Project Writeup

Virtual Keys Repository Application Prototype

End of Phase 1- OOPS Using Java Data Structures

Student: Nia Kelley Jester
niakelley@gmail.com

Full Stack Java Developer Master's Program

Summer 2022

Table of Contents

List of Tables, Figures & Pictures

Phase 1 Overview

The main objectives of Phase 1 of the Simplilearn Full Stack Java Development program were

- to gain an understanding of core concepts of the Java Programming Language (abstraction, polymorphism, inheritance, and encapsulation),
- embrace the Eclipse Integrated Development Environment (IDE),
- understand the Agile software development life cycle, and
- gain familiarity with Java data structures for object-oriented applications.

Phase 1 ended with a culminating project to demonstrate application of the concepts. The purpose of this paper is to document the project in detail.

Problem Statement

Lockers Pvt. Ltd. aims to digitize their product catalog. For the first phase of the project, they wish to develop a prototype of the application. The prototype of the application will be then presented to the relevant stakeholders for the budget approval, with the goal of delivering a high-end quality product as early as possible.

Lockers Pvt. Ltd. would like a presentation on the following topics in the next 15 working days (3 weeks):

- Specification document - Product's capabilities, appearance, and user interactions
- Number and duration of sprints required
- Setting up Git and GitHub account to store and track your enhancements of the prototype
- Java concepts being used in the project
- Data Structures where sorting and searching techniques are used
- Generic features and three operations:
    - Retrieving the file names in an ascending order
    - Business-level operations:
        - Option to add a user specified file to the application
        - Option to delete a user specified file from the application
        - Option to search a user specified file from the application
        - Navigation option to close the current execution context and return to the main context
- Option to close the application

Agile Project Management

This section will cover the project management details surrounding the software development life cycle for the virtual keys repository application prototype. **Error! Reference source not found.** provides an overview of the project and high-level software project management milestones.

Project Overview	
Client	Lockers Pvt. Ltd.
Consultant	Nia Kelley Jester Full Stack Java Developer
Application Name	LockedMe.com
Application Phase	Prototype
Phase 1 Project Deliverable	Console-based virtual keys repository Application intended for Budget Approval
Planning Project Management	
Project Duration	3 Weeks (15 working days)
Number of Sprints	3
Planned Sprint Duration	1 Week (5 working days)
Total Number of Product Backlog Items	14

Application User Roles

Error! Reference source not found. captures the roles for targeted software for the LockedMe.com virtual keys repository application prototype.

Role	Description	Software Version
General User	The General User will use the prototype application for file handling.	Initial Release
Admin	The Admin will use the protype application for maintaining users and setting user directory & file permissions.	Future Release

Project Planning Details

Error! Reference source not found. captures the project planning management overview for the virtual keys repository application prototype.

Sprint Number	Sprint Duration	Planned Start Date	Planned Finish Date	Product Backlog Items
1	1 Week	Monday, June 6, 2022	Friday, June 10, 2022	1.1 – Welcome Screen List (1.1.1, 1.1.2, 1.1.3)

				1.2 – Main Menu Options List (1.2.1, 1.2.2, 1.2.3)
				1.2.2 – Business Level Menu Options List
				1.2.2.4 – Business Level Option to go back to main menu
2	1 Week	6/13/2022	Friday, June 17, 2022	File Retrieval Options & Sorting Implementation (1.2.1.1, 1.2.1.2)
				File Addition Implementation (1.2.2.1)
3	1 Week	6/20/2022	Friday, June 24, 2022	File Deletion Implementation (1.2.2.2)
				File Search Implementation (1.2.2.3)

Product Backlog/User Stories

Error! Reference source not found. captures the User Stories created to implement the software feature.

Product Backlog ID	Role	Backlog Item (User Story)	Story Points	Assigned Sprint
1.1	General User	The LockedMe.com application will present the General User with a **Welcome screen** on the console.	Small	1
1.1.1	General User	The Welcome screen should clearly identify the application name on the console.	Small	1
1.1.2	General User	The Welcome screen should identify the developer's name and role on the console.	Small	1
1.1.3	General User	The application should provide the user with an option to proceed to the next menu.	Small	1
1.2	General User	The application should present a numerical menu of 3 user level interactions on the console. This will be the considered the **main menu** for the application.	Small	1
1.2.1	General User	The first main menu option should retrieve the current file names in an ascending order.	Small	1
1.2.1.1	General User	Ask the General User to specify the target directory. Once specified, the target directory cannot be changed.	Medium	2
1.2.1.2	General User	Sort the files in ascending order and display the resultant list on the console.	Medium	2

	1.2.2		General User	The second main menu option should provide **business level operations menu** with 4 options.	Small	1
		1.2.2.1	General User	The first business level operation is to a add new file in the target directory. The case sensitivity can be ignored for the file names.	Medium	2
		1.2.2.2	General User	The second business level operation is to delete a file from the target directory.	Medium	3
				• The delete functionality should incorporate case sensitivity on the file name to ensure that the right file is deleted from the directory listing.		
				• Once the file is found, ask the User for confirmation prior to file deletion.		
				• Provide an appropriate message once the file has been deleted.		
				• Return a message to the console if the file is not found.		
		1.2.2.3	General User	The third business level operation is to search for a user-specified file in the target directory.	Medium	3
				• The search functionality should incorporate case sensitivity on the file name to ensure that the right file is retrieved from the directory listing.		
				• Provide appropriate messages for successful operation.		
				• Provide appropriate message for unsuccessful operations.		
		1.2.2.4	General User	The fourth business level operation is to provide the option to go back to the main menu.	Small	1
	1.2.3		General User	The third main menu option should trigger an application close/exit operation.	Small	1

Implemented Java Concepts

This section will highlight the Java concepts used to create the virtual keys repository application prototype.

Packages

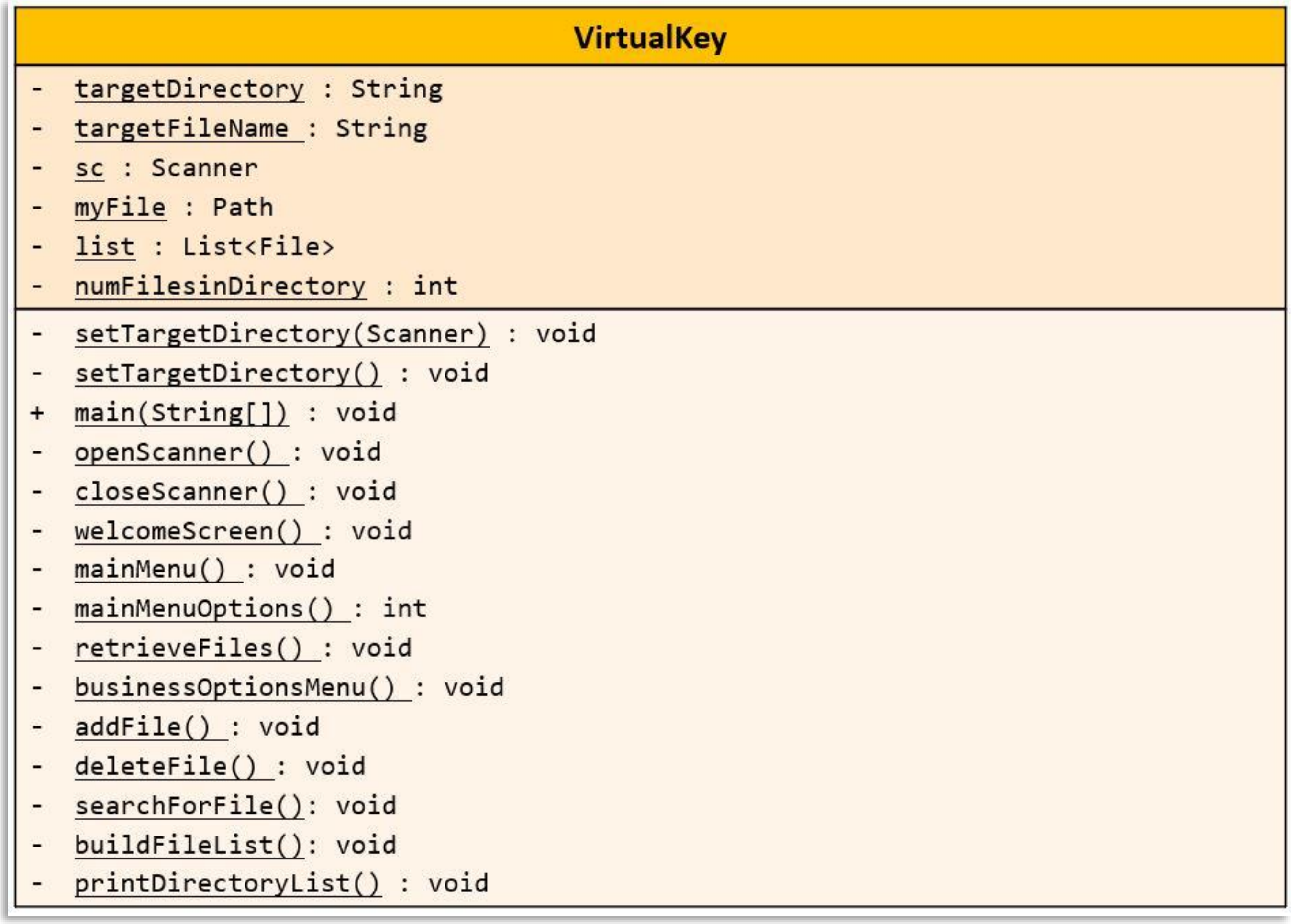
I chose to create a package dedicated to the practice problems for the Simplilearn program –

com.simplilearn.project.virtualkey

Classes, Objects, & Methods

Figure 1 captures the class diagram for the application. For this implementation, I chose to implement the program logic in the `main()` method.

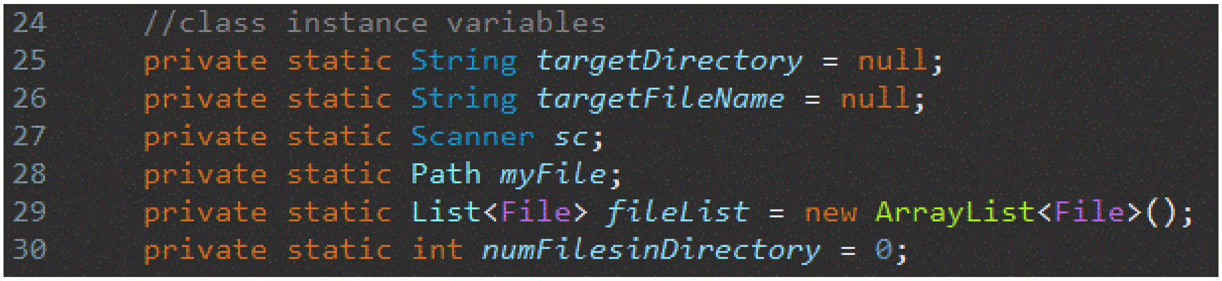
Figure 1 - UML Class Diagram for LockedMe.com Application



```
VirtualKey

- targetDirectory : String
- targetFileName : String
- sc : Scanner
- myFile : Path
- list : List<File>
- numFilesinDirectory : int

- setTargetDirectory(Scanner) : void
- setTargetDirectory() : void
+ main(String[]) : void
- openScanner() : void
- closeScanner() : void
- welcomeScreen() : void
- mainMenu() : void
- mainMenuOptions() : int
- retrieveFiles() : void
- businessOptionsMenu() : void
- addFile() : void
- deleteFile() : void
- searchForFile(): void
- buildFileList(): void
- printDirectoryList() : void
```

Instance Variables

I chose to declare 6 `private` instance variables for the LockedMe.com application prototype. I chose to initialization 4 of the static variables at the same time at declaration, via a single line statements.



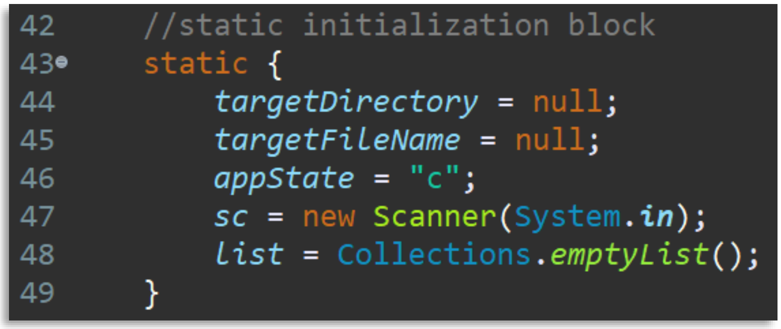
```java
    //class instance variables
    private static String targetDirectory = null;
    private static String targetFileName = null;
    private static Scanner sc;
    private static Path myFile;
    private static List<File> fileList = new ArrayList<File>();
    private static int numFilesinDirectory = 0;
```

For this version of the code, the instance variables are `private` and can only be accessed from the class. These variables are accessible to all the constructors and methods of the class. These instance variables are also `static`, which means they belong to the class. Objects created from the class cannot access them.

Since all the code for the Lockedme.com application was written in a single file, static variables can be called using the variable name only; there is no need to precede the static variable name with the class name and dot operator.

Static Initialization Block

For my first version of the code, I initialized the `static` instance variables using a `static` initialization block. This code will be executed only once when the class is loaded. However, I chose to remove the block to improve code readability. Please note that the following screenshot was only to demonstrate the static initialization block; the variable names and values were adjusted in the submitted version of the code.



```java
    //static initialization block
    static {
        targetDirectory = null;
        targetFileName = null;
        appState = "c";
        sc = new Scanner(System.in);
        list = Collections.emptyList();
    }
```

Static Class Methods

I made all my user-defined methods `static` and `private` for this version of the application. As such, these methods belong to the class. I chose not to instantiate any VirtualKey objects in my code, which means that I don't have any instance methods. I didn't need or want to access any object instance variables for this implementation. The `main()` method is the only publicly available method, which is of course `static`. Since all the code for the Lockedme.com application was written in a single file, static methods can be called using the method name only; there is no need to precede the static method name with the class name and dot operator.

Console Input and Output

Per the system requirements, the application is console based. Therefore, I used the `Scanner` class to

- Retrieve data from the console (using the `System.in` object to create a stream for the console input)
- Output messages to the console (using the `System.out` object and associated methods to output data to the console)

Given that there is so much interaction with the console, I chose to perform exception handling any time I requested console input from the user. This application really taught me when to apply exception handling in a practical sense. Previously, I understood the concept at a high level, but didn't know when to apply it. Also, I took advantage of the clues provided by the Eclipse IDE.

Control Statements

The program utilizes the following control statements to direct the desired logic:

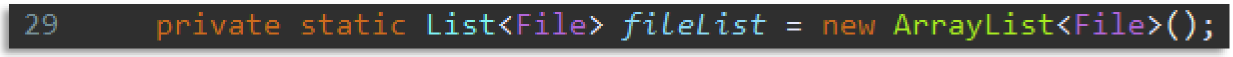
- `while` loop – Controls the program flow by prompting the User for main menu and the business options sub-menu, performing the desired operations, and terminates when the User wishes to quite the program.
- `switch` statement – Executes the desired code statements associated with the main menu and the business level options sub-menu based on the value entered by the user.

File I/O

When the application launches, the first piece of user interaction is choosing the directory the application will use. The directory will be fixed and cannot be changed during the program. I created a "test mode" for the application, using the `C:\test` directory on my laptop. However, the User can choose to use the `C:\test` directory or provide a user-specified directory. Since the directory is console input, I implemented exception handling.

Populating the list of files in the directory

Once the target directory is set, a method called `buildFileList()` will populate the `fileList` collection for the first time. The `fileList` variable is a collection (`ArrayList`) of `File` objects.



```
    private static List<File> fileList = new ArrayList<File>();
```

Since the `fileList` variable is a collection, I wanted to use the built-in `sort()` method provided by the Java API for Collections. That way, I could call the sort() method on an as-needed basis. Although I used a counter and `numFilesinDirectory` variables, I could have optimized my code by using the `size()` method provided by the `ArrayList`. Lastly, demonstrated understanding of `forEach` loops and Collection iterators.

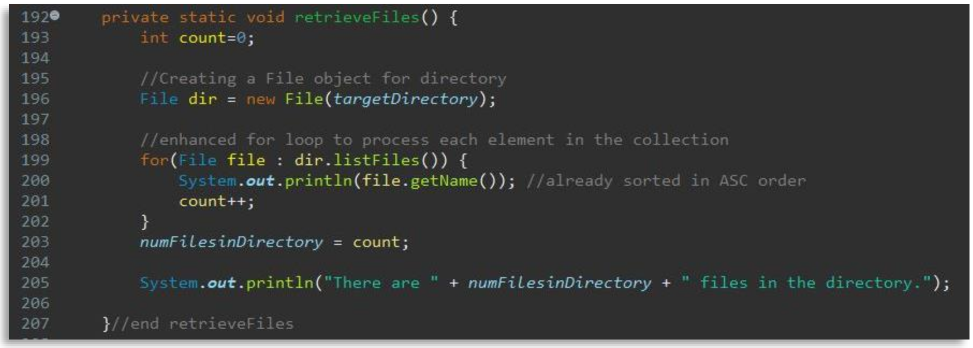
I went back and forth about which packages to use for file I/O. The `java.nio.file` is recommended. However, the Java API also provides the older class in the `java.io.File` package, which also works. I wanted to use a dedicated approach, selecting one approach for all the file I/O operations. Table 1 lists the methods that I wrote for file I/O for the application prototype, and which Java package I used.

Table 1 - Java Packages Utilized for User-Defined File I/O Methods & Variables

File I/O Method	**Java Package Utilized**	
	`java.nio.file`	`java.io.File`
`buildFileList()`		☑
`retrieveFiles()`	☑	
`addFile()`	☑	
`deleteFile()`	☑	☑
`searchForFile()`		☑ I chose to implement this package because I was directly working on File object parsing. I wanted to use the File object methods for the canonical file names.
File I/O Variables		
`myFile : Path`	☑	

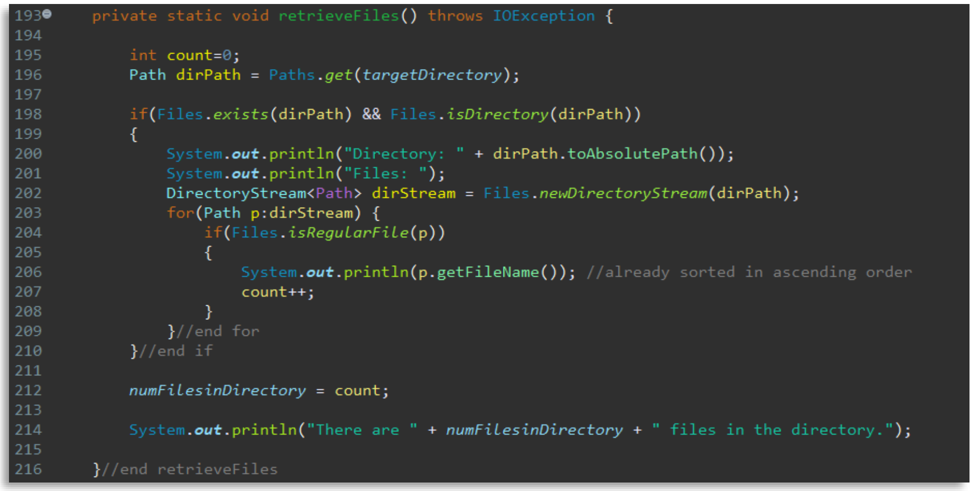
Admittedly, I am still familiarizing myself with both packages. For example, I wrote two versions of the `retrieveFiles()` method to implement the first application requirement of "retrieving the file names in an ascending order." The first version used `java.io.File` package; the second version used the `java.nio.file` package. They both work.

Previous version of `retrieveFiles()` method



```java
    private static void retrieveFiles() {
        int count=0;

        //Creating a File object for directory
        File dir = new File(targetDirectory);

        //enhanced for loop to process each element in the collection
        for(File file : dir.listFiles()) {
            System.out.println(file.getName()); //already sorted in ASC order
            count++;
        }
        numFilesinDirectory = count;

        System.out.println("There are " + numFilesinDirectory + " files in the directory.");

    }//end retrieveFiles
```

Current version of `retrieveFiles()` method



```
    private static void retrieveFiles() throws IOException {

        int count=0;
        Path dirPath = Paths.get(targetDirectory);

        if(Files.exists(dirPath) && Files.isDirectory(dirPath))
        {
            System.out.println("Directory: " + dirPath.toAbsolutePath());
            System.out.println("Files: ");
            DirectoryStream<Path> dirStream = Files.newDirectoryStream(dirPath);
            for(Path p:dirStream) {
                if(Files.isRegularFile(p))
                {
                    System.out.println(p.getFileName()); //already sorted in ascending order
                    count++;
                }
            }//end for
        }//end if

        numFilesinDirectory = count;

        System.out.println("There are " + numFilesinDirectory + " files in the directory.");

    }//end retrieveFiles
```

Program Flow Chart

Figure 2, Figure 3, and Figure 4 depict the overall program flow for the virtual keys repository application prototype.

Figure 2 - Application Flow Chart (page 1:3)

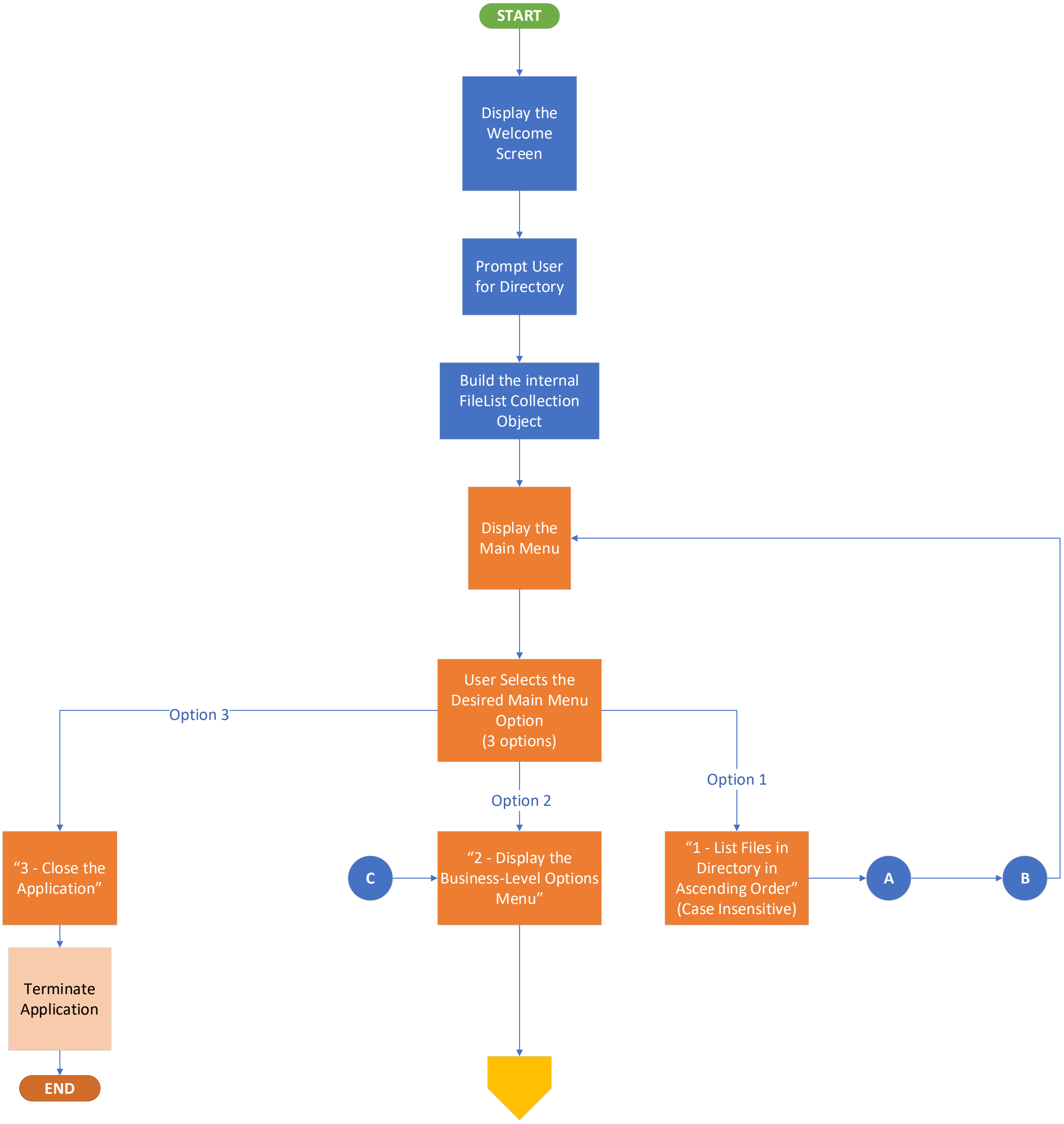

START
Display the Welcome Screen
Prompt User for Directory
Build the internal FileList Collection Object
Display the Main Menu
User Selects the Desired Main Menu Option (3 options)
Option 3
Option 2
Option 1
"3 - Close the Application"
C
"2 - Display the Business-Level Options Menu"
"1 - List Files in Directory in Ascending Order" (Case Insensitive)
A
B
Terminate Application
END

