperability with Java. The Scala programming language, like Kotlin, is statically typed, but it is much more complex and comprehensive in comparison (Odersky *et al*. 2004). This makes Scala programming more difficult to understand. Among the advantages of Scala are pattern matching statements which are ideal for processing substantial amounts of data. In addition to the high degree of flexibility with which it can be coded. As for the disadvantages, it is difficult to invoke Java classes from Scala, even though Scala is interoperable with Java the more complex classes may cause an error. Additionally, when it comes to the speed of compilation of Scala in comparison to Java and Kotlin, it is a critical issue when considering its use in a large-scale application. Moreover, Scala is an established programming language. Consequently, there will be a higher level of community support.

Kotlin, on the other hand, is an object-oriented programming language. Even though Kotlin is not a general-purpose language, its concise nature makes it more convenient for developers to learn the code structure and reduces the level of bugs in the written code compared to Scala. The Kotlin programming language is also interoperable with Java, allowing users to call code from Java into Kotlin, and vice versa (Samuel and Bocutiu, 2017). Its concise nature makes Kotlin easier to understand than Scala since it is not as complicated. One of the advantages of Kotlin is its compact programming. This allows programmers to write fewer lines, thereby reducing bugs in the code while also keeping it easier to read. As a further advantage, Kotlin has complete interoperability with Java, allowing programmers to write adaptable code since Java classes are callable from Kotlin and vice versa. There are shortcomings to Kotlin, including the fact that pattern matching is not fully supported. Additionally, Kotlin's applicability to existing development cycles is limited because of its relative newness when compared to Scala. Because Kotlin is a newly developed language, it does not have a strong user community that supports it, compared to Scala, which is a severe problem for developers.

Overall, Scala and Kotlin have different use cases. As a result of its better pattern matching capabilities and complex nature, Scala is an ideal language for large-scale operations such as big data-based solutions or machine learning algorithms. Due to its concise nature, easy understanding, and maintainability, Kotlin is a suitable choice for Android and web development.

Section 2

The development of the project started with the determination of the components of the application. The components were the front end of the application and the back end of the application. The front end of the application consists of the user interface and related features for manipulation of the database. The front end of the application was implemented in the Java programming language. The features of the front end include editing, deleting, and adding data and displaying existing data from the database in the user interface. The back end of the application includes sorting algorithms such as bubble, merge, and radix sort. Bubble and merge sort was implemented in Kotlin programming language and Radix sort was implemented in Scala programming language. The purpose of the sorting algorithms is to sort the database file into a sorted file to be displayed in the user interface. The sorting is conducted according to the user's choice of algorithm in the interface. The implementation required Kotlin and Scala classes to be called into Java for them to be assigned to elements in the user interface, such as buttons. This allowed the user to be able to choose between sorting algorithms.

The first problem we encountered during the development and integration process was about the difference in input and output format of the user interface and the algorithms. The output of the algorithms was displayed in the terminal instead of being written as a sorted file to be displayed in the interface. The issue has been quickly resolved by determining the source of the problem and establishing practical solutions to the issue. Another problem we had was with case sensitive sorting in the sorting algorithms. The issue has been solved in the Kotlin algorithms however, it persists in the Scala algorithm. Our final issue came up during integration when we were calling Kotlin and Scala classes and functions through the Java interface. To resolve the problem, we have divided the Kotlin and Scala classes into different components. Our solution allowed us to divide the functional parts and the parts that needed to be called into the Java interface and resolve the integration issue.

Being the most experienced member of the team in the Java programming language, I assumed the role of leader of the team. This involved helping the other members to understand their designated roles and providing advice when necessary. Among my leadership responsibilities, I oversaw developing the interface and functionalities related to the integration part of the project. Also, I managed designing the application and setting up the boundaries for the other team members. This included how each algorithm's output should be coordinated with everyone else's. Throughout the project, the team members were hardworking, open to advice, and dedicated to completing their assigned tasks. The development of the project mostly progressed smoothly considering the time it took for the team to set objectives and to work consistently towards achieving them. The thing that could be improved in the project is the addition of graphics and other visual elements. This will make the user interface look more appealing to users of the application. Furthermore, the user interface window does not currently adapt to custom window sizes. This means that if the user maximizes the window size or makes any changes, majority of the elements in the interface will not change their locations or sizes. It may be possible to overcome this problem by using another panel layout for displaying the elements within the frame. In addition, the database text file chosen for the project could be changed to a more sophisticated database file system such as a CSV file or a SQL file. This would enable the database file to have more flexibility and functionalities while viewing or editing the database file manually. Managing a team and maintaining coordination among team members was an excellent experience. Furthermore, I have gained a better understanding of the Kotlin and Scala languages from the amount of research and practice I have done in these languages. This has allowed me to provide advice and alternative solutions to the team.

References

Samuel, S. and Bocutiu, S., 2017. *Programming kotlin*. Packt Publishing Ltd.

Odersky, M., Altherr, P., Cremet, V., Emir, B., Maneth, S., Micheloud, S., Mihaylov, N., Schinz, M., Stenman, E. and Zenger, M., 2004. An overview of the Scala programming language.