Figure 3 - Application Flow Chart (page 2:3)

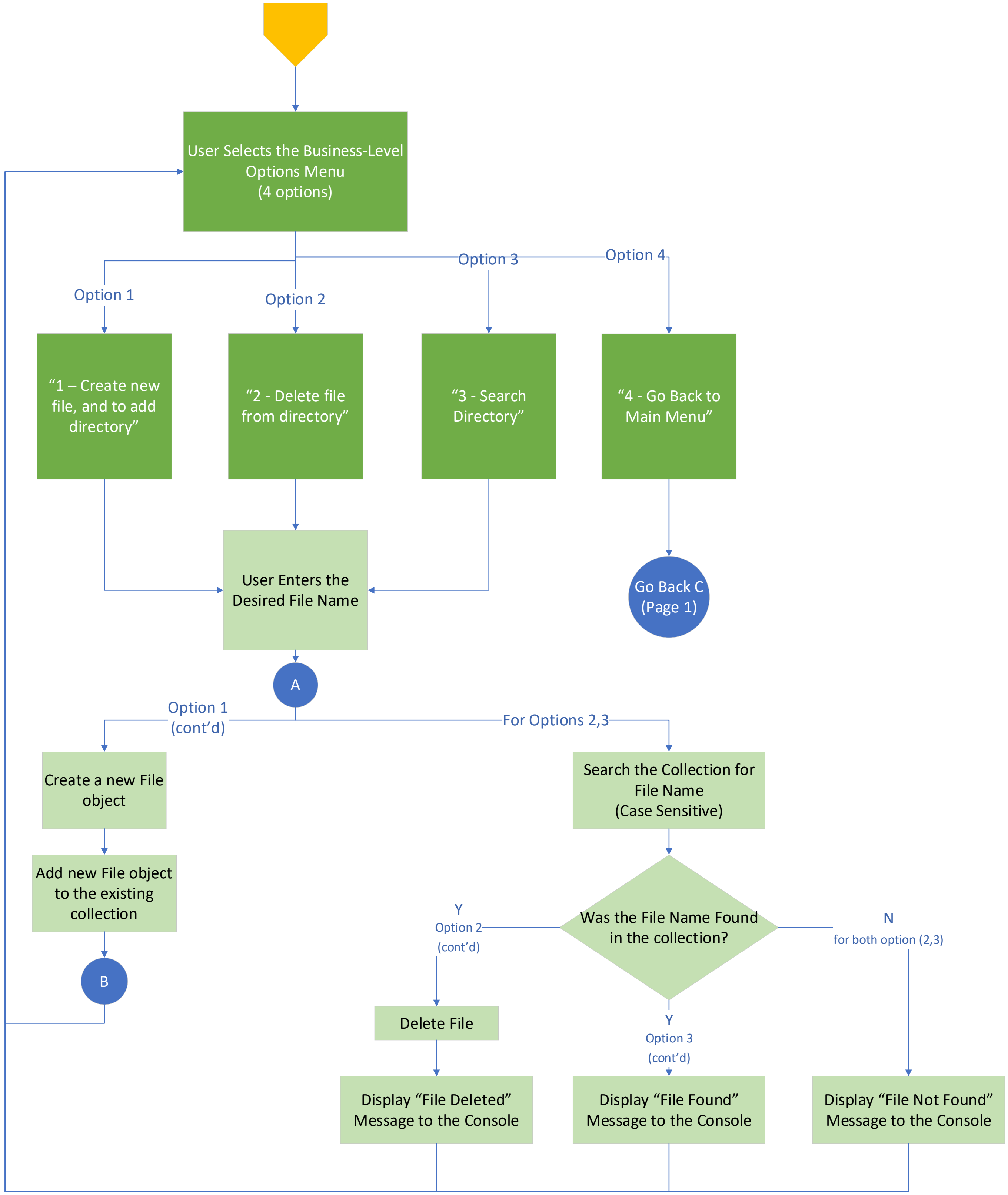
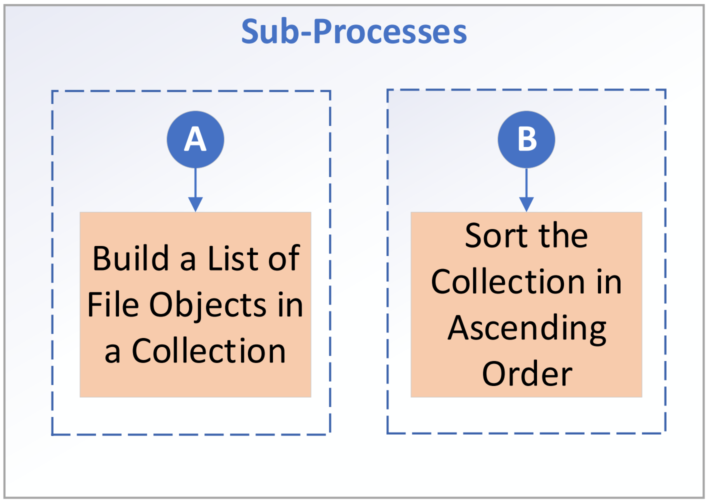

User Selects the Business-Level Options Menu (4 options)
Option 1
Option 2
Option 3
Option 4
"1 – Create new file, and to add directory"
"2 - Delete file from directory"
"3 - Search Directory"
"4 - Go Back to Main Menu"
User Enters the Desired File Name
Go Back C (Page 1)
A
Option 1 (cont'd)
For Options 2,3
Create a new File object
Add new File object to the existing collection
B
Search the Collection for File Name (Case Sensitive)
Was the File Name Found in the collection?
Y
Option 2 (cont'd)
N
for both option (2,3)
Delete File
Y
Option 3 (cont'd)
Display "File Deleted" Message to the Console
Display "File Found" Message to the Console
Display "File Not Found" Message to the Console

Figure 4 - Application Flow Chart (page 3:3)

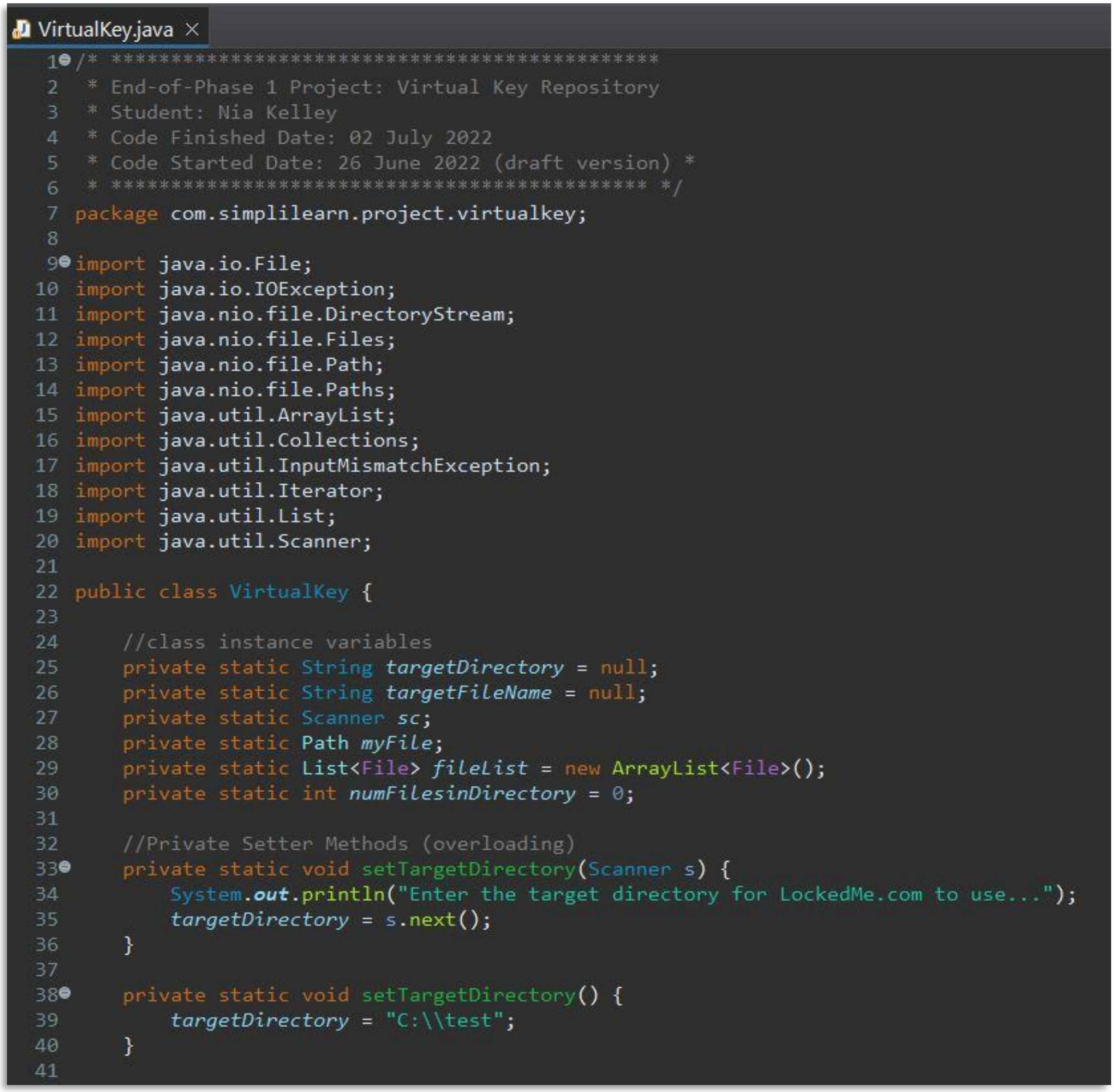

Sub-Processes
A
Build a List of File Objects in a Collection
B
Sort the Collection in Ascending Order

Code Screenshots

Approximately 390 lines of code were written to implement the application prototype.

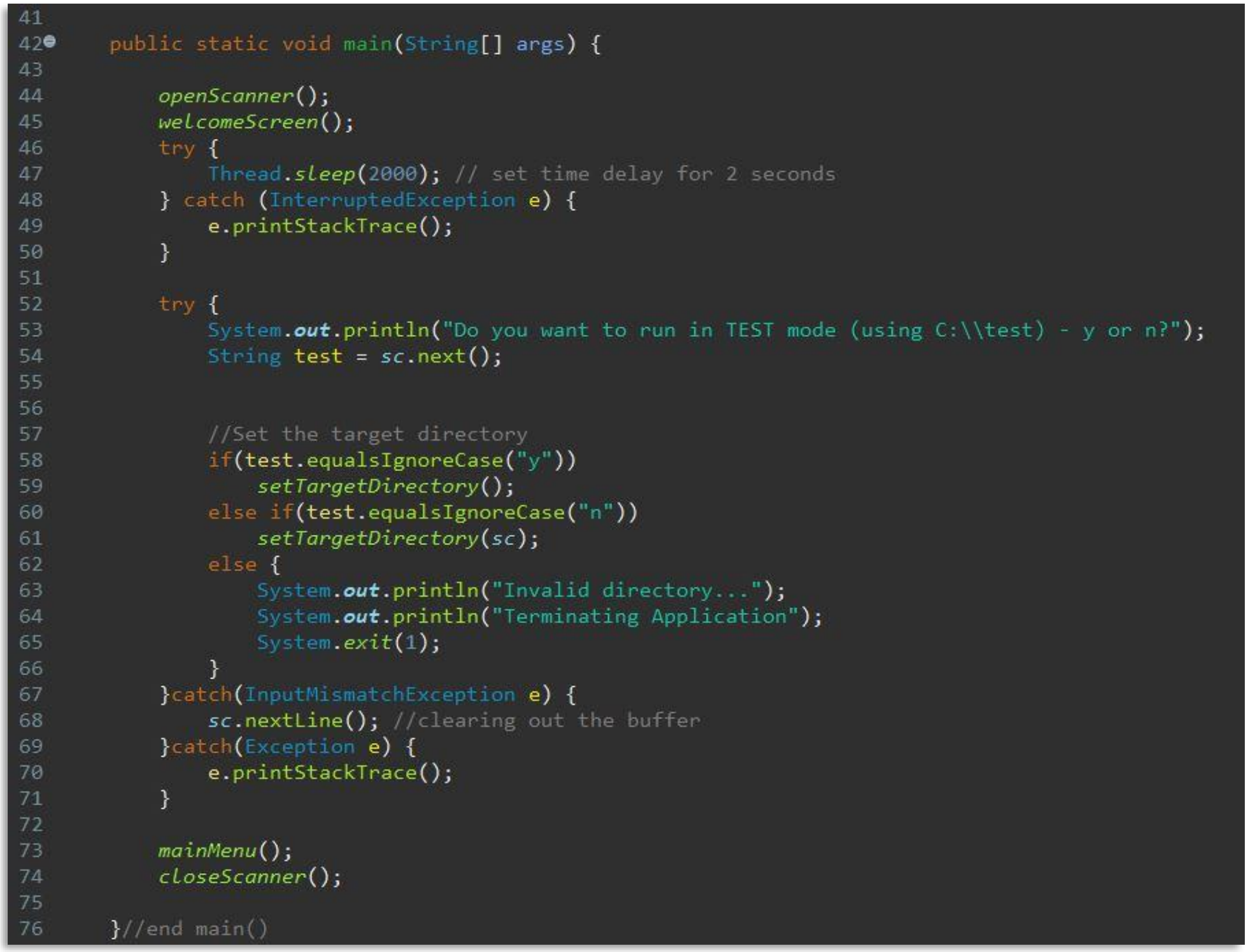
The source code is captured in the following 11 pictures in this document.

Picture 1- Code Screenshot (1:11)



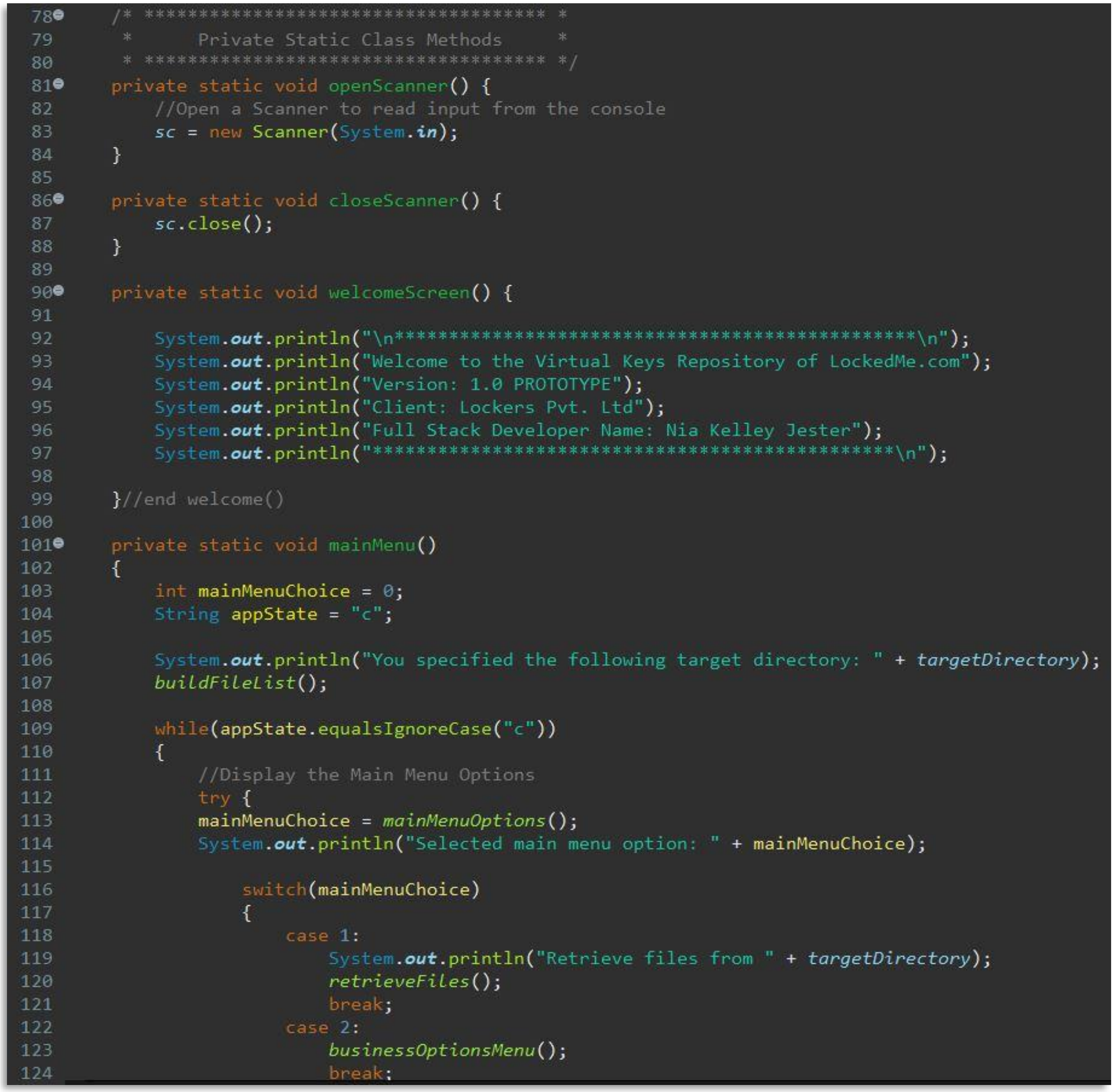
```java
/* **********************************************
 * End-of-Phase 1 Project: Virtual Key Repository
 * Student: Nia Kelley
 * Code Finished Date: 02 July 2022
 * Code Started Date: 26 June 2022 (draft version) *
 * ********************************************* */
package com.simplilearn.project.virtualkey;

import java.io.File;
import java.io.IOException;
import java.nio.file.DirectoryStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.Collections;
import java.util.InputMismatchException;
import java.util.Iterator;
import java.util.List;
import java.util.Scanner;

public class VirtualKey {

    //class instance variables
    private static String targetDirectory = null;
    private static String targetFileName = null;
    private static Scanner sc;
    private static Path myFile;
    private static List<File> fileList = new ArrayList<File>();
    private static int numFilesinDirectory = 0;

    //Private Setter Methods (overloading)
    private static void setTargetDirectory(Scanner s) {
        System.out.println("Enter the target directory for LockedMe.com to use...");
        targetDirectory = s.next();
    }

    private static void setTargetDirectory() {
        targetDirectory = "C:\\test";
    }

```

Picture 2 - Code Screenshot (2:11)



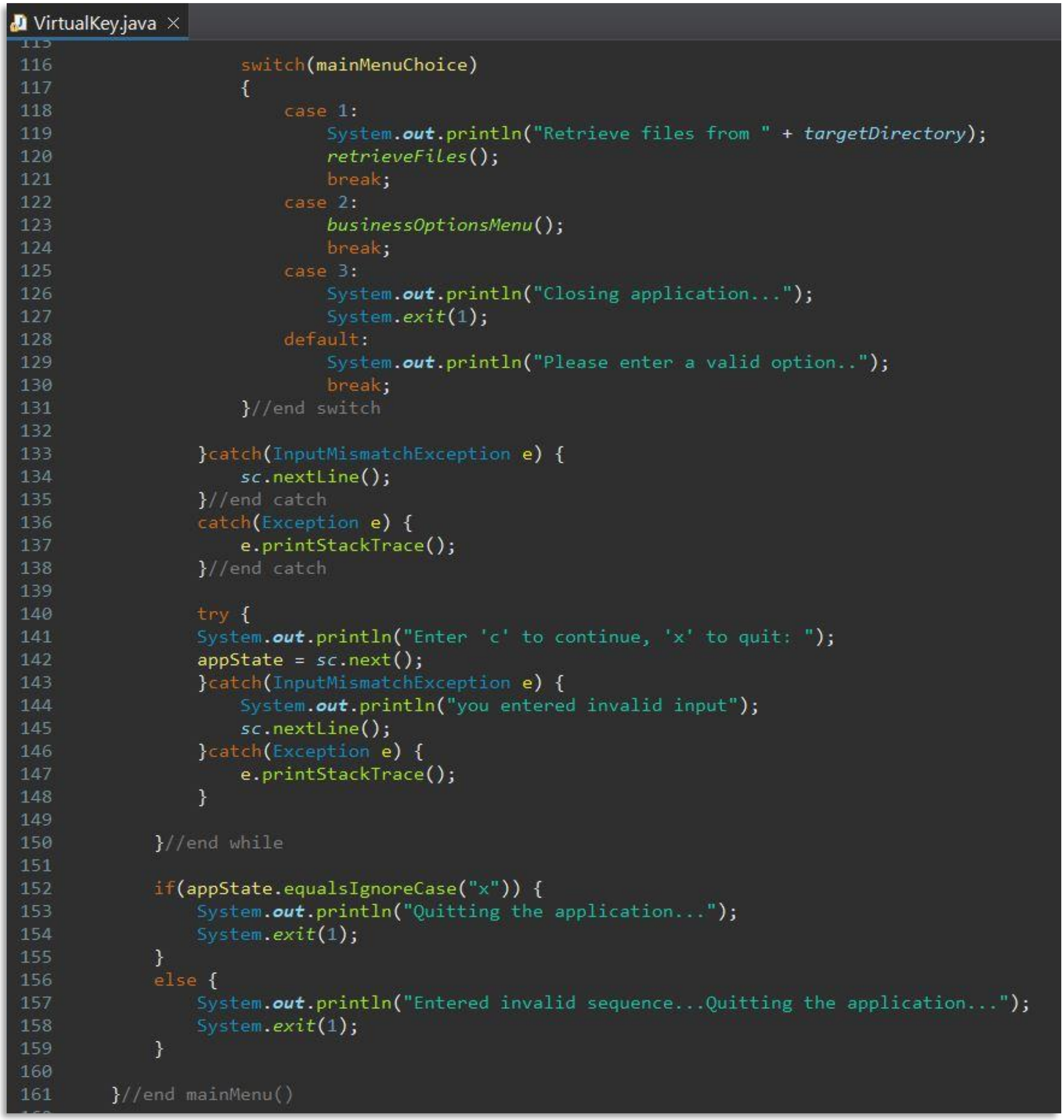
```java

    public static void main(String[] args) {

        openScanner();
        welcomeScreen();
        try {
            Thread.sleep(2000); // set time delay for 2 seconds
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        try {
            System.out.println("Do you want to run in TEST mode (using C:\\test) - y or n?");
            String test = sc.next();


            //Set the target directory
            if(test.equalsIgnoreCase("y"))
                setTargetDirectory();
            else if(test.equalsIgnoreCase("n"))
                setTargetDirectory(sc);
            else {
                System.out.println("Invalid directory...");
                System.out.println("Terminating Application");
                System.exit(1);
            }
        }catch(InputMismatchException e) {
            sc.nextLine(); //clearing out the buffer
        }catch(Exception e) {
            e.printStackTrace();
        }

        mainMenu();
        closeScanner();

    }//end main()
```

Picture 3 - Code Screenshot (3:11)



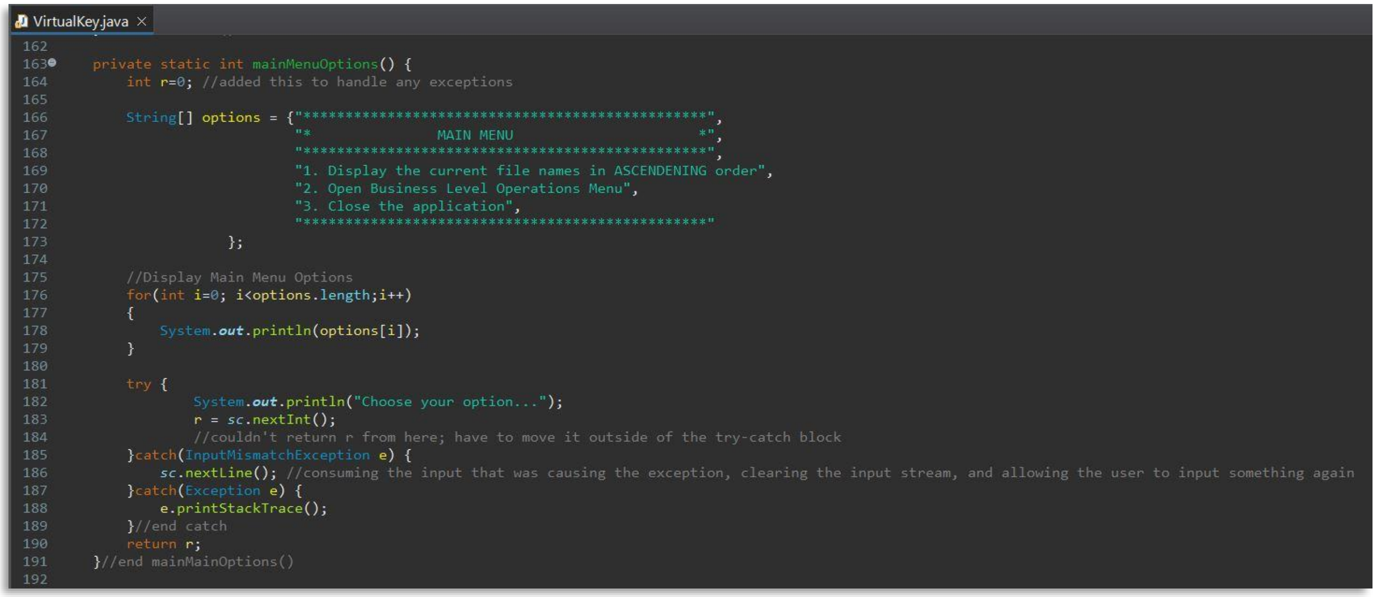
```
/* ************************************* *
 *       Private Static Class Methods      *
 * ************************************* */
private static void openScanner() {
    //Open a Scanner to read input from the console
    sc = new Scanner(System.in);
}

private static void closeScanner() {
    sc.close();
}

private static void welcomeScreen() {

    System.out.println("\n************************************************\n");
    System.out.println("Welcome to the Virtual Keys Repository of LockedMe.com");
    System.out.println("Version: 1.0 PROTOTYPE");
    System.out.println("Client: Lockers Pvt. Ltd");
    System.out.println("Full Stack Developer Name: Nia Kelley Jester");
    System.out.println("************************************************\n");

}//end welcome()

private static void mainMenu()
{
    int mainMenuChoice = 0;
    String appState = "c";

    System.out.println("You specified the following target directory: " + targetDirectory);
    buildFileList();

    while(appState.equalsIgnoreCase("c"))
    {
        //Display the Main Menu Options
        try {
        mainMenuChoice = mainMenuOptions();
        System.out.println("Selected main menu option: " + mainMenuChoice);

            switch(mainMenuChoice)
            {
                case 1:
                    System.out.println("Retrieve files from " + targetDirectory);
                    retrieveFiles();
                    break;
                case 2:
                    businessOptionsMenu();
                    break;
```

Picture 4 - Code Screenshot (4:11)



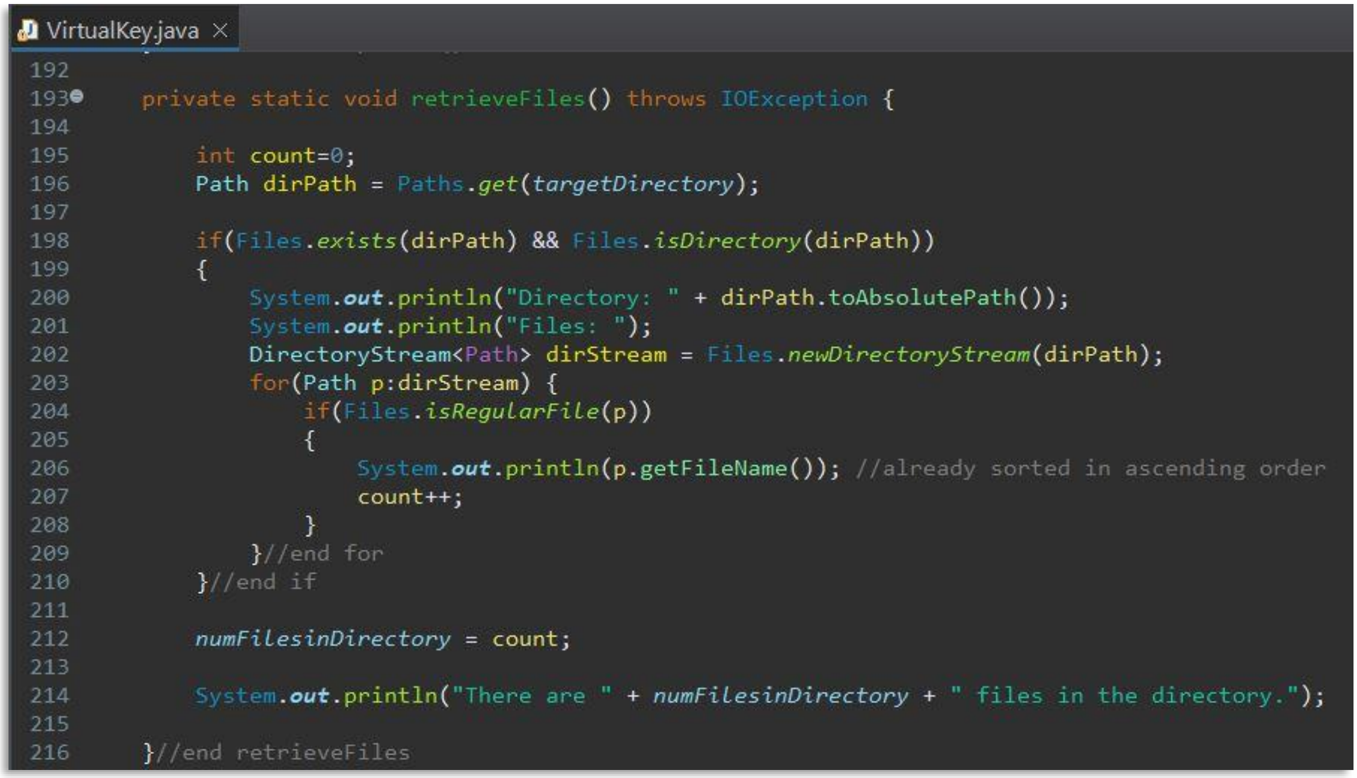
```
                switch(mainMenuChoice)
                {
                    case 1:
                        System.out.println("Retrieve files from " + targetDirectory);
                        retrieveFiles();
                        break;
                    case 2:
                        businessOptionsMenu();
                        break;
                    case 3:
                        System.out.println("Closing application...");
                        System.exit(1);
                    default:
                        System.out.println("Please enter a valid option..");
                        break;
                }//end switch

            }catch(InputMismatchException e) {
                sc.nextLine();
            }//end catch
            catch(Exception e) {
                e.printStackTrace();
            }//end catch

            try {
            System.out.println("Enter 'c' to continue, 'x' to quit: ");
            appState = sc.next();
            }catch(InputMismatchException e) {
                System.out.println("you entered invalid input");
                sc.nextLine();
            }catch(Exception e) {
                e.printStackTrace();
            }

        }//end while

        if(appState.equalsIgnoreCase("x")) {
            System.out.println("Quitting the application...");
            System.exit(1);
        }
        else {
            System.out.println("Entered invalid sequence...Quitting the application...");
            System.exit(1);
        }

    }//end mainMenu()
```

Picture 5 - Code Screenshot (5:11)



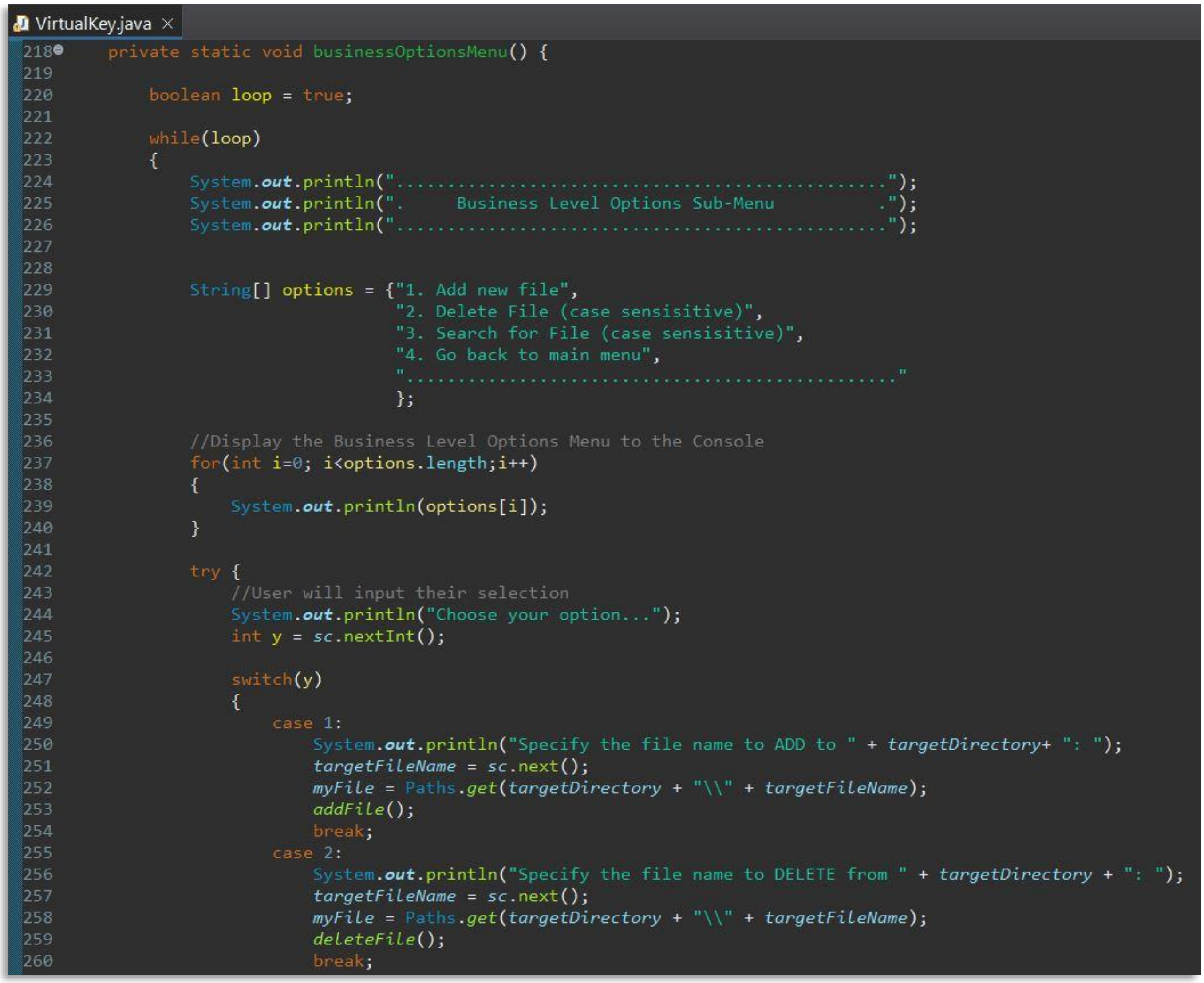
```java
    private static int mainMenuOptions() {
        int r=0; //added this to handle any exceptions

        String[] options = {"************************************************",
                            "*                 MAIN MENU                    *",
                            "************************************************",
                            "1. Display the current file names in ASCENDENING order",
                            "2. Open Business Level Operations Menu",
                            "3. Close the application",
                            "************************************************"
                    };

        //Display Main Menu Options
        for(int i=0; i<options.length;i++)
        {
            System.out.println(options[i]);
        }

        try {
                System.out.println("Choose your option...");
                r = sc.nextInt();
                //couldn't return r from here; have to move it outside of the try-catch block
        }catch(InputMismatchException e) {
            sc.nextLine(); //consuming the input that was causing the exception, clearing the input stream, and allowing the user to input something again
        }catch(Exception e) {
            e.printStackTrace();
        }//end catch
        return r;
    }//end mainMainOptions()
```

Picture 6 - Code Screenshot (6:11)



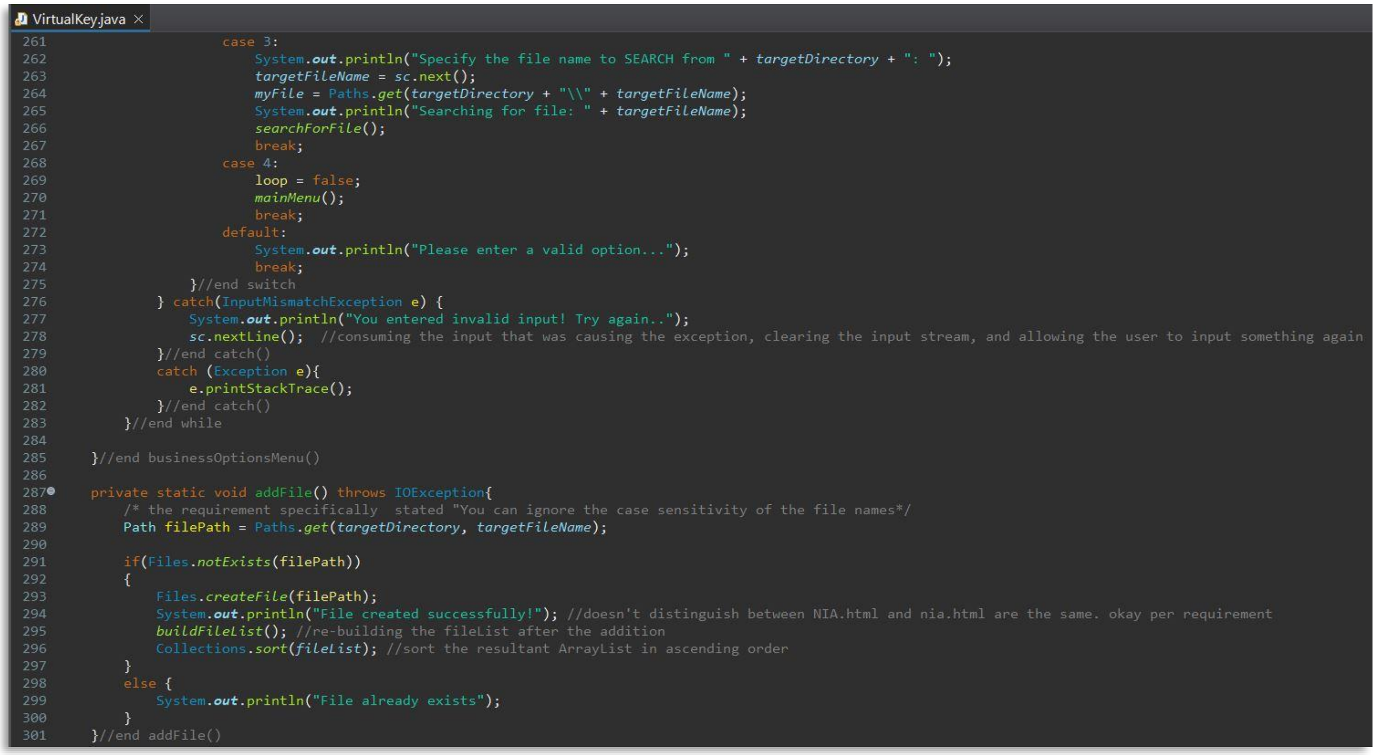
```java
    private static void retrieveFiles() throws IOException {

        int count=0;
        Path dirPath = Paths.get(targetDirectory);

        if(Files.exists(dirPath) && Files.isDirectory(dirPath))
        {
            System.out.println("Directory: " + dirPath.toAbsolutePath());
            System.out.println("Files: ");
            DirectoryStream<Path> dirStream = Files.newDirectoryStream(dirPath);
            for(Path p:dirStream) {
                if(Files.isRegularFile(p))
                {
                    System.out.println(p.getFileName()); //already sorted in ascending order
                    count++;
                }
            }//end for
        }//end if

        numFilesinDirectory = count;

        System.out.println("There are " + numFilesinDirectory + " files in the directory.");

    }//end retrieveFiles
```

Picture 7 - Code Screenshot (7:11)



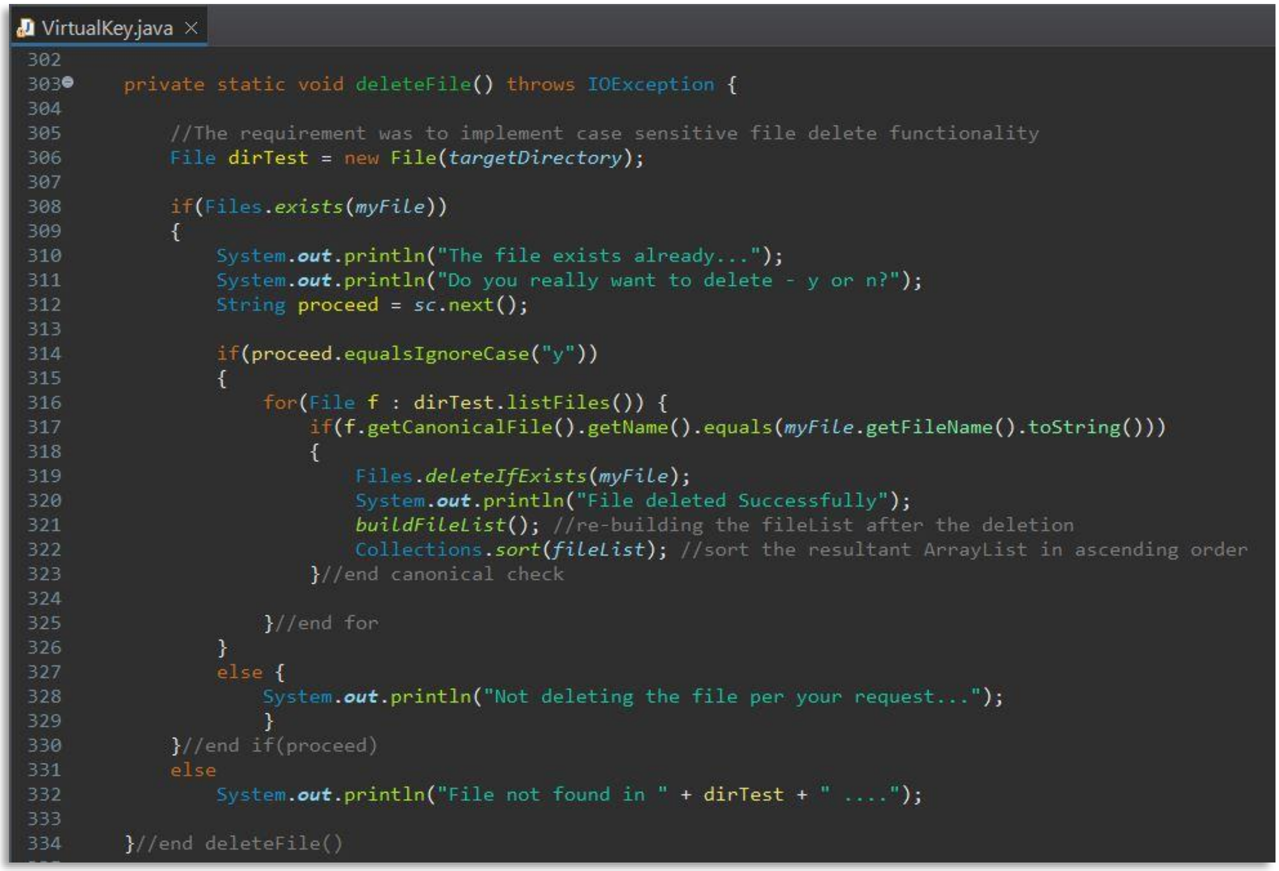
```java
    private static void businessOptionsMenu() {

        boolean loop = true;

        while(loop)
        {
            System.out.println("...............................................");
            System.out.println(".     Business Level Options Sub-Menu          .");
            System.out.println("...............................................");


            String[] options = {"1. Add new file",
                                "2. Delete File (case sensisitive)",
                                "3. Search for File (case sensisitive)",
                                "4. Go back to main menu",
                                "................................................"
                                };

            //Display the Business Level Options Menu to the Console
            for(int i=0; i<options.length;i++)
            {
                System.out.println(options[i]);
            }

            try {
                //User will input their selection
                System.out.println("Choose your option...");
                int y = sc.nextInt();

                switch(y)
                {
                    case 1:
                        System.out.println("Specify the file name to ADD to " + targetDirectory+ ": ");
                        targetFileName = sc.next();
                        myFile = Paths.get(targetDirectory + "\\" + targetFileName);
                        addFile();
                        break;
                    case 2:
                        System.out.println("Specify the file name to DELETE from " + targetDirectory + ": ");
                        targetFileName = sc.next();
                        myFile = Paths.get(targetDirectory + "\\" + targetFileName);
                        deleteFile();
                        break;
```

Picture 8 - Code Screenshot (8:11)



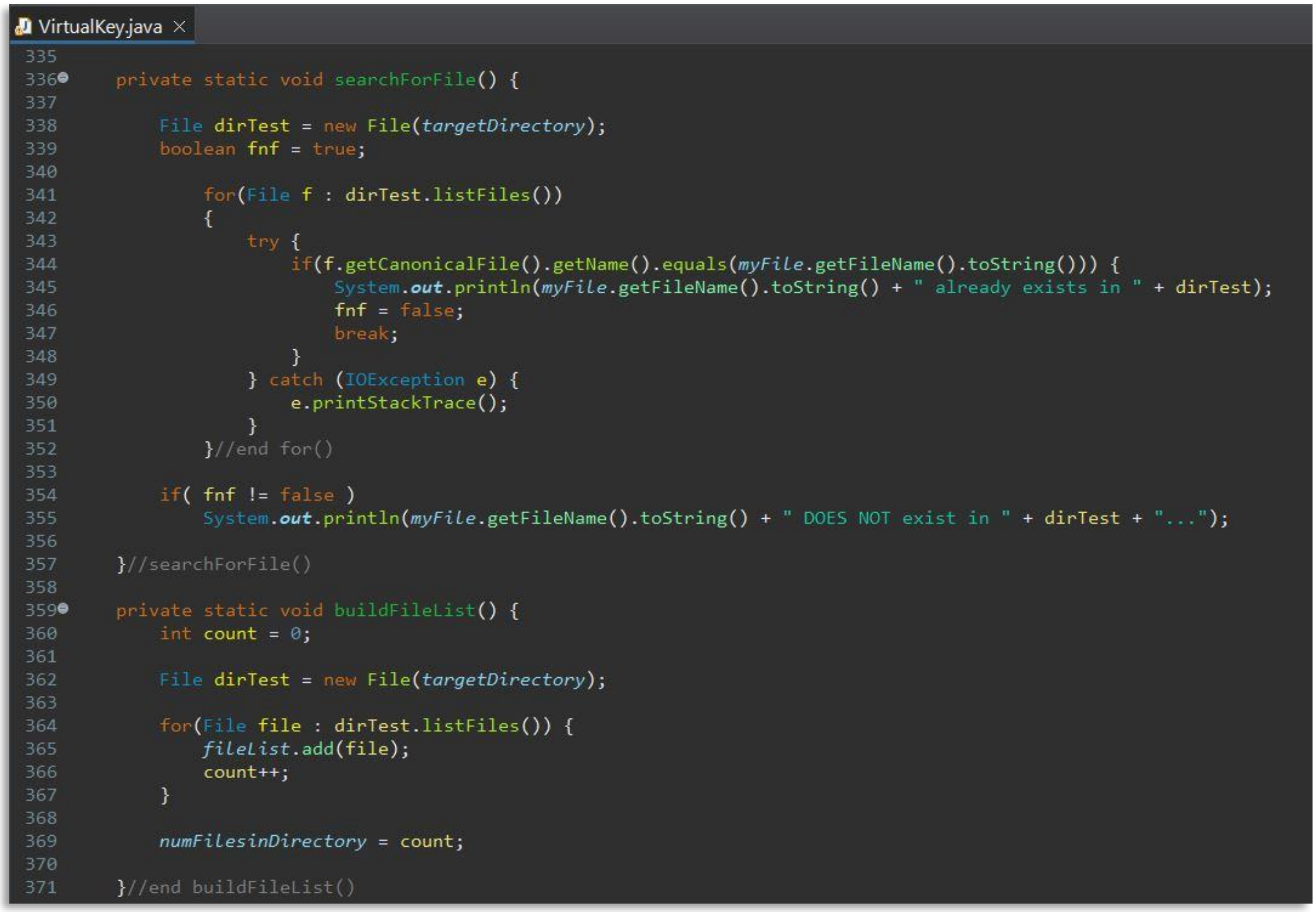
```java
                    case 3:
                        System.out.println("Specify the file name to SEARCH from " + targetDirectory + ": ");
                        targetFileName = sc.next();
                        myFile = Paths.get(targetDirectory + "\\" + targetFileName);
                        System.out.println("Searching for file: " + targetFileName);
                        searchForFile();
                        break;
                    case 4:
                        loop = false;
                        mainMenu();
                        break;
                    default:
                        System.out.println("Please enter a valid option...");
                        break;
                }//end switch
            } catch(InputMismatchException e) {
                System.out.println("You entered invalid input! Try again..");
                sc.nextLine();  //consuming the input that was causing the exception, clearing the input stream, and allowing the user to input something again
            }//end catch()
            catch (Exception e){
                e.printStackTrace();
            }//end catch()
        }//end while

    }//end businessOptionsMenu()

    private static void addFile() throws IOException{
        /* the requirement specifically  stated "You can ignore the case sensitivity of the file names*/
        Path filePath = Paths.get(targetDirectory, targetFileName);

        if(Files.notExists(filePath))
        {
            Files.createFile(filePath);
            System.out.println("File created successfully!"); //doesn't distinguish between NIA.html and nia.html are the same. okay per requirement
            buildFileList(); //re-building the fileList after the addition
            Collections.sort(fileList); //sort the resultant ArrayList in ascending order
        }
        else {
            System.out.println("File already exists");
        }
    }//end addFile()
```

Picture 9 - Code Screenshot (9:11)



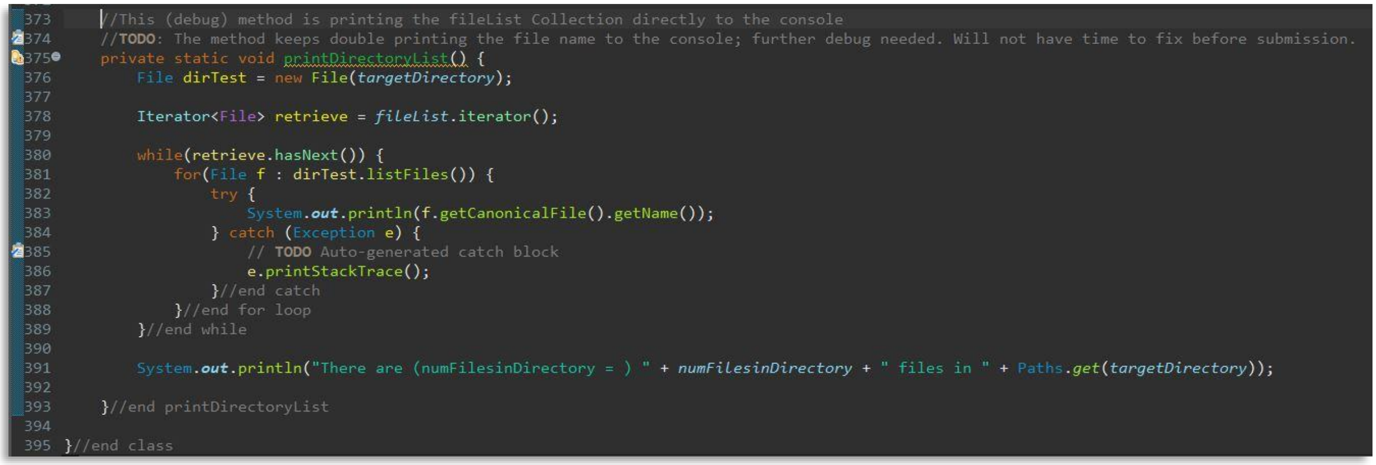
```
    private static void deleteFile() throws IOException {

        //The requirement was to implement case sensitive file delete functionality
        File dirTest = new File(targetDirectory);

        if(Files.exists(myFile))
        {
            System.out.println("The file exists already...");
            System.out.println("Do you really want to delete - y or n?");
            String proceed = sc.next();

            if(proceed.equalsIgnoreCase("y"))
            {
                for(File f : dirTest.listFiles()) {
                    if(f.getCanonicalFile().getName().equals(myFile.getFileName().toString()))
                    {
                        Files.deleteIfExists(myFile);
                        System.out.println("File deleted Successfully");
                        buildFileList(); //re-building the fileList after the deletion
                        Collections.sort(fileList); //sort the resultant ArrayList in ascending order
                    }//end canonical check

                }//end for
            }
            else {
                System.out.println("Not deleting the file per your request...");
                }
        }//end if(proceed)
        else
            System.out.println("File not found in " + dirTest + " ....");

    }//end deleteFile()
```

Picture 10- Code Screenshot (10:11)



```java

    private static void searchForFile() {

        File dirTest = new File(targetDirectory);
        boolean fnf = true;

            for(File f : dirTest.listFiles())
            {
                try {
                    if(f.getCanonicalFile().getName().equals(myFile.getFileName().toString())) {
                        System.out.println(myFile.getFileName().toString() + " already exists in " + dirTest);
                        fnf = false;
                        break;
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }//end for()

        if( fnf != false )
            System.out.println(myFile.getFileName().toString() + " DOES NOT exist in " + dirTest + "...");

    }//searchForFile()

    private static void buildFileList() {
        int count = 0;

        File dirTest = new File(targetDirectory);

        for(File file : dirTest.listFiles()) {
            fileList.add(file);
            count++;
        }

        numFilesinDirectory = count;

    }//end buildFileList()
```

Picture 11 - Code Screenshot (11:11)



```java
    //This (debug) method is printing the fileList Collection directly to the console
    //TODO: The method keeps double printing the file name to the console; further debug needed. Will not have time to fix before submission.
    private static void printDirectoryList() {
        File dirTest = new File(targetDirectory);

        Iterator<File> retrieve = fileList.iterator();

        while(retrieve.hasNext()) {
            for(File f : dirTest.listFiles()) {
                try {
                    System.out.println(f.getCanonicalFile().getName());
                } catch (Exception e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }//end catch
            }//end for loop
        }//end while

        System.out.println("There are (numFilesinDirectory = ) " + numFilesinDirectory + " files in " + Paths.get(targetDirectory));

    }//end printDirectoryList

}//end class
```

Functional Test Cases

Error! Reference source not found.his section captures the following for the LockedMe.com virtual keys repository application prototype:

- Outlines 19 functional test case scenarios
- Documents the excepted result of the test case
- Captures the test case outcome (pass/fail)
- Records the test case input
- Provides a snapshot of the program console output
- Provides the directory structure before and after snapshots (where applicable)

Functional Test Cases – “Good” Path

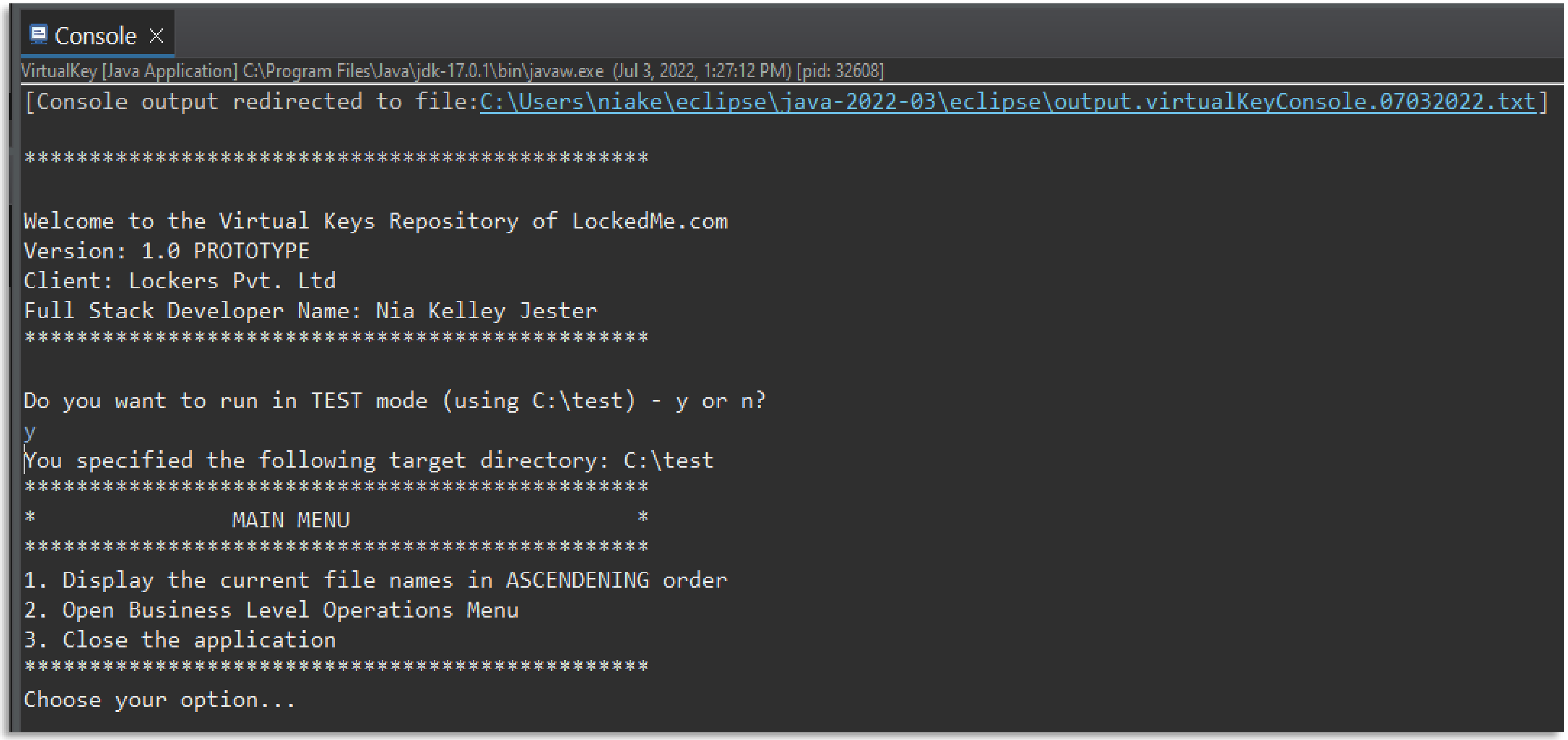
Test Case Scenario: Launch Program – Use Test Mode

Expected Result: User launches the application and indicates that the User wants to the application in Test Mode (i.e. using C:\test)

Test Outcome: Pass

Test Case Input: y

Test Case Output:

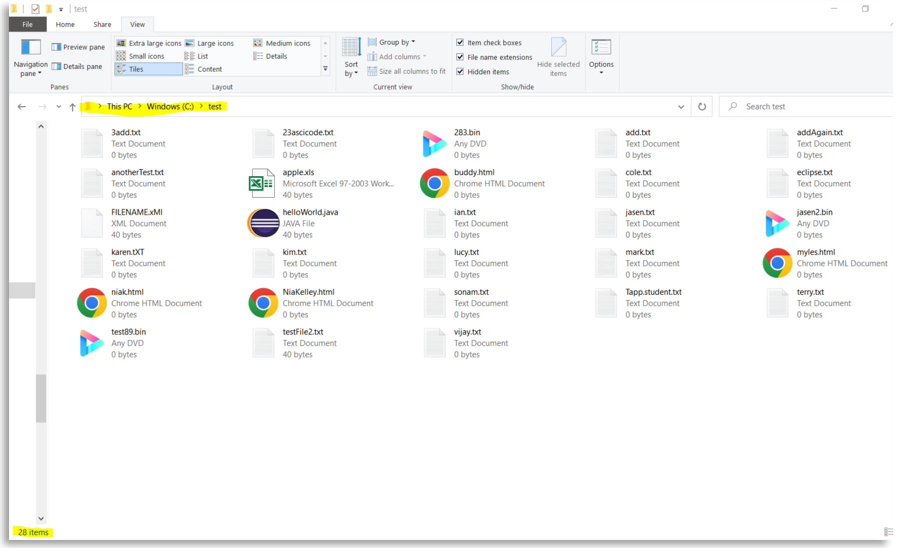
Program Console


Console

```
VirtualKey [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe  (Jul 3, 2022, 1:27:12 PM) [pid: 32608]
[Console output redirected to file:C:\Users\niake\eclipse\java-2022-03\eclipse\output.virtualKeyConsole.07032022.txt]

************************************************


Welcome to the Virtual Keys Repository of LockedMe.com
Version: 1.0 PROTOTYPE
Client: Lockers Pvt. Ltd
Full Stack Developer Name: Nia Kelley Jester
************************************************


Do you want to run in TEST mode (using C:\test) - y or n?
y
You specified the following target directory: C:\test
************************************************
*               MAIN MENU                      *
************************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
************************************************
Choose your option...
```

File Directory Structure

Before



This PC > Windows (C:) > test

3add.txt
Text Document
0 bytes

23ascicode.txt
Text Document
0 bytes

283.bin
Any DVD
0 bytes

add.txt
Text Document
0 bytes

addAgain.txt
Text Document
0 bytes

anotherTest.txt
Text Document
0 bytes

apple.xls
Microsoft Excel 97-2003 Work...
40 bytes

buddy.html
Chrome HTML Document
0 bytes

cole.txt
Text Document
0 bytes

eclipse.txt
Text Document
0 bytes

FILENAME.xMl
XML Document
40 bytes

helloWorld.java
JAVA File
40 bytes

ian.txt
Text Document
0 bytes

jasen.txt
Text Document
0 bytes

jasen2.bin
Any DVD
0 bytes

karen.tXT
Text Document
0 bytes

kim.txt
Text Document
0 bytes

lucy.txt
Text Document
0 bytes

mark.txt
Text Document
0 bytes

myles.html
Chrome HTML Document
0 bytes

niak.html
Chrome HTML Document
0 bytes

NiaKelley.html
Chrome HTML Document
0 bytes

sonam.txt
Text Document
0 bytes

Tapp.student.txt
Text Document
0 bytes

terry.txt
Text Document
0 bytes

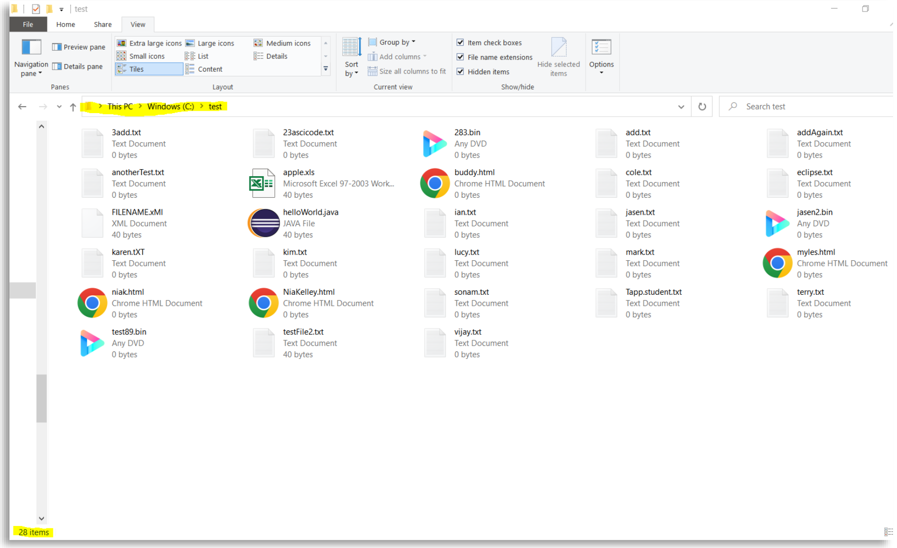
test89.bin
Any DVD
0 bytes

testFile2.txt
Text Document
40 bytes

vijay.txt
Text Document
0 bytes

28 items

After



This PC > Windows (C:) > test

3add.txt
Text Document
0 bytes

23ascicode.txt
Text Document
0 bytes

283.bin
Any DVD
0 bytes

add.txt
Text Document
0 bytes

addAgain.txt
Text Document
0 bytes

anotherTest.txt
Text Document
0 bytes

apple.xls
Microsoft Excel 97-2003 Work...
40 bytes

buddy.html
Chrome HTML Document
0 bytes

cole.txt
Text Document
0 bytes

eclipse.txt
Text Document
0 bytes

FILENAME.xMl
XML Document
40 bytes

helloWorld.java
JAVA File
40 bytes

ian.txt
Text Document
0 bytes

jasen.txt
Text Document
0 bytes

jasen2.bin
Any DVD
0 bytes

karen.tXT
Text Document
0 bytes

kim.txt
Text Document
0 bytes

lucy.txt
Text Document
0 bytes

mark.txt
Text Document
0 bytes

myles.html
Chrome HTML Document
0 bytes

niak.html
Chrome HTML Document
0 bytes

NiaKelley.html
Chrome HTML Document
0 bytes

sonam.txt
Text Document
0 bytes

Tapp.student.txt
Text Document
0 bytes

terry.txt
Text Document
0 bytes

test89.bin
Any DVD
0 bytes

testFile2.txt
Text Document
40 bytes

vijay.txt
Text Document
0 bytes

28 items

Test Case Scenario: List Files

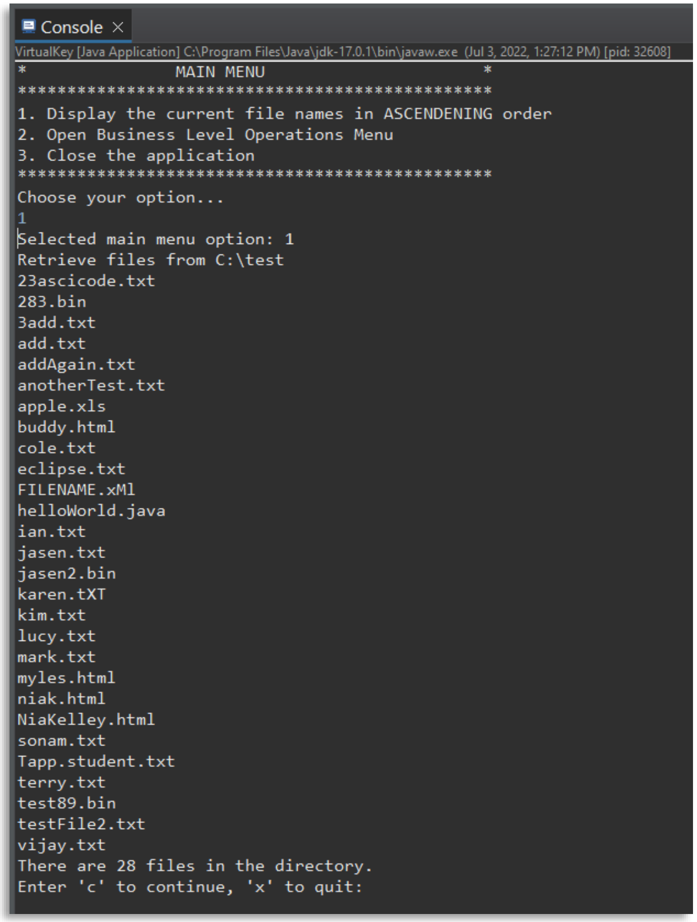
Expected Result: User opts to “display the current file names in Ascending order” from the Main Menu

Test Outcome: Pass

Test Case Input: 1

Test Case Output:

Program Console

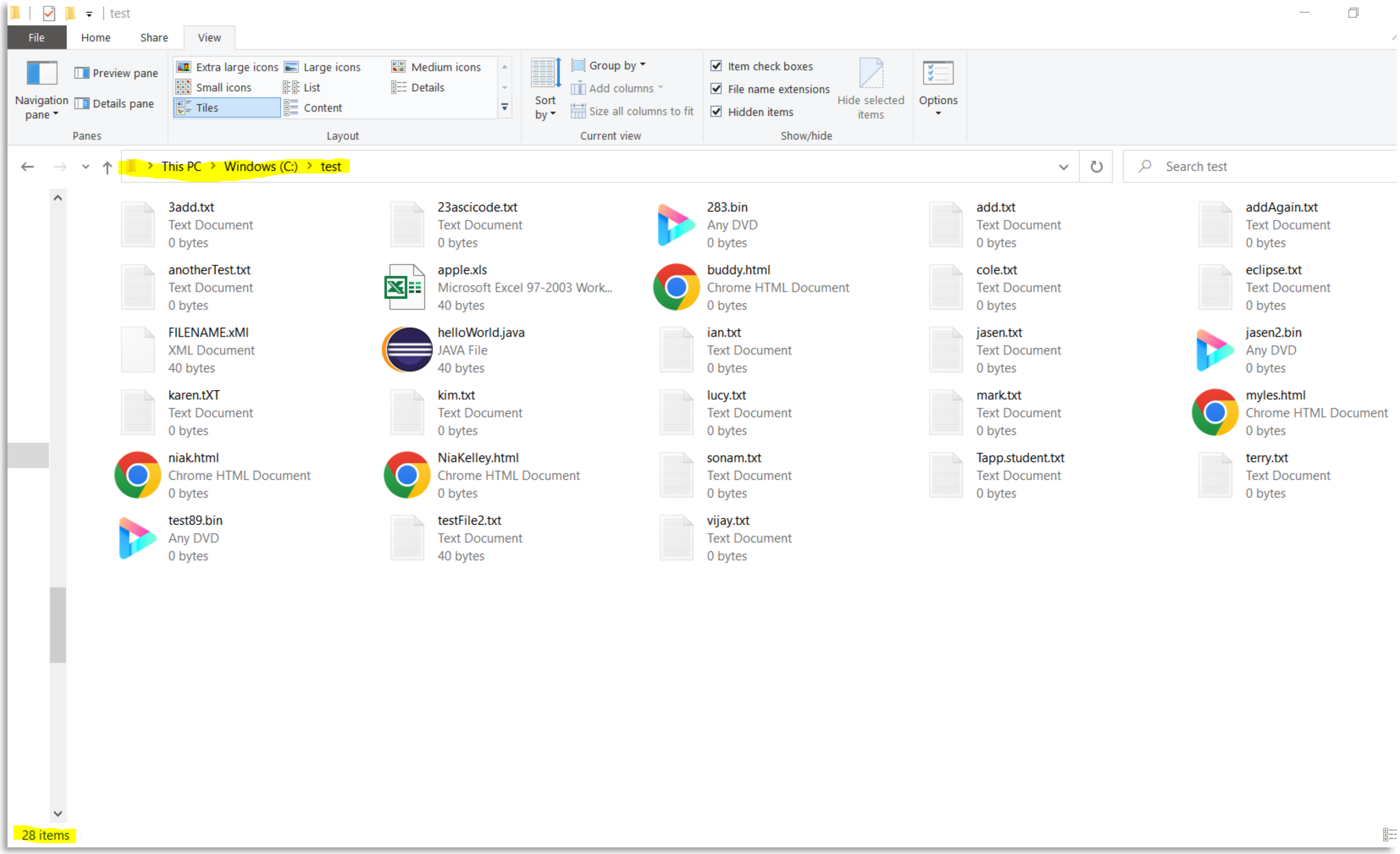


```
VirtualKey [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe  (Jul 3, 2022, 1:27:12 PM) [pid: 32608]
*               MAIN MENU                      *
************************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
************************************************
Choose your option...
1
Selected main menu option: 1
Retrieve files from C:\test
23ascicode.txt
283.bin
3add.txt
add.txt
addAgain.txt
anotherTest.txt
apple.xls
buddy.html
cole.txt
eclipse.txt
FILENAME.xMl
helloWorld.java
ian.txt
jasen.txt
jasen2.bin
karen.tXT
kim.txt
lucy.txt
mark.txt
myles.html
niak.html
NiaKelley.html
sonam.txt
Tapp.student.txt
terry.txt
test89.bin
testFile2.txt
vijay.txt
There are 28 files in the directory.
Enter 'c' to continue, 'x' to quit:
```

File Directory Structure

Before

The directory has 28 files



This PC > Windows (C:) > test

3add.txt
Text Document
0 bytes

23ascicode.txt
Text Document
0 bytes

283.bin
Any DVD
0 bytes

add.txt
Text Document
0 bytes

addAgain.txt
Text Document
0 bytes

anotherTest.txt
Text Document
0 bytes

apple.xls
Microsoft Excel 97-2003 Work...
40 bytes

buddy.html
Chrome HTML Document
0 bytes

cole.txt
Text Document
0 bytes

eclipse.txt
Text Document
0 bytes

FILENAME.xMl
XML Document
40 bytes

helloWorld.java
JAVA File
40 bytes

ian.txt
Text Document
0 bytes

jasen.txt
Text Document
0 bytes

jasen2.bin
Any DVD
0 bytes

karen.tXT
Text Document
0 bytes

kim.txt
Text Document
0 bytes

lucy.txt
Text Document
0 bytes

mark.txt
Text Document
0 bytes

myles.html
Chrome HTML Document
0 bytes

niak.html
Chrome HTML Document
0 bytes

NiaKelley.html
Chrome HTML Document
0 bytes

sonam.txt
Text Document
0 bytes

Tapp.student.txt
Text Document
0 bytes

terry.txt
Text Document
0 bytes

test89.bin
Any DVD
0 bytes

testFile2.txt
Text Document
40 bytes

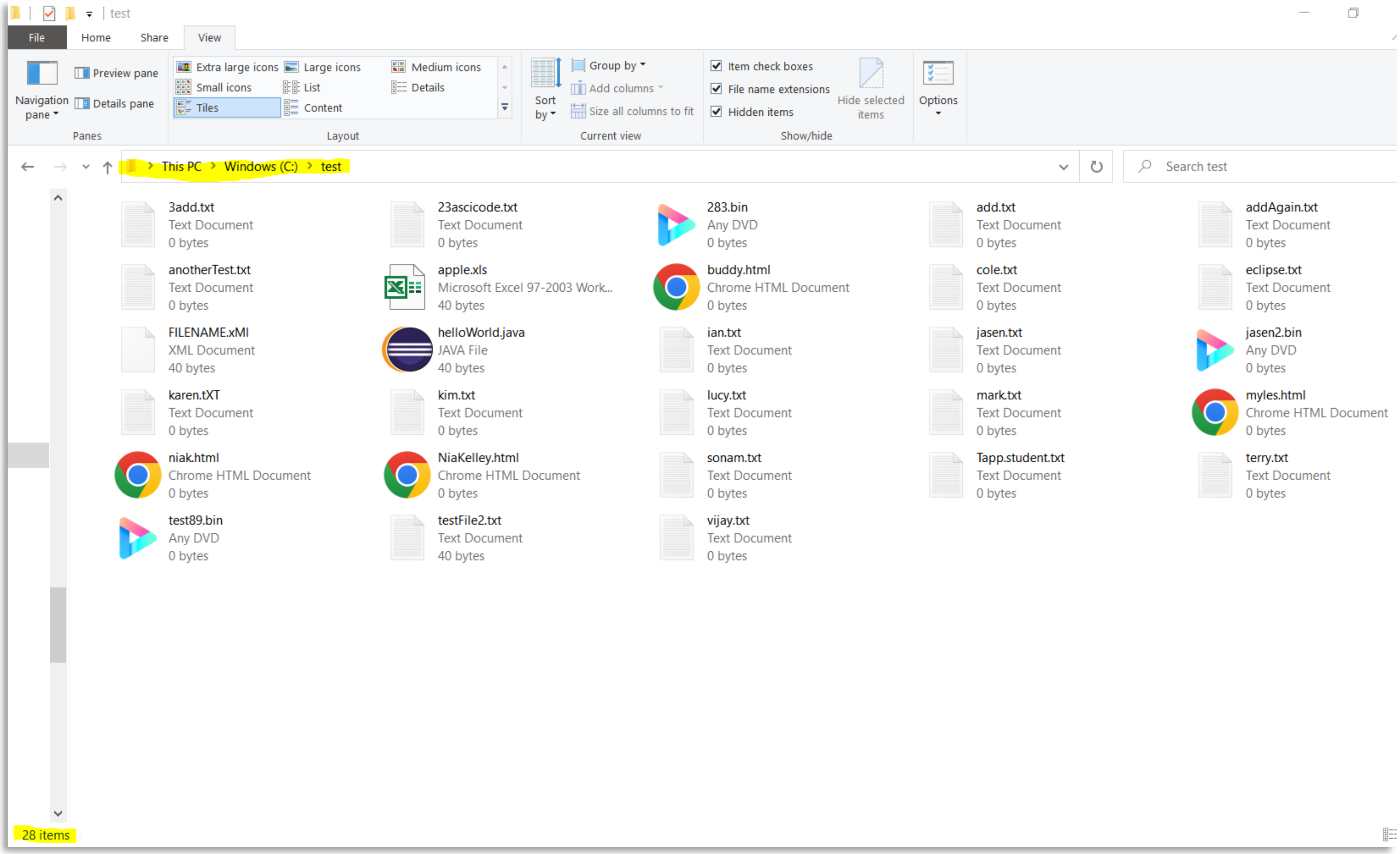
vijay.txt
Text Document
0 bytes

28 items

After

File Directory Structure

The directory has 28 files



☑ Item check boxes
☑ File name extensions
☑ Hidden items

This PC > Windows (C:) > test

3add.txt
Text Document
0 bytes

23ascicode.txt
Text Document
0 bytes

283.bin
Any DVD
0 bytes

add.txt
Text Document
0 bytes

addAgain.txt
Text Document
0 bytes

anotherTest.txt
Text Document
0 bytes

apple.xls
Microsoft Excel 97-2003 Work...
40 bytes

buddy.html
Chrome HTML Document
0 bytes

cole.txt
Text Document
0 bytes

eclipse.txt
Text Document
0 bytes

FILENAME.xMl
XML Document
40 bytes

helloWorld.java
JAVA File
40 bytes

ian.txt
Text Document
0 bytes

jasen.txt
Text Document
0 bytes

jasen2.bin
Any DVD
0 bytes

karen.tXT
Text Document
0 bytes

kim.txt
Text Document
0 bytes

lucy.txt
Text Document
0 bytes

mark.txt
Text Document
0 bytes

myles.html
Chrome HTML Document
0 bytes

niak.html
Chrome HTML Document
0 bytes

NiaKelley.html
Chrome HTML Document
0 bytes

sonam.txt
Text Document
0 bytes

Tapp.student.txt
Text Document
0 bytes

terry.txt
Text Document
0 bytes

test89.bin
Any DVD
0 bytes

testFile2.txt
Text Document
40 bytes

vijay.txt
Text Document
0 bytes

28 items

Test Case Scenario: Continue Program after List Files

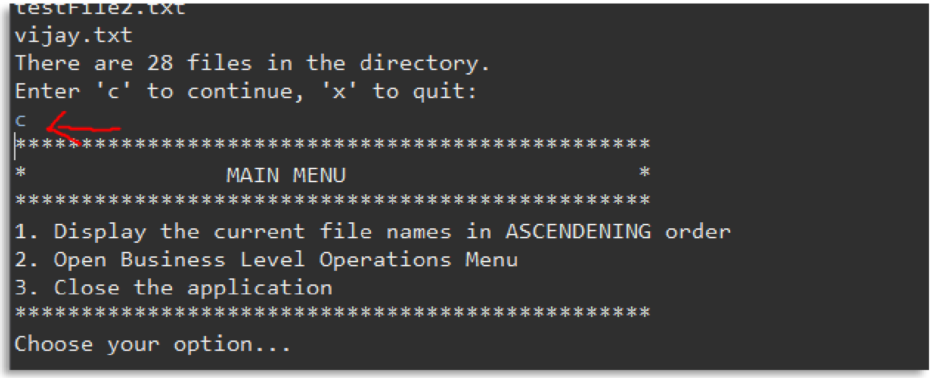
Expected Result: User opts to continue the program after the list files operation. The program will accept “C” or “c” to continue the program.

Test Outcome: Pass

Test Case Input: c

Test Case Output:

Program Console



```
testFile2.txt
vijay.txt
There are 28 files in the directory.
Enter 'c' to continue, 'x' to quit:
c
************************************************
*               MAIN MENU                      *
************************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
************************************************
Choose your option...
```

File Directory Structure: No need to display for this test

Test Case Scenario: Open Business Level Options Menu

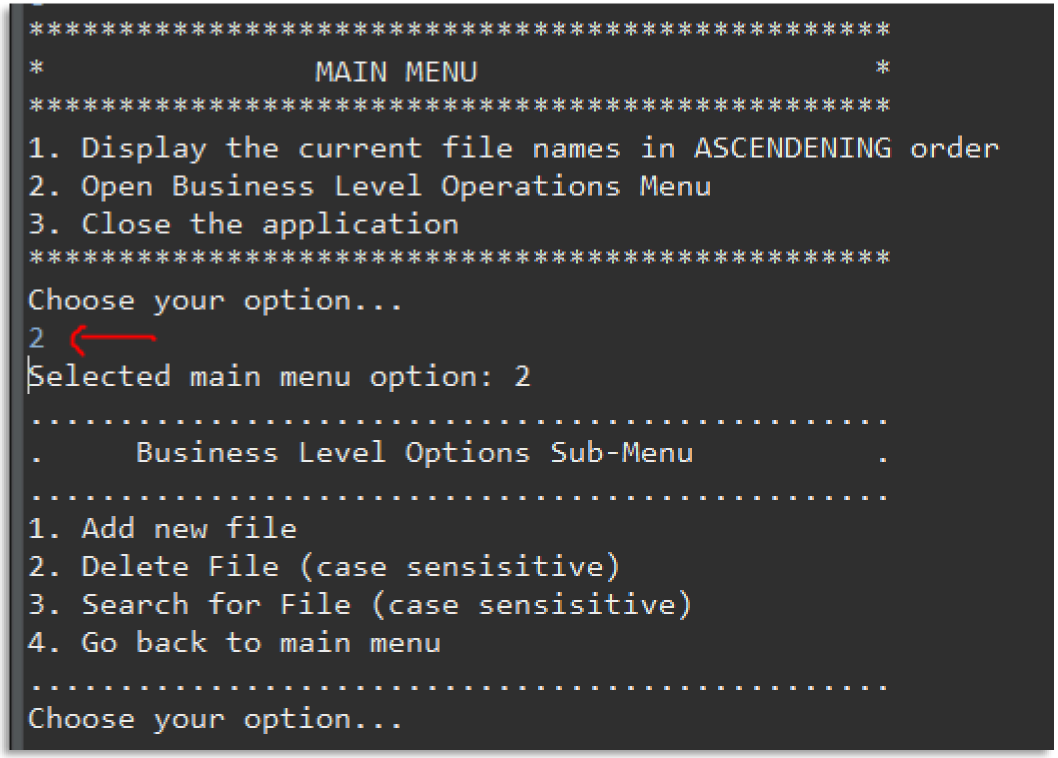
Expected Result: User opts to open the Business Level Options Sub-Menu.

Test Outcome: Pass

Test Case Input: 2

Test Case Output:

Program Console



```
************************************************
*               MAIN MENU                      *
************************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
************************************************
Choose your option...
2 ←
Selected main menu option: 2
................................................
.     Business Level Options Sub-Menu          .
................................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
................................................
Choose your option...
```

File Directory Structure: No need to display for this test

Test Case Scenario: Add New File

Expected Result: User wants to add a new file to the directory. The User will input the desired file name for addition, and the application will create the file in the directory. Per the requirement, the application ignores case sensitivity for this feature.

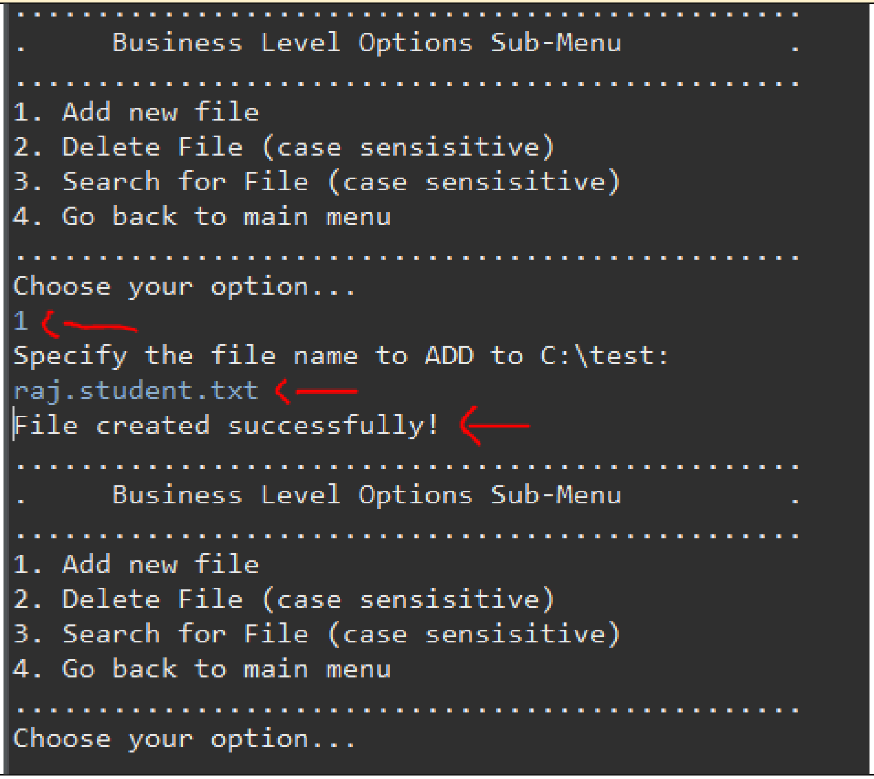
Test Outcome: Pass

Test Case Input:

- 1
- raj.student.txt

Test Case Output:

Program Console

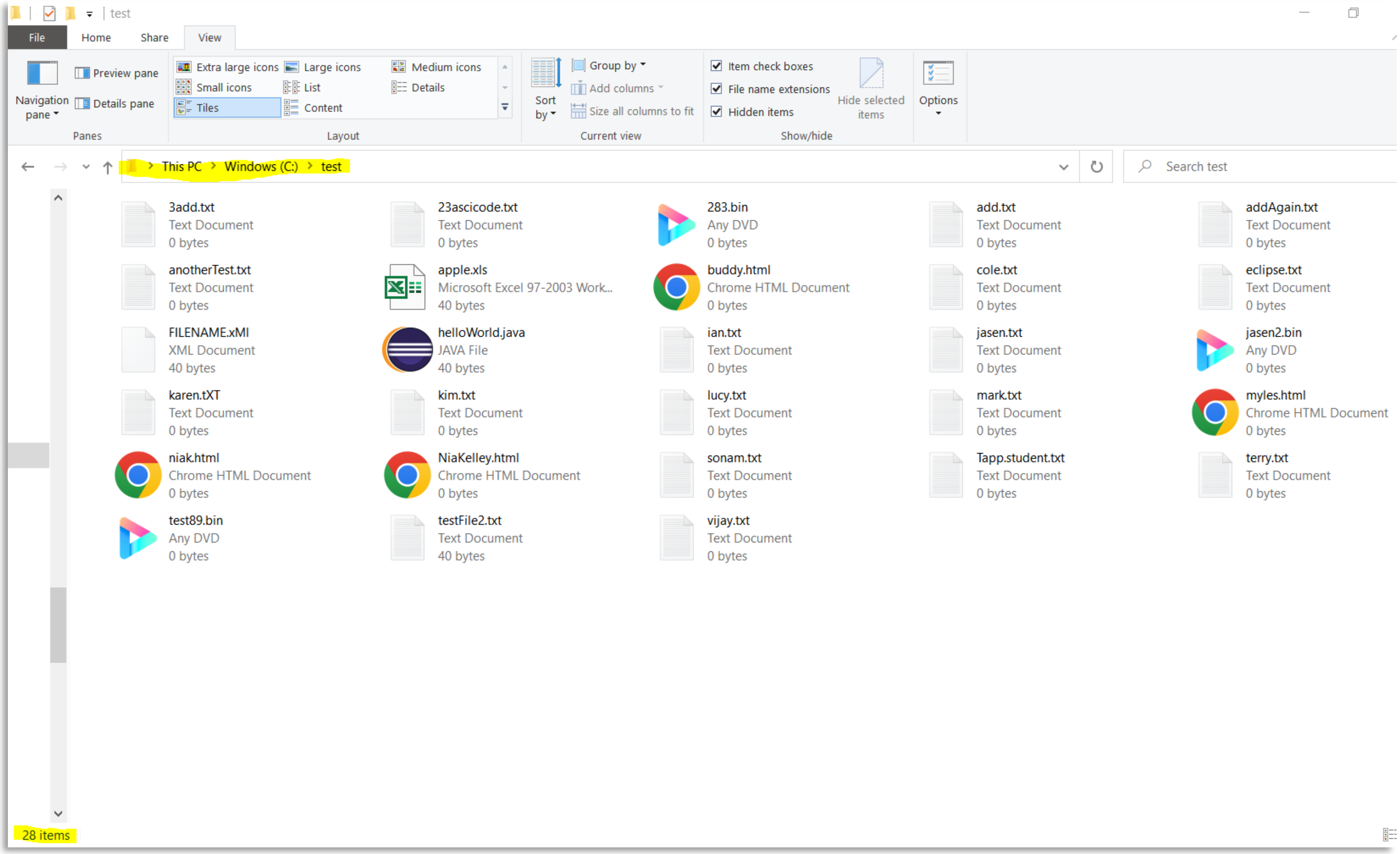


```
................................................
.     Business Level Options Sub-Menu          .
................................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
................................................
Choose your option...
1
Specify the file name to ADD to C:\test:
raj.student.txt
File created successfully!
................................................
.     Business Level Options Sub-Menu          .
................................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
................................................
Choose your option...
```

File Directory Structure

Before

The directory has 28 files



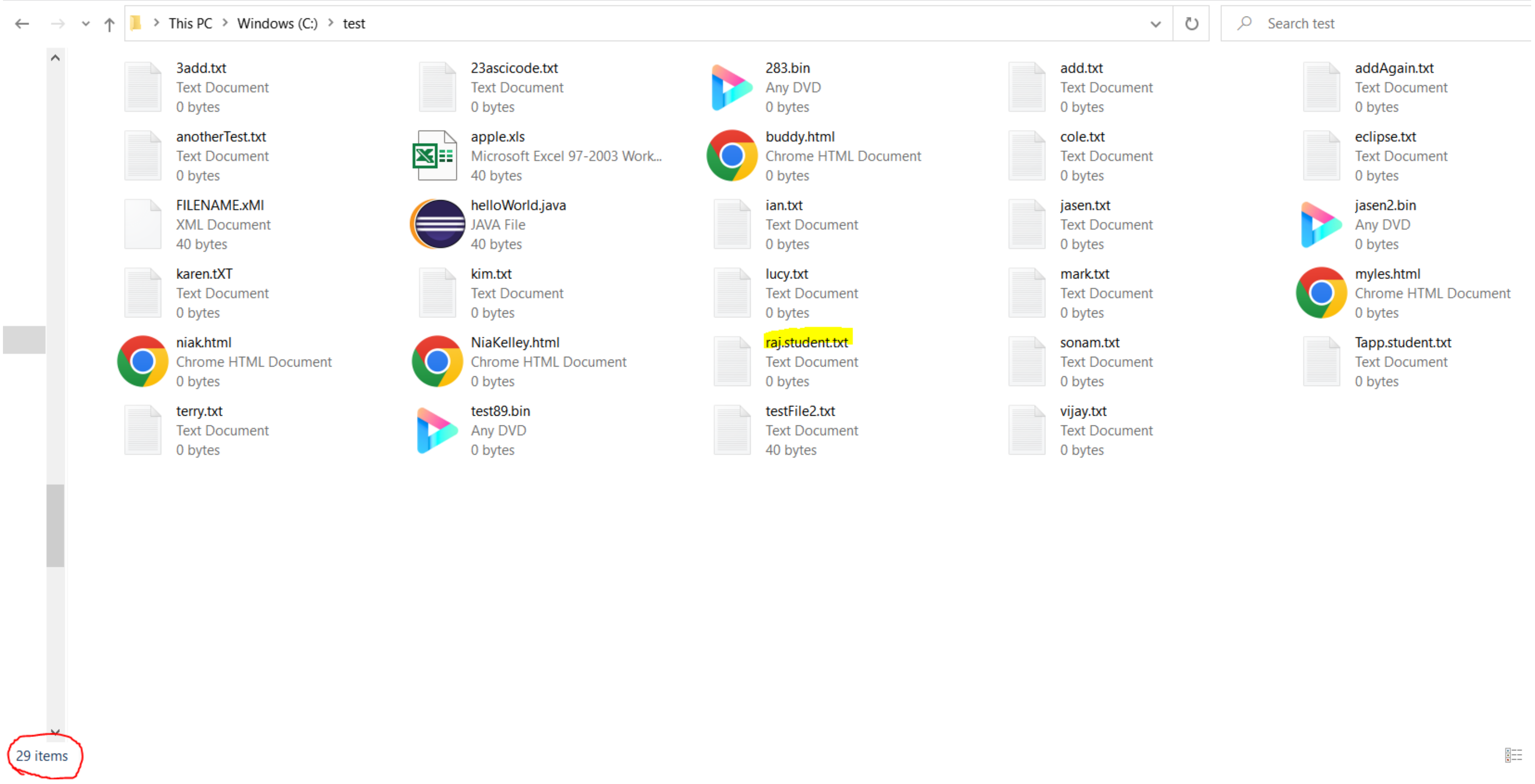
3add.txt Text Document 0 bytes	23ascicode.txt Text Document 0 bytes	283.bin Any DVD 0 bytes	add.txt Text Document 0 bytes	addAgain.txt Text Document 0 bytes
anotherTest.txt Text Document 0 bytes	apple.xls Microsoft Excel 97-2003 Work... 40 bytes	buddy.html Chrome HTML Document 0 bytes	cole.txt Text Document 0 bytes	eclipse.txt Text Document 0 bytes
FILENAME.xMl XML Document 40 bytes	helloWorld.java JAVA File 40 bytes	ian.txt Text Document 0 bytes	jasen.txt Text Document 0 bytes	jasen2.bin Any DVD 0 bytes
karen.tXT Text Document 0 bytes	kim.txt Text Document 0 bytes	lucy.txt Text Document 0 bytes	mark.txt Text Document 0 bytes	myles.html Chrome HTML Document 0 bytes
niak.html Chrome HTML Document 0 bytes	NiaKelley.html Chrome HTML Document 0 bytes	sonam.txt Text Document 0 bytes	Tapp.student.txt Text Document 0 bytes	terry.txt Text Document 0 bytes
test89.bin Any DVD 0 bytes	testFile2.txt Text Document 40 bytes	vijay.txt Text Document 0 bytes		

28 items

After

File Directory Structure

The directory has 29 files



3add.txt
Text Document
0 bytes

23ascicode.txt
Text Document
0 bytes

283.bin
Any DVD
0 bytes

add.txt
Text Document
0 bytes

addAgain.txt
Text Document
0 bytes

anotherTest.txt
Text Document
0 bytes

apple.xls
Microsoft Excel 97-2003 Work...
40 bytes

buddy.html
Chrome HTML Document
0 bytes

cole.txt
Text Document
0 bytes

eclipse.txt
Text Document
0 bytes

FILENAME.xMl
XML Document
40 bytes

helloWorld.java
JAVA File
40 bytes

ian.txt
Text Document
0 bytes

jasen.txt
Text Document
0 bytes

jasen2.bin
Any DVD
0 bytes

karen.tXT
Text Document
0 bytes

kim.txt
Text Document
0 bytes

lucy.txt
Text Document
0 bytes

mark.txt
Text Document
0 bytes

myles.html
Chrome HTML Document
0 bytes

niak.html
Chrome HTML Document
0 bytes

NiaKelley.html
Chrome HTML Document
0 bytes

raj.student.txt
Text Document
0 bytes

sonam.txt
Text Document
0 bytes

Tapp.student.txt
Text Document
0 bytes

terry.txt
Text Document
0 bytes

test89.bin
Any DVD
0 bytes

testFile2.txt
Text Document
40 bytes

vijay.txt
Text Document
0 bytes

29 items

Test Case Scenario: Delete File – File Found – Proceed with Deletion

Expected Result: User inputs a file name for deletion. The application conducts a case sensitive search to determine if the file name already exists. If the file name already exists, the applicate will display a message and will not delete the file. If the file name does not exist, the application asks the User to confirm the deletion. Upon indicating “Y” or “y”, the application will delete the file from the directory, and displays a confirmation message on the console.

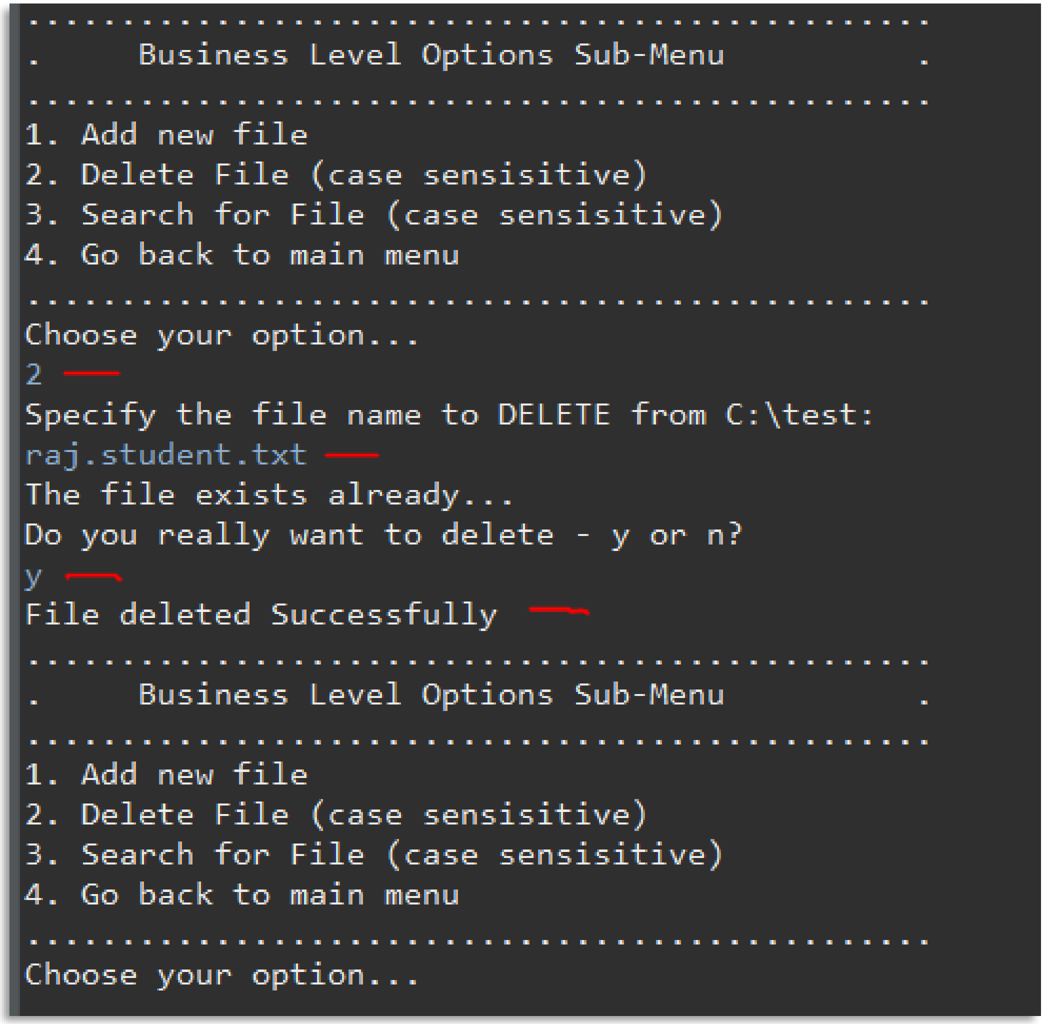
Test Outcome: Pass

Test Case Input:

- 2
- raj.student.txt
- y

Test Case Output:

Program Console

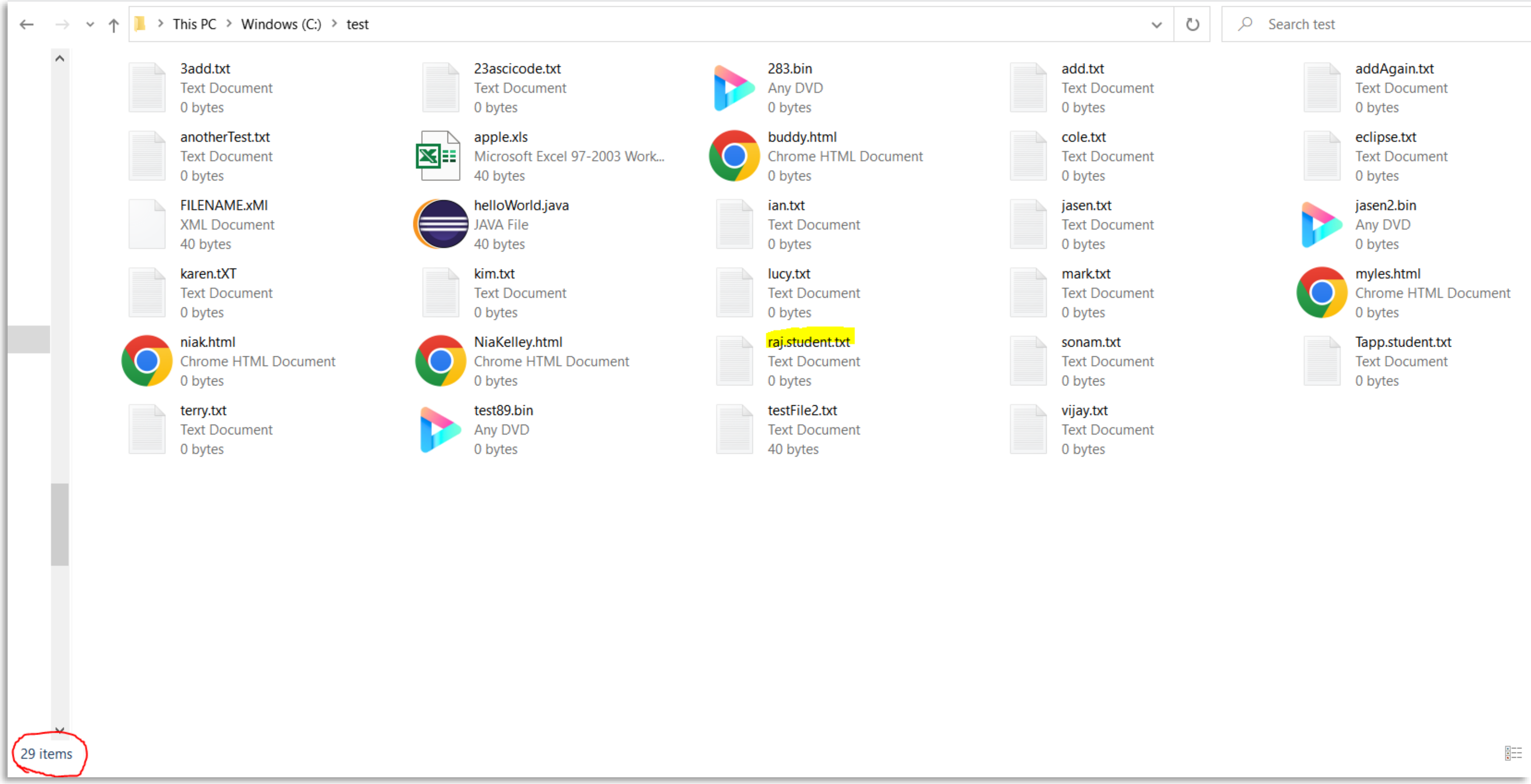


```
................................................
.     Business Level Options Sub-Menu          .
................................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
................................................
Choose your option...
2
Specify the file name to DELETE from C:\test:
raj.student.txt
The file exists already...
Do you really want to delete - y or n?
y
File deleted Successfully
................................................
.     Business Level Options Sub-Menu          .
................................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
................................................
Choose your option...
```

File Directory Structure

Before

The directory has 29 files



This PC > Windows (C:) > test

Search test

3add.txt
Text Document
0 bytes

23ascicode.txt
Text Document
0 bytes

283.bin
Any DVD
0 bytes

add.txt
Text Document
0 bytes

addAgain.txt
Text Document
0 bytes

anotherTest.txt
Text Document
0 bytes

apple.xls
Microsoft Excel 97-2003 Work...
40 bytes

buddy.html
Chrome HTML Document
0 bytes

cole.txt
Text Document
0 bytes

eclipse.txt
Text Document
0 bytes

FILENAME.xMl
XML Document
40 bytes

helloWorld.java
JAVA File
40 bytes

ian.txt
Text Document
0 bytes

jasen.txt
Text Document
0 bytes

jasen2.bin
Any DVD
0 bytes

karen.tXT
Text Document
0 bytes

kim.txt
Text Document
0 bytes

lucy.txt
Text Document
0 bytes

mark.txt
Text Document
0 bytes

myles.html
Chrome HTML Document
0 bytes

niak.html
Chrome HTML Document
0 bytes

NiaKelley.html
Chrome HTML Document
0 bytes

raj.student.txt
Text Document
0 bytes

sonam.txt
Text Document
0 bytes

Tapp.student.txt
Text Document
0 bytes

terry.txt
Text Document
0 bytes

test89.bin
Any DVD
0 bytes

testFile2.txt
Text Document
40 bytes

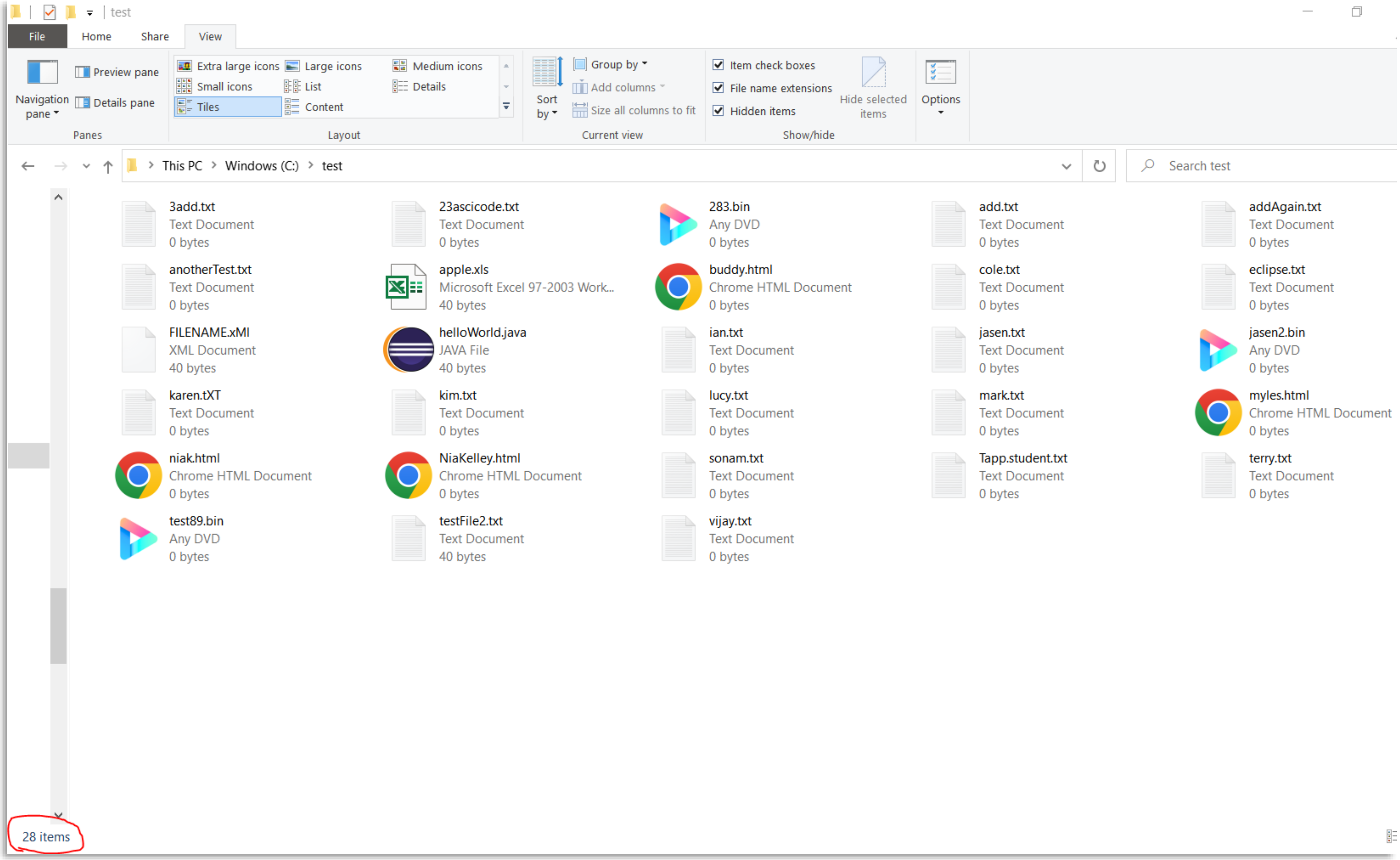
vijay.txt
Text Document
0 bytes

29 items

File Directory Structure

After

The directory has 28 files



3add.txt
Text Document
0 bytes

23ascicode.txt
Text Document
0 bytes

283.bin
Any DVD
0 bytes

add.txt
Text Document
0 bytes

addAgain.txt
Text Document
0 bytes

anotherTest.txt
Text Document
0 bytes

apple.xls
Microsoft Excel 97-2003 Work...
40 bytes

buddy.html
Chrome HTML Document
0 bytes

cole.txt
Text Document
0 bytes

eclipse.txt
Text Document
0 bytes

FILENAME.xMI
XML Document
40 bytes

helloWorld.java
JAVA File
40 bytes

ian.txt
Text Document
0 bytes

jasen.txt
Text Document
0 bytes

jasen2.bin
Any DVD
0 bytes

karen.tXT
Text Document
0 bytes

kim.txt
Text Document
0 bytes

lucy.txt
Text Document
0 bytes

mark.txt
Text Document
0 bytes

myles.html
Chrome HTML Document
0 bytes

niak.html
Chrome HTML Document
0 bytes

NiaKelley.html
Chrome HTML Document
0 bytes

sonam.txt
Text Document
0 bytes

Tapp.student.txt
Text Document
0 bytes

terry.txt
Text Document
0 bytes

test89.bin
Any DVD
0 bytes

testFile2.txt
Text Document
40 bytes

vijay.txt
Text Document
0 bytes

28 items

Test Case Scenario: Delete File – File Found – Do not proceed with Deletion

Expected Result: User inputs a file name for deletion. The application conducts a case sensitive search to determine if the file name already exists. If the file name already exists, the applicate will display a message and will not delete the file. If the file name does not exist, the application asks the User to confirm the deletion. The application asks the User to confirm the deletion. Upon indicating “N” or “n”, the application does not delete the file from the directory and displays an appropriate message on the console.

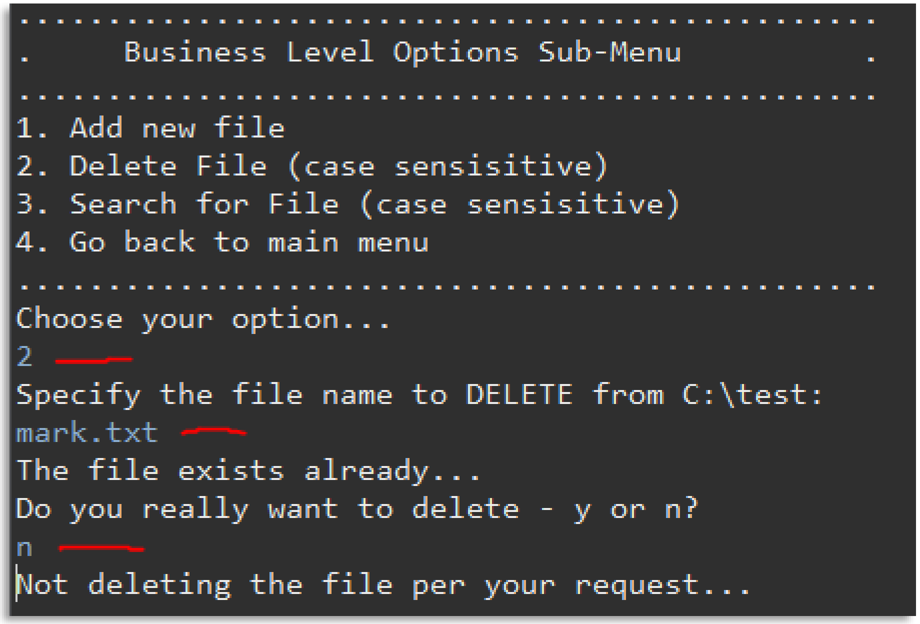
Outcome: Pass

Test Case Input:

- 2
- mark.txt
- n

Test Case Output:

Program Console

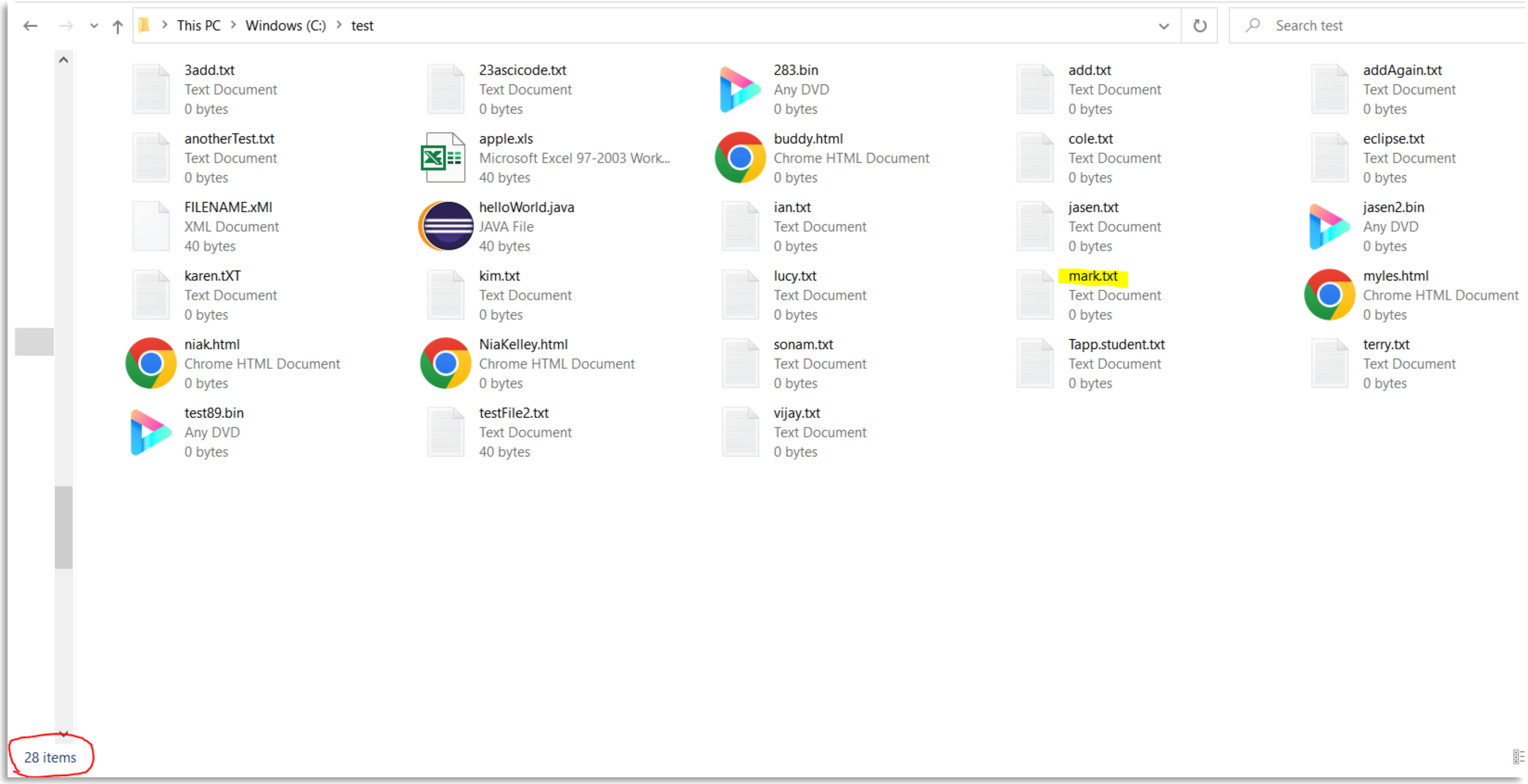


```
...............................................
.     Business Level Options Sub-Menu         .
...............................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
...............................................
Choose your option...
2
Specify the file name to DELETE from C:\test:
mark.txt
The file exists already...
Do you really want to delete - y or n?
n
Not deleting the file per your request...
```

File Directory Structure

Before

The directory has 28 files

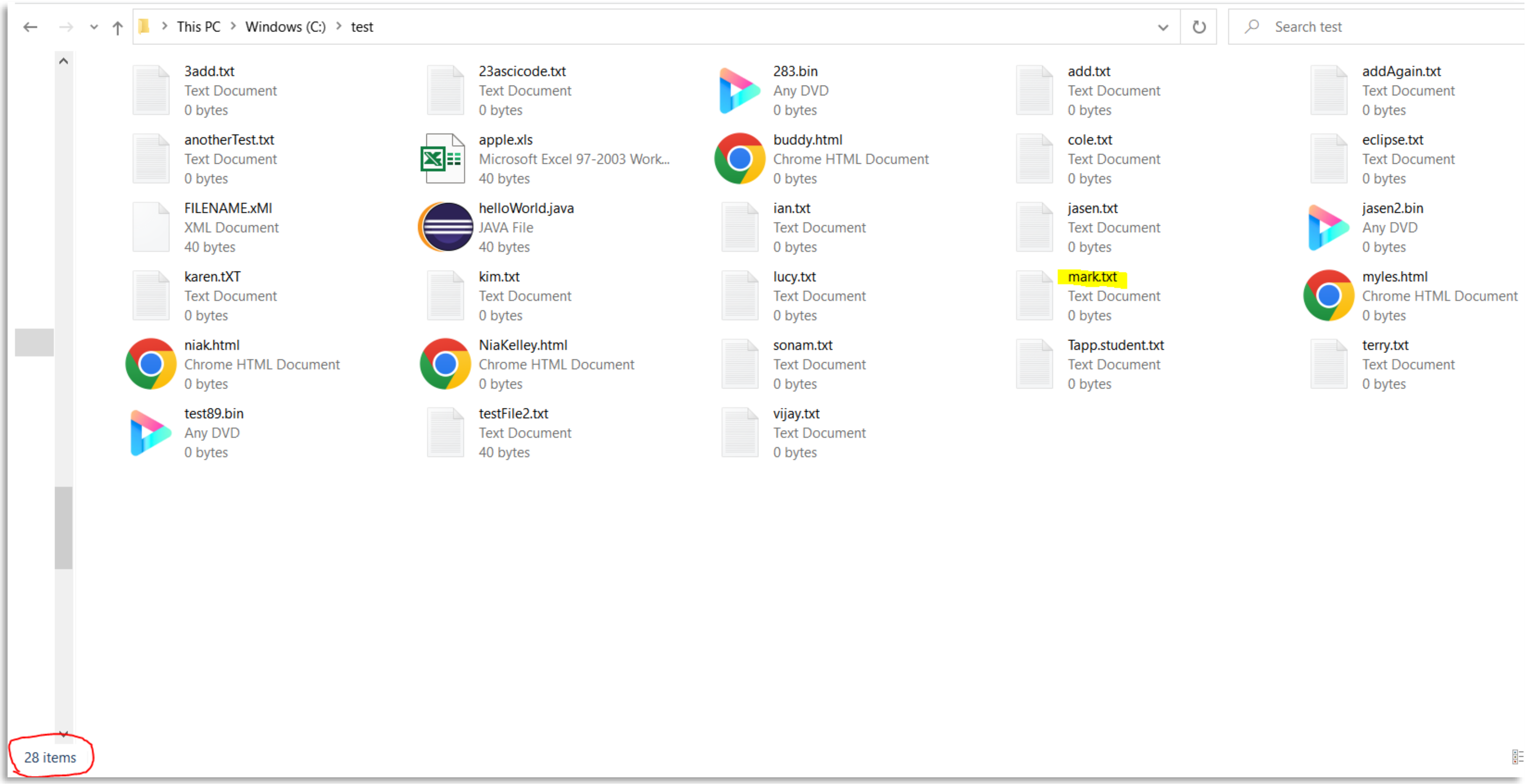


3add.txt Text Document 0 bytes	23ascicode.txt Text Document 0 bytes	283.bin Any DVD 0 bytes	add.txt Text Document 0 bytes	addAgain.txt Text Document 0 bytes
anotherTest.txt Text Document 0 bytes	apple.xls Microsoft Excel 97-2003 Work... 40 bytes	buddy.html Chrome HTML Document 0 bytes	cole.txt Text Document 0 bytes	eclipse.txt Text Document 0 bytes
FILENAME.xMI XML Document 40 bytes	helloWorld.java JAVA File 40 bytes	ian.txt Text Document 0 bytes	jasen.txt Text Document 0 bytes	jasen2.bin Any DVD 0 bytes
karen.tXT Text Document 0 bytes	kim.txt Text Document 0 bytes	lucy.txt Text Document 0 bytes	mark.txt Text Document 0 bytes	myles.html Chrome HTML Document 0 bytes
niak.html Chrome HTML Document 0 bytes	NiaKelley.html Chrome HTML Document 0 bytes	sonam.txt Text Document 0 bytes	Tapp.student.txt Text Document 0 bytes	terry.txt Text Document 0 bytes
test89.bin Any DVD 0 bytes	testFile2.txt Text Document 40 bytes	vijay.txt Text Document 0 bytes		

28 items

After

The directory has 28 files



This PC > Windows (C:) > test

Search test

3add.txt
Text Document
0 bytes

23ascicode.txt
Text Document
0 bytes

283.bin
Any DVD
0 bytes

add.txt
Text Document
0 bytes

addAgain.txt
Text Document
0 bytes

anotherTest.txt
Text Document
0 bytes

apple.xls
Microsoft Excel 97-2003 Work...
40 bytes

buddy.html
Chrome HTML Document
0 bytes

cole.txt
Text Document
0 bytes

eclipse.txt
Text Document
0 bytes

FILENAME.xMl
XML Document
40 bytes

helloWorld.java
JAVA File
40 bytes

ian.txt
Text Document
0 bytes

jasen.txt
Text Document
0 bytes

jasen2.bin
Any DVD
0 bytes

karen.tXT
Text Document
0 bytes

kim.txt
Text Document
0 bytes

lucy.txt
Text Document
0 bytes

mark.txt
Text Document
0 bytes

myles.html
Chrome HTML Document
0 bytes

niak.html
Chrome HTML Document
0 bytes

NiaKelley.html
Chrome HTML Document
0 bytes

sonam.txt
Text Document
0 bytes

Tapp.student.txt
Text Document
0 bytes

terry.txt
Text Document
0 bytes

test89.bin
Any DVD
0 bytes

testFile2.txt
Text Document
40 bytes

vijay.txt
Text Document
0 bytes

28 items

Test Case Scenario: Search for File

Expected Result: The User is prompted for a file name to search for. The application will conduct a case sensitive search on the directory, and a resultant message will be displayed on the console.

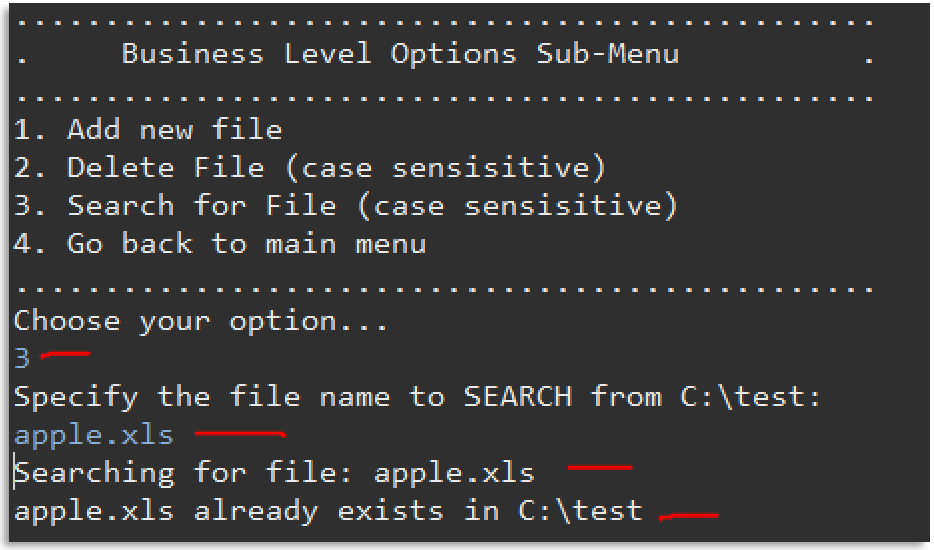
Outcome: **Pass**

Test Case Input:

- 3
- apple.xls

Test Case Output:

Program Console

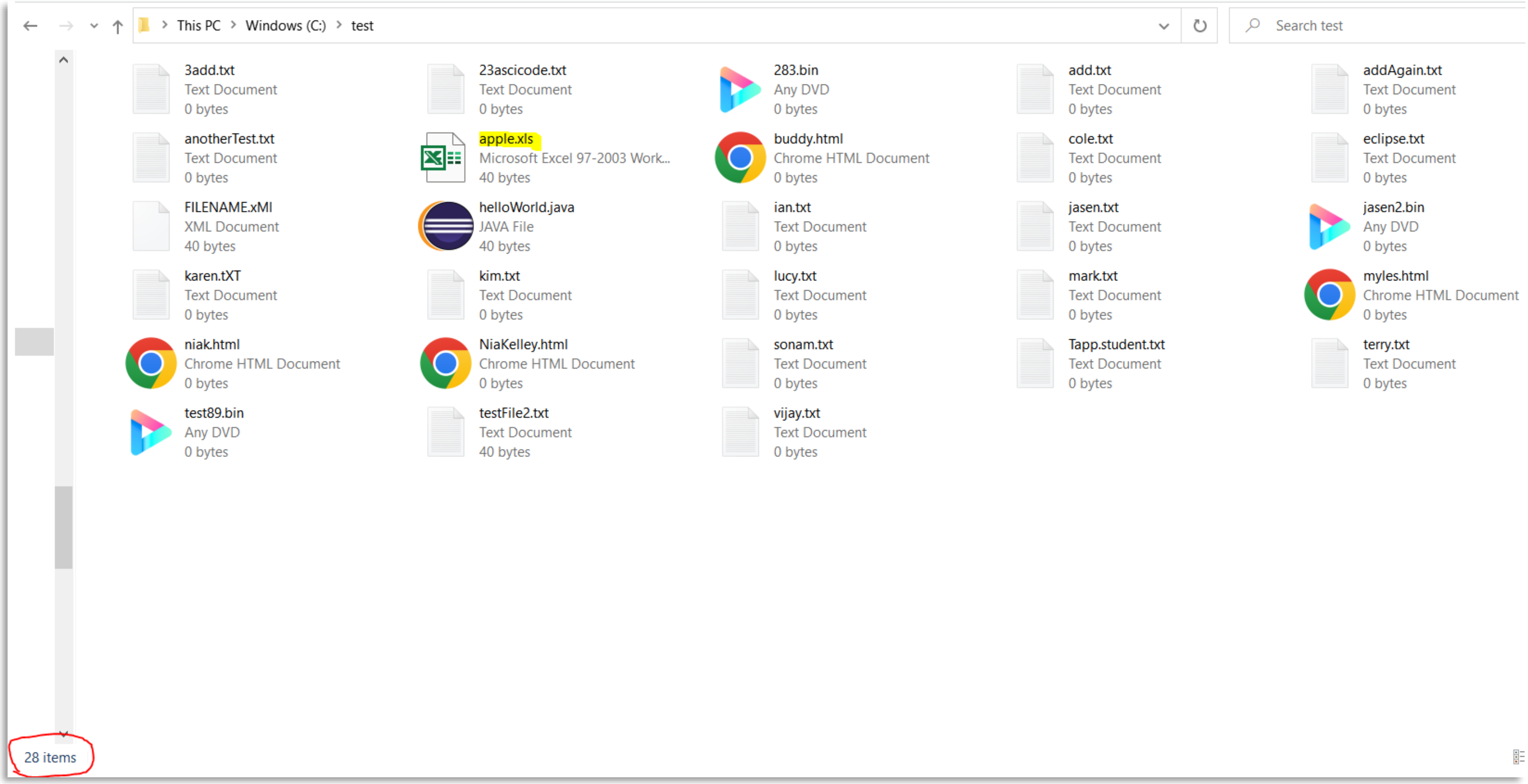


```
................................................
.     Business Level Options Sub-Menu          .
................................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
................................................
Choose your option...
3
Specify the file name to SEARCH from C:\test:
apple.xls
Searching for file: apple.xls
apple.xls already exists in C:\test
```

File Directory Structure

Before

The directory has 28 files



This PC > Windows (C:) > test

Search test

3add.txt
Text Document
0 bytes

23ascicode.txt
Text Document
0 bytes

283.bin
Any DVD
0 bytes

add.txt
Text Document
0 bytes

addAgain.txt
Text Document
0 bytes

anotherTest.txt
Text Document
0 bytes

apple.xls
Microsoft Excel 97-2003 Work...
40 bytes

buddy.html
Chrome HTML Document
0 bytes

cole.txt
Text Document
0 bytes

eclipse.txt
Text Document
0 bytes

FILENAME.xMl
XML Document
40 bytes

helloWorld.java
JAVA File
40 bytes

ian.txt
Text Document
0 bytes

jasen.txt
Text Document
0 bytes

jasen2.bin
Any DVD
0 bytes

karen.tXT
Text Document
0 bytes

kim.txt
Text Document
0 bytes

lucy.txt
Text Document
0 bytes

mark.txt
Text Document
0 bytes

myles.html
Chrome HTML Document
0 bytes

niak.html
Chrome HTML Document
0 bytes

NiaKelley.html
Chrome HTML Document
0 bytes

sonam.txt
Text Document
0 bytes

Tapp.student.txt
Text Document
0 bytes

terry.txt
Text Document
0 bytes

test89.bin
Any DVD
0 bytes

testFile2.txt
Text Document
40 bytes

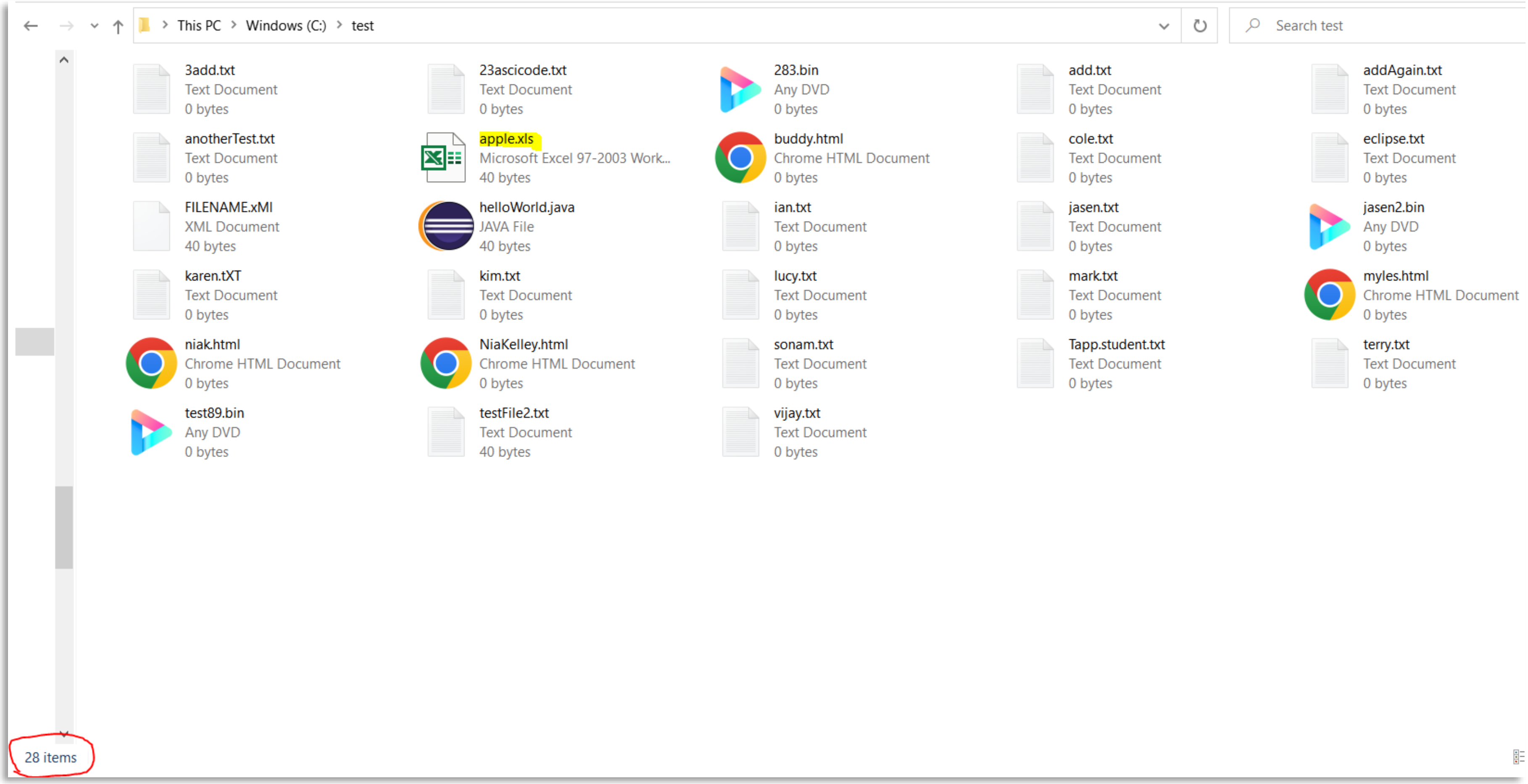
vijay.txt
Text Document
0 bytes

28 items

File Directory Structure

After

The directory has 28 files



3add.txt	Text Document	0 bytes
23ascicode.txt	Text Document	0 bytes
283.bin	Any DVD	0 bytes
add.txt	Text Document	0 bytes
addAgain.txt	Text Document	0 bytes
anotherTest.txt	Text Document	0 bytes
apple.xls	Microsoft Excel 97-2003 Work...	40 bytes
buddy.html	Chrome HTML Document	0 bytes
cole.txt	Text Document	0 bytes
eclipse.txt	Text Document	0 bytes
FILENAME.xMl	XML Document	40 bytes
helloWorld.java	JAVA File	40 bytes
ian.txt	Text Document	0 bytes
jasen.txt	Text Document	0 bytes
jasen2.bin	Any DVD	0 bytes
karen.tXT	Text Document	0 bytes
kim.txt	Text Document	0 bytes
lucy.txt	Text Document	0 bytes
mark.txt	Text Document	0 bytes
myles.html	Chrome HTML Document	0 bytes
niak.html	Chrome HTML Document	0 bytes
NiaKelley.html	Chrome HTML Document	0 bytes
sonam.txt	Text Document	0 bytes
Tapp.student.txt	Text Document	0 bytes
terry.txt	Text Document	0 bytes
test89.bin	Any DVD	0 bytes
testFile2.txt	Text Document	40 bytes
vijay.txt	Text Document	0 bytes

28 items

Test Case Scenario: Search for File

Expected Result: The User is prompted for a file name to search for. The application will conduct a case sensitive search on the directory, and a resultant message will be displayed on the console.

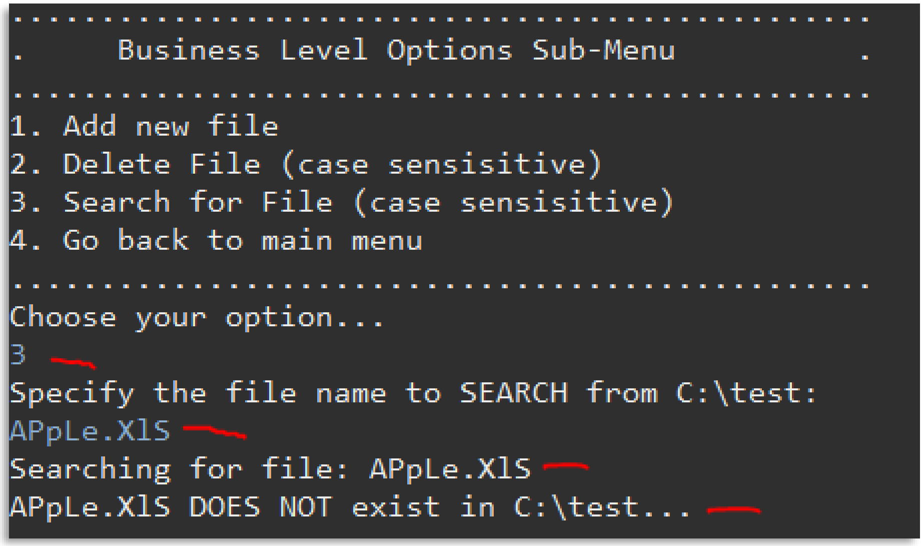
Outcome: Pass

Test Case Input:

- 3
- APpLe.XlS

Test Case Output:

Program Console

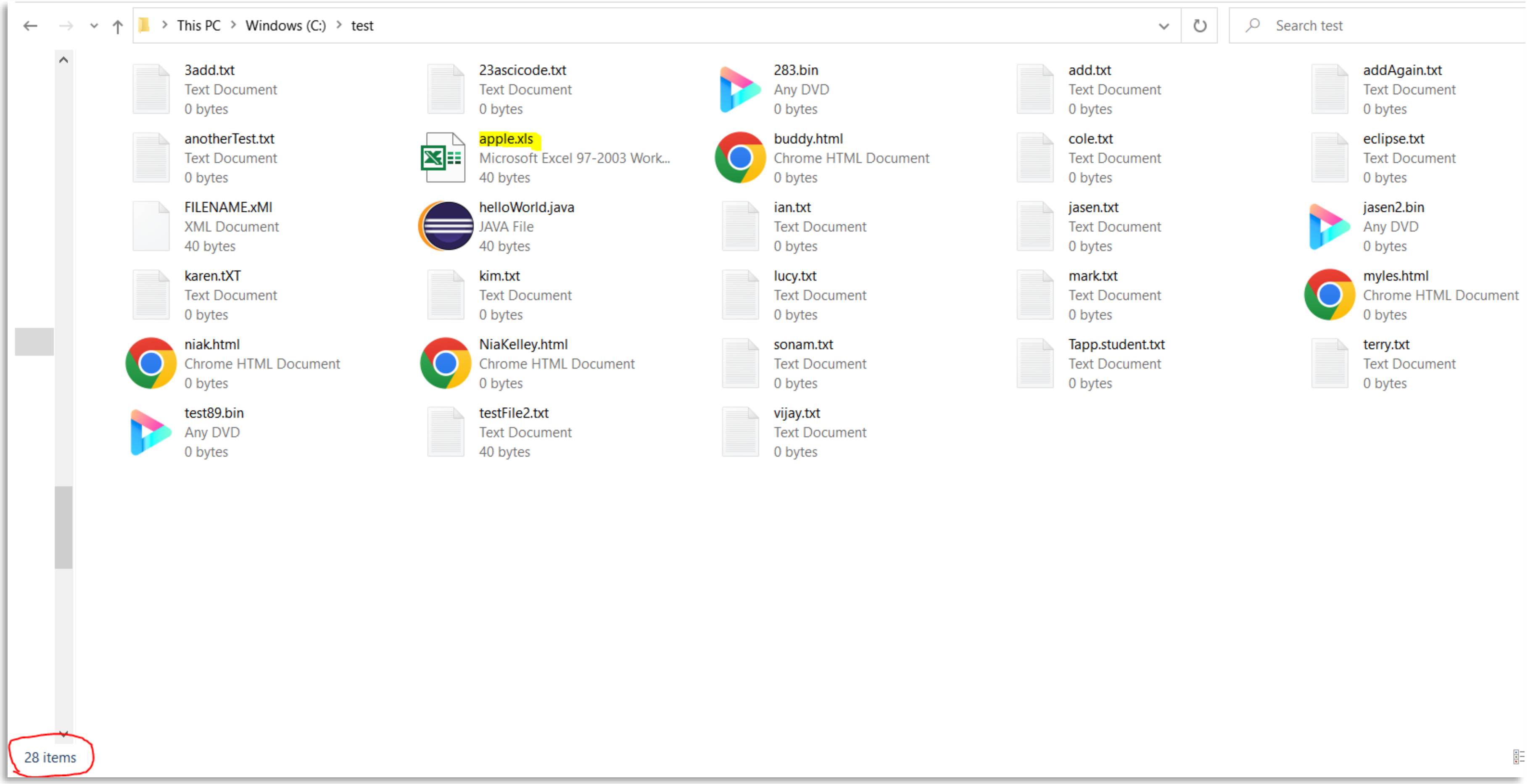


```
................................................
.     Business Level Options Sub-Menu          .
................................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
................................................
Choose your option...
3
Specify the file name to SEARCH from C:\test:
APpLe.XlS
Searching for file: APpLe.XlS
APpLe.XlS DOES NOT exist in C:\test...
```

File Directory Structure

Before

The directory has 28 files



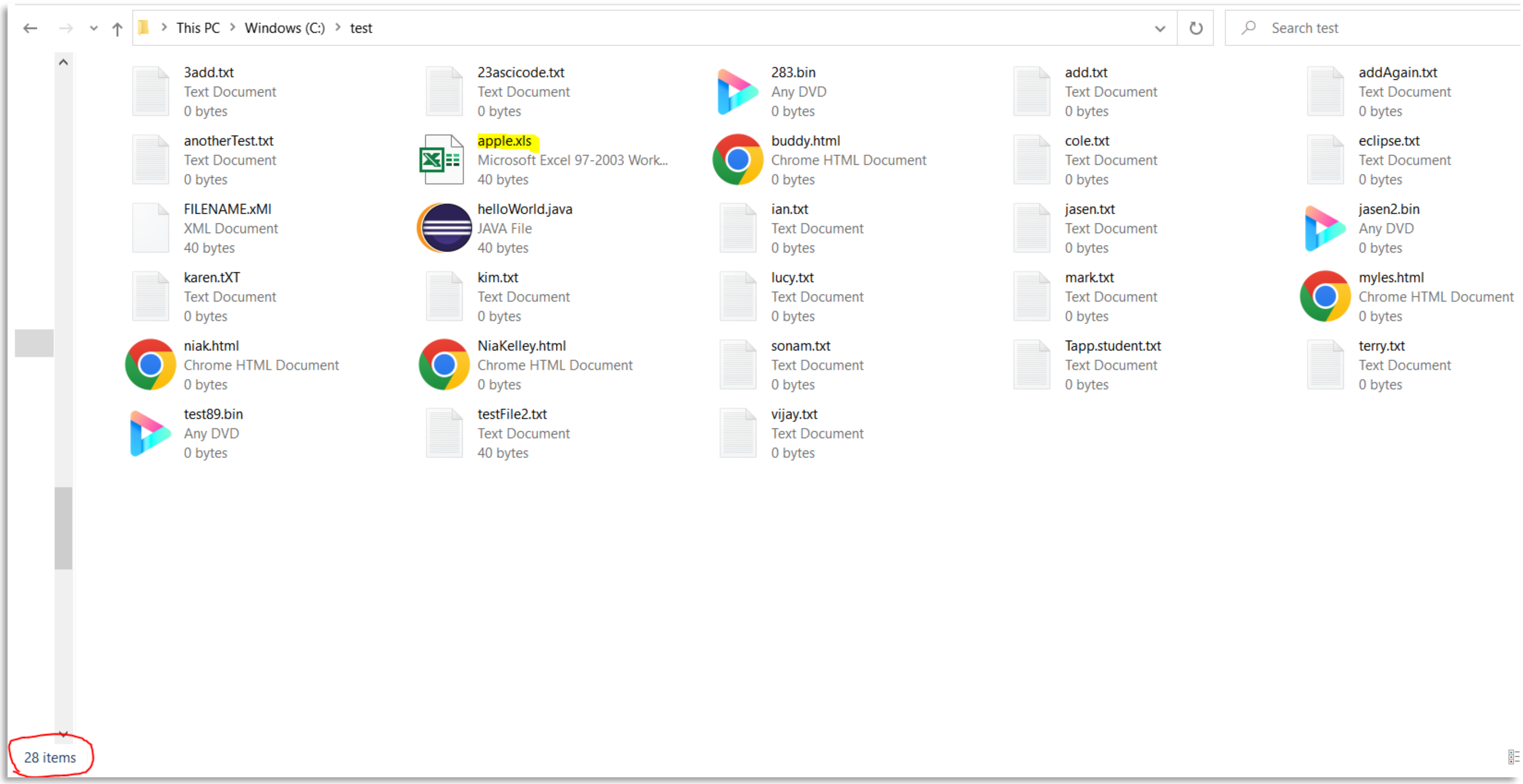
3add.txt	Text Document	0 bytes
23ascicode.txt	Text Document	0 bytes
283.bin	Any DVD	0 bytes
add.txt	Text Document	0 bytes
addAgain.txt	Text Document	0 bytes
anotherTest.txt	Text Document	0 bytes
apple.xls	Microsoft Excel 97-2003 Work...	40 bytes
buddy.html	Chrome HTML Document	0 bytes
cole.txt	Text Document	0 bytes
eclipse.txt	Text Document	0 bytes
FILENAME.xMl	XML Document	40 bytes
helloWorld.java	JAVA File	40 bytes
ian.txt	Text Document	0 bytes
jasen.txt	Text Document	0 bytes
jasen2.bin	Any DVD	0 bytes
karen.tXT	Text Document	0 bytes
kim.txt	Text Document	0 bytes
lucy.txt	Text Document	0 bytes
mark.txt	Text Document	0 bytes
myles.html	Chrome HTML Document	0 bytes
niak.html	Chrome HTML Document	0 bytes
NiaKelley.html	Chrome HTML Document	0 bytes
sonam.txt	Text Document	0 bytes
Tapp.student.txt	Text Document	0 bytes
terry.txt	Text Document	0 bytes
test89.bin	Any DVD	0 bytes
testFile2.txt	Text Document	40 bytes
vijay.txt	Text Document	0 bytes

28 items

After

File Directory Structure

The directory has 28 files



3add.txt Text Document 0 bytes	23ascicode.txt Text Document 0 bytes	283.bin Any DVD 0 bytes	add.txt Text Document 0 bytes	addAgain.txt Text Document 0 bytes
anotherTest.txt Text Document 0 bytes	apple.xls Microsoft Excel 97-2003 Work... 40 bytes	buddy.html Chrome HTML Document 0 bytes	cole.txt Text Document 0 bytes	eclipse.txt Text Document 0 bytes
FILENAME.xMl XML Document 40 bytes	helloWorld.java JAVA File 40 bytes	ian.txt Text Document 0 bytes	jasen.txt Text Document 0 bytes	jasen2.bin Any DVD 0 bytes
karen.tXT Text Document 0 bytes	kim.txt Text Document 0 bytes	lucy.txt Text Document 0 bytes	mark.txt Text Document 0 bytes	myles.html Chrome HTML Document 0 bytes
niak.html Chrome HTML Document 0 bytes	NiaKelley.html Chrome HTML Document 0 bytes	sonam.txt Text Document 0 bytes	Tapp.student.txt Text Document 0 bytes	terry.txt Text Document 0 bytes
test89.bin Any DVD 0 bytes	testFile2.txt Text Document 40 bytes	vijay.txt Text Document 0 bytes		

28 items

Test Case Scenario: Go Back to Main Menu from Business Level Options Sub-Menu

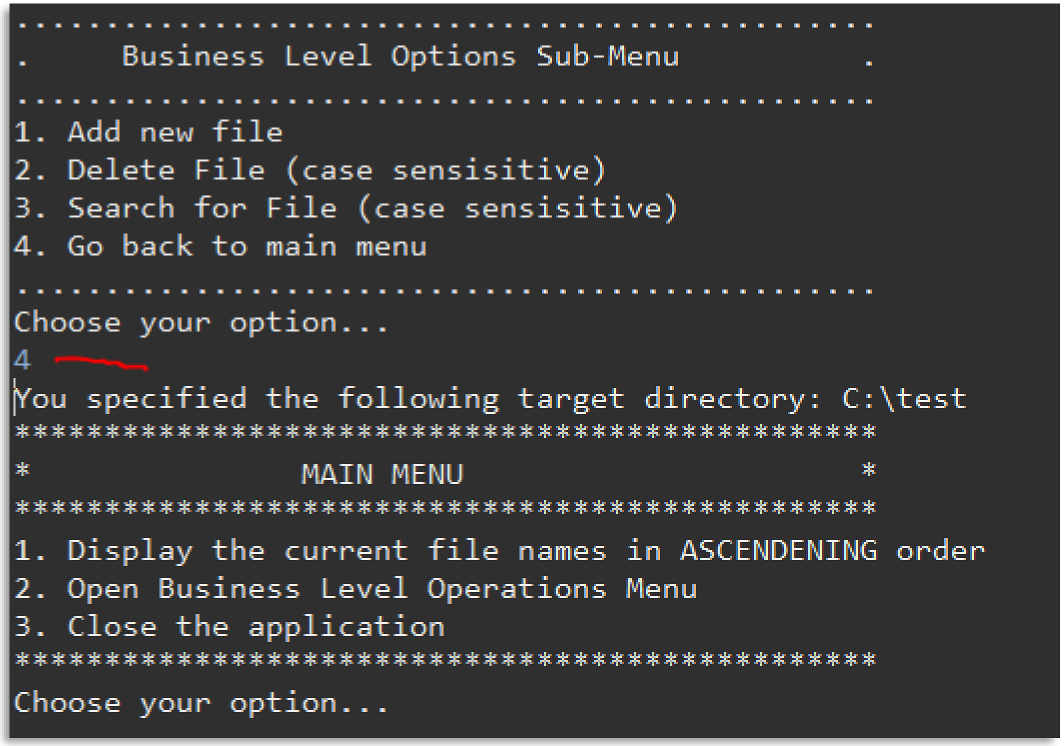
Expected Result: From the Business Level Options (sub-menu), the User can select “4 – go back to main menu” to return to the Main Menu.

Outcome: Pass

Test Case Input: 4

Test Case Output:

Program Console



```
................................................
.     Business Level Options Sub-Menu          .
................................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
................................................
Choose your option...
4
You specified the following target directory: C:\test
************************************************
*                MAIN MENU                     *
************************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
************************************************
Choose your option...
```

File Directory Structure: No need to display for this test

Test Case Scenario: Exit Program from Main Menu

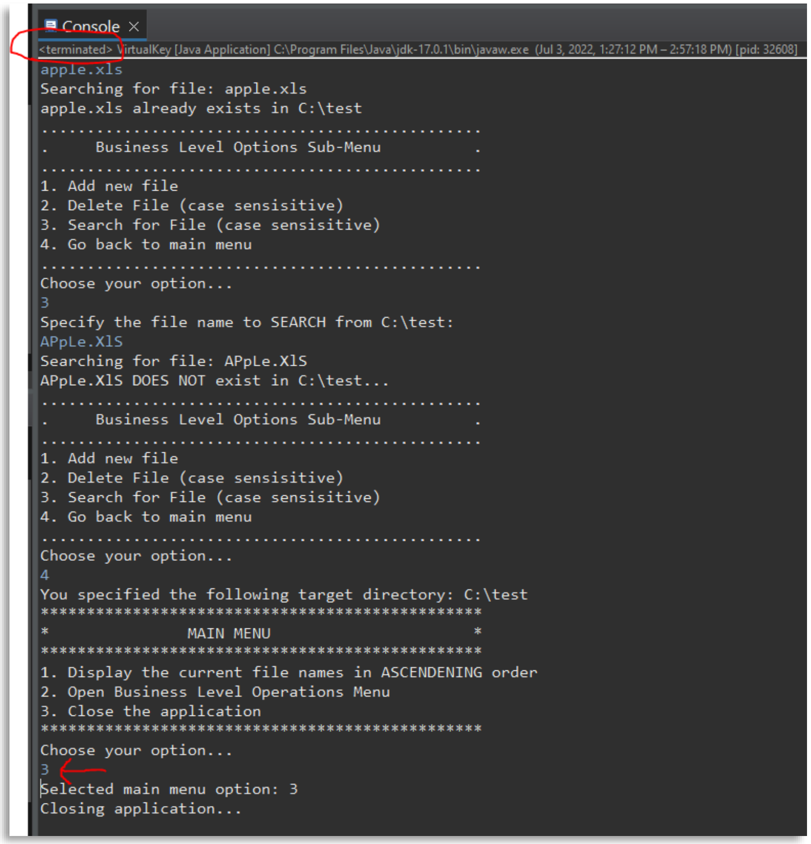
Expected Result: The User can use option “3 – Close the application” to gracefully terminate the application.

Outcome: Pass

Test Case Input: 3

Test Case Output:

Program Console



```
<terminated> VirtualKey [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Jul 3, 2022, 1:27:12 PM – 2:57:18 PM) [pid: 32608]
apple.xls
Searching for file: apple.xls
apple.xls already exists in C:\test
................................................
.     Business Level Options Sub-Menu          .
................................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
................................................
Choose your option...
3
Specify the file name to SEARCH from C:\test:
APpLe.XlS
Searching for file: APpLe.XlS
APpLe.XlS DOES NOT exist in C:\test...
................................................
.     Business Level Options Sub-Menu          .
................................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
................................................
Choose your option...
4
You specified the following target directory: C:\test
************************************************
*               MAIN MENU                      *
************************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
************************************************
Choose your option...
3
Selected main menu option: 3
Closing application...
```

File Directory Structure: No need to display for this test

Test Case Scenario: Launch Program – User-Specified Directory

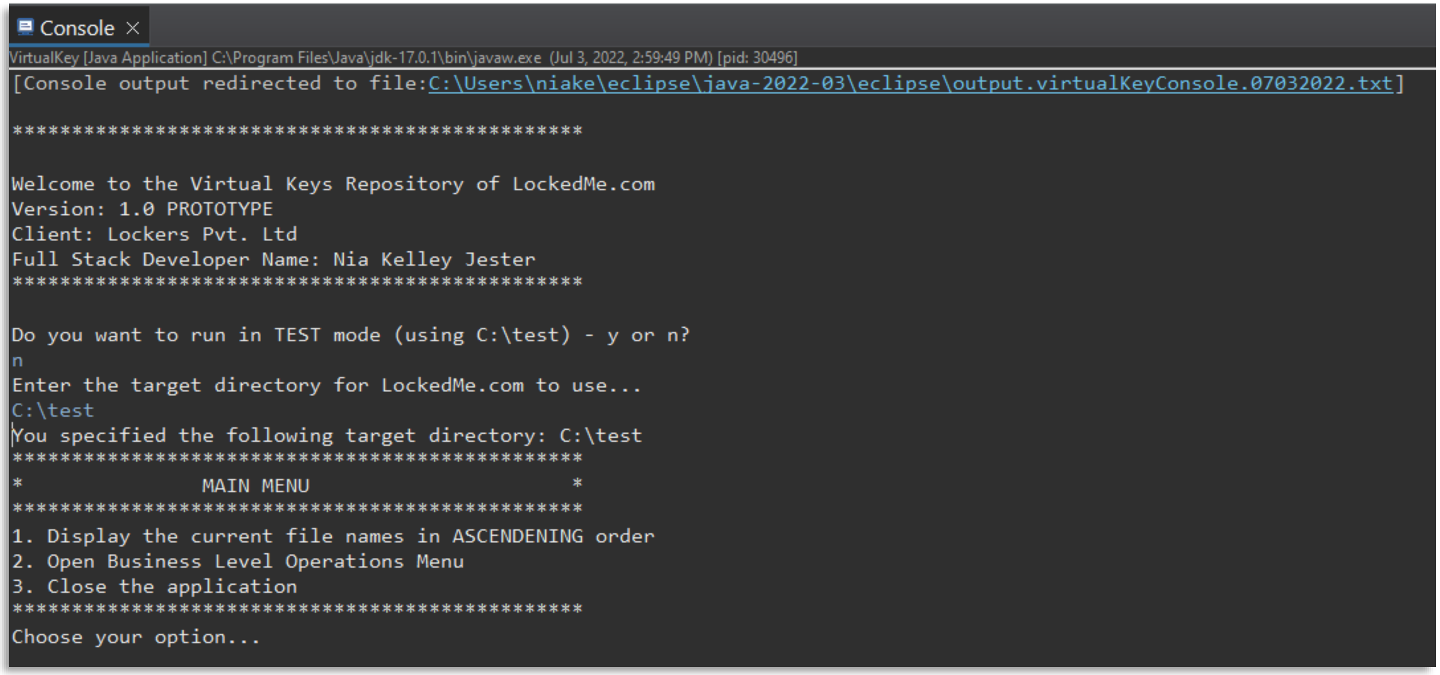
Expected Result: When the application launches, the User can specify a directory to use for the application (i.e., not running in Test mode). The User can enter “N’ or “n” when asked if you want to run the application in Test mode. the application will ask the User to enter a directory for the application. The application will display a message on the console specifying the target directory being used.

Outcome: Pass

Test Case Input:

- n
- C:\test

Test Case Output:

Program Console


```
VirtualKey [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe  (Jul 3, 2022, 2:59:49 PM) [pid: 30496]
[Console output redirected to file:C:\Users\niake\eclipse\java-2022-03\eclipse\output.virtualKeyConsole.07032022.txt]

********************************************


Welcome to the Virtual Keys Repository of LockedMe.com
Version: 1.0 PROTOTYPE
Client: Lockers Pvt. Ltd
Full Stack Developer Name: Nia Kelley Jester
********************************************


Do you want to run in TEST mode (using C:\test) - y or n?
n
Enter the target directory for LockedMe.com to use...
C:\test
You specified the following target directory: C:\test
********************************************
*             MAIN MENU                    *
********************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
********************************************
Choose your option...
```

File Directory Structure: No need to display for this test

Functional Test Cases - Exception Handling

Test Case Scenario: Launch Program – User-Specified Directory – Non-valid Input

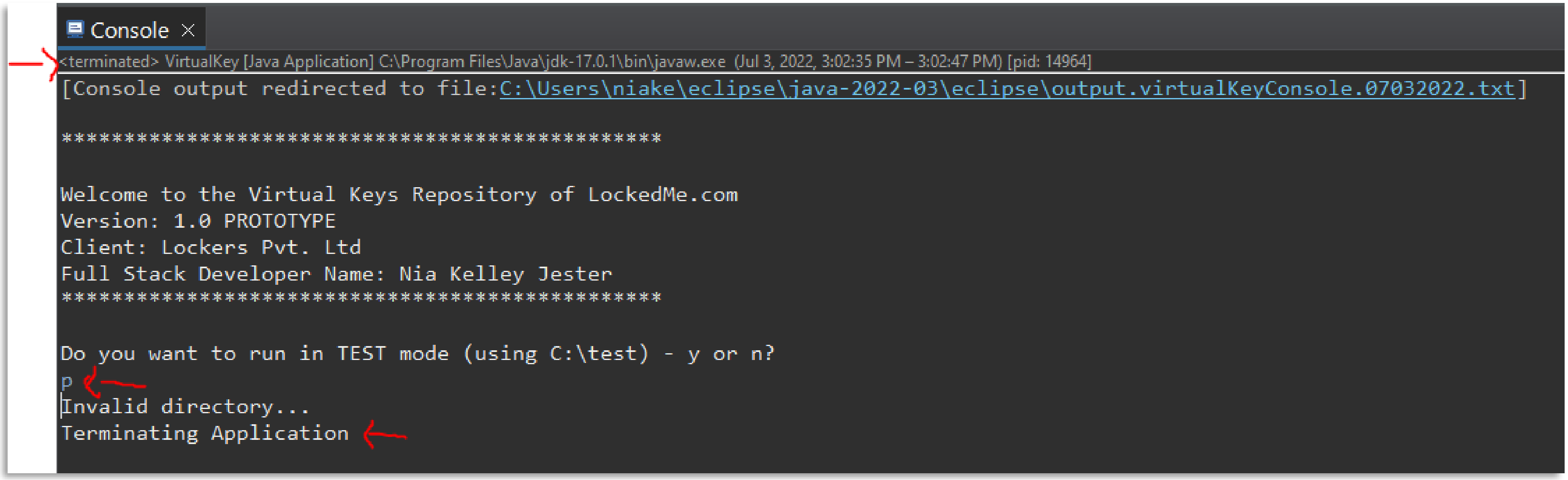
Expected Result: When the application launches, the User can specify a directory to use for the application (i.e., not running in Test mode). The User enters an invalid question, and the application will terminate.

Outcome: **Pass - Handled exception as designed**

Test Case Input: p

Test Case Output:

Program Console



```
<terminated> VirtualKey [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe  (Jul 3, 2022, 3:02:35 PM – 3:02:47 PM) [pid: 14964]
[Console output redirected to file:C:\Users\niake\eclipse\java-2022-03\eclipse\output.virtualKeyConsole.07032022.txt]

************************************************

Welcome to the Virtual Keys Repository of LockedMe.com
Version: 1.0 PROTOTYPE
Client: Lockers Pvt. Ltd
Full Stack Developer Name: Nia Kelley Jester
************************************************

Do you want to run in TEST mode (using C:\test) - y or n?
p
Invalid directory...
Terminating Application
```

File Directory Structure: No need to display for this test

Test Case Scenario: Non-valid Main Menu option

Expected Result: User enters an invalid data for the Main Menu option. The program will indicate that the User entered invalid data and gives the User the option to continue with program execution. The User will be presented with the option to continue or quit the application. The program will accept “C” or “c” to continue the program or “X” or “x” to terminate the application. If the User enters “C” or “c”, the application will display the main menu.

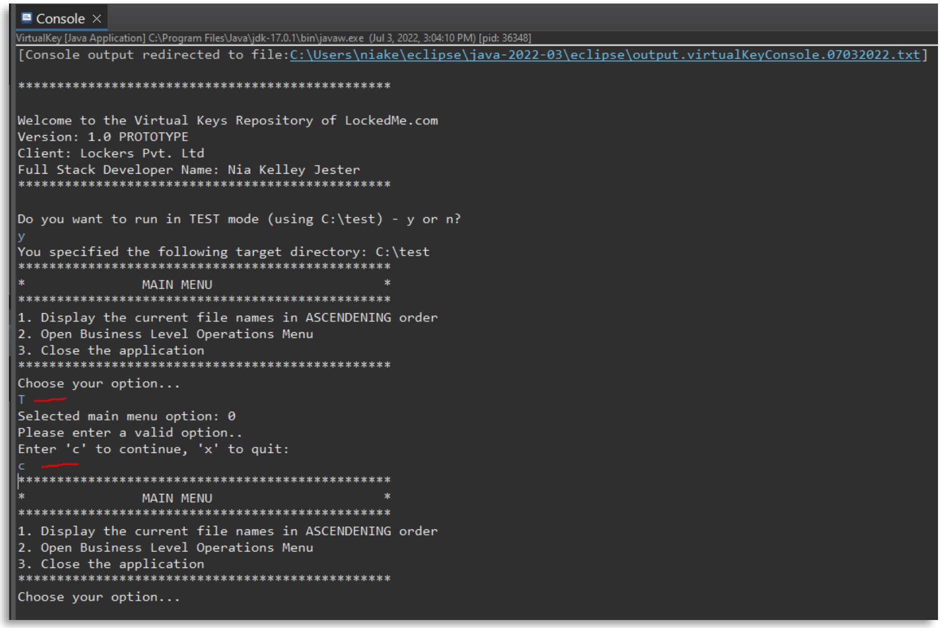
Outcome: **Pass - Handled exception as designed**

Test Case Input:

- T
- c

Test Case Output:

Program Console



```
VirtualKey [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Jul 3, 2022, 3:04:10 PM) [pid: 36348]
[Console output redirected to file:C:\Users\niake\eclipse\java-2022-03\eclipse\output.virtualKeyConsole.07032022.txt]

************************************************

Welcome to the Virtual Keys Repository of LockedMe.com
Version: 1.0 PROTOTYPE
Client: Lockers Pvt. Ltd
Full Stack Developer Name: Nia Kelley Jester
************************************************

Do you want to run in TEST mode (using C:\test) - y or n?
y
You specified the following target directory: C:\test
************************************************
*               MAIN MENU                      *
************************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
************************************************
Choose your option...
T
Selected main menu option: 0
Please enter a valid option..
Enter 'c' to continue, 'x' to quit:
c
************************************************
*               MAIN MENU                      *
************************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
************************************************
Choose your option...
```

File Directory Structure: No need to display for this test

Test Case Scenario: Non-valid Main Menu option

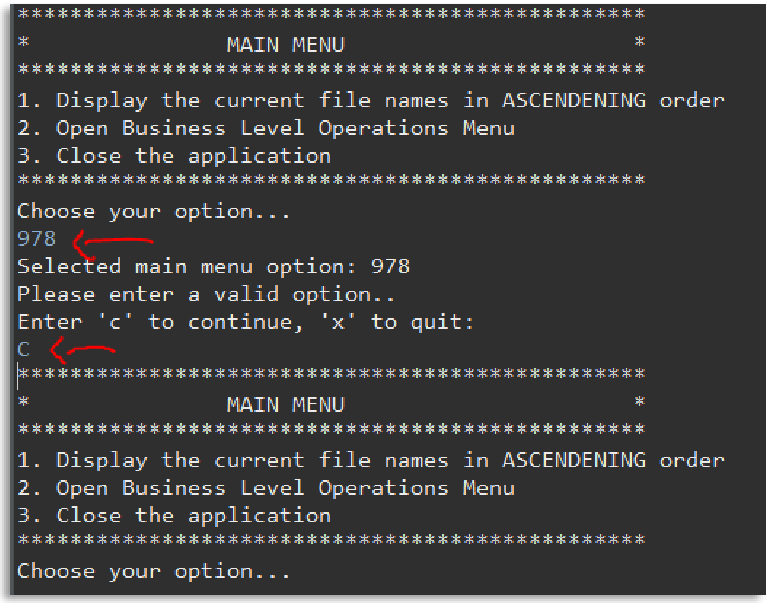
Expected Result: User enters an invalid data for the Main Menu option. The program will indicate that the User entered invalid data and gives the User the option to continue with program execution. The User will be presented with the option to continue or quit the application. The program will accept “C” or “c” to continue the program or “X” or “x” to terminate the application. If the User enters “C” or “c”, the application will display the main menu.

Outcome: **Pass - Handled exception as designed**

Test Case Input:

- 978
- C

Test Case Output:

Program Console


```
************************************************
*                 MAIN MENU                    *
************************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
************************************************
Choose your option...
978
Selected main menu option: 978
Please enter a valid option..
Enter 'c' to continue, 'x' to quit:
C
************************************************
*                 MAIN MENU                    *
************************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
************************************************
Choose your option...
```

File Directory Structure: No need to display for this test

Test Case Scenario: Non-valid Main Menu option

Expected Result: User enters an invalid data for the Main Menu option. The program will indicate that the User entered invalid data and gives the User the option to continue with program execution. The User will be presented with the option to continue or quit the application. The program will accept “C” or “c” to continue the program or “X” or “x” to terminate the application. If the User enters “C” or “c”, the application will display the main menu.

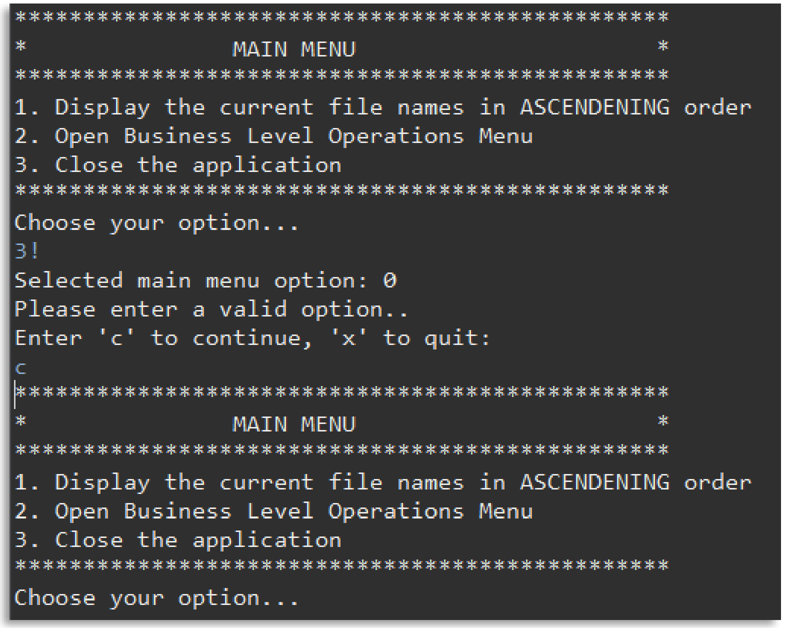
Outcome: **Pass - Handled exception as designed**

Test Case Input:

- 3!
- c

Test Case Output:

Program Console



```
************************************************
*               MAIN MENU                      *
************************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
************************************************
Choose your option...
3!
Selected main menu option: 0
Please enter a valid option..
Enter 'c' to continue, 'x' to quit:
c
************************************************
*               MAIN MENU                      *
************************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
************************************************
Choose your option...
```

File Directory Structure: No need to display for this test

Test Case Scenario: Non-valid Business Options Sub-Menu option

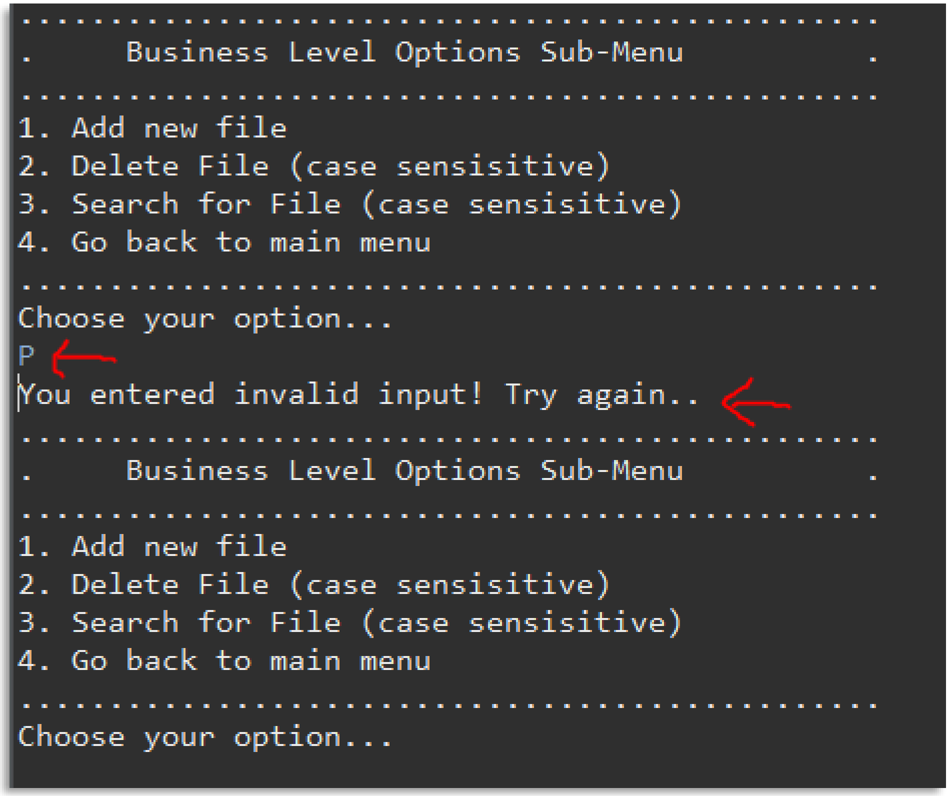
Expected Result: User enters an invalid data for the Business Level Options sub-menu. The program will indicate that the User entered invalid data, displays an error message, displays the menu again, and gives the User another opportunity to make a selection.

Outcome: **Pass - Handled exception as designed**

Test Case Input: P

Test Case Output:

Program Console



```
................................................
.     Business Level Options Sub-Menu          .
................................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
................................................
Choose your option...
P
You entered invalid input! Try again..
................................................
.     Business Level Options Sub-Menu          .
................................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
................................................
Choose your option...
```

File Directory Structure: No need to display for this test

Test Case Scenario: Non-valid Business Options Sub-Menu option

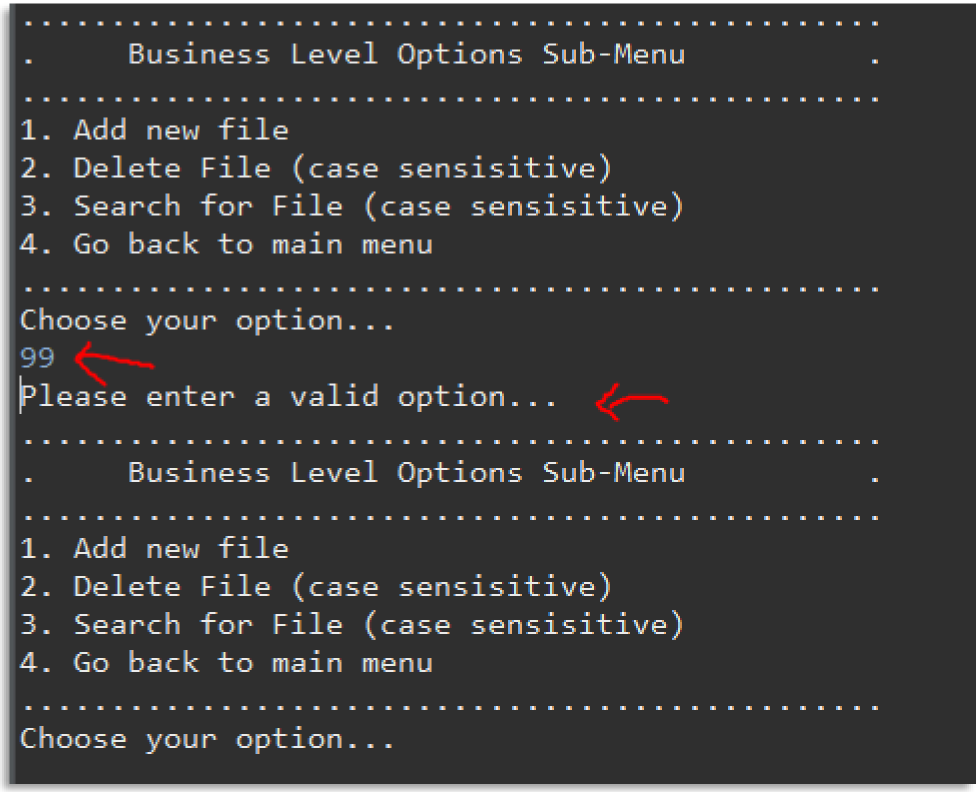
Expected Result: User enters an invalid data for the Business Level Options sub-menu. The program will indicate that the User entered invalid data, displays an error message, displays the menu again, and gives the User another opportunity to make a selection.

Outcome: **Pass - Handled exception as designed**

Test Case Input: 99

Test Case Output:

Program Console



```
...............................................
.     Business Level Options Sub-Menu         .
...............................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
...............................................
Choose your option...
99
Please enter a valid option...
...............................................
.     Business Level Options Sub-Menu         .
...............................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
...............................................
Choose your option...
```

File Directory Structure: No need to display for this test

Test Case Scenario: Non-valid Business Options Sub-Menu option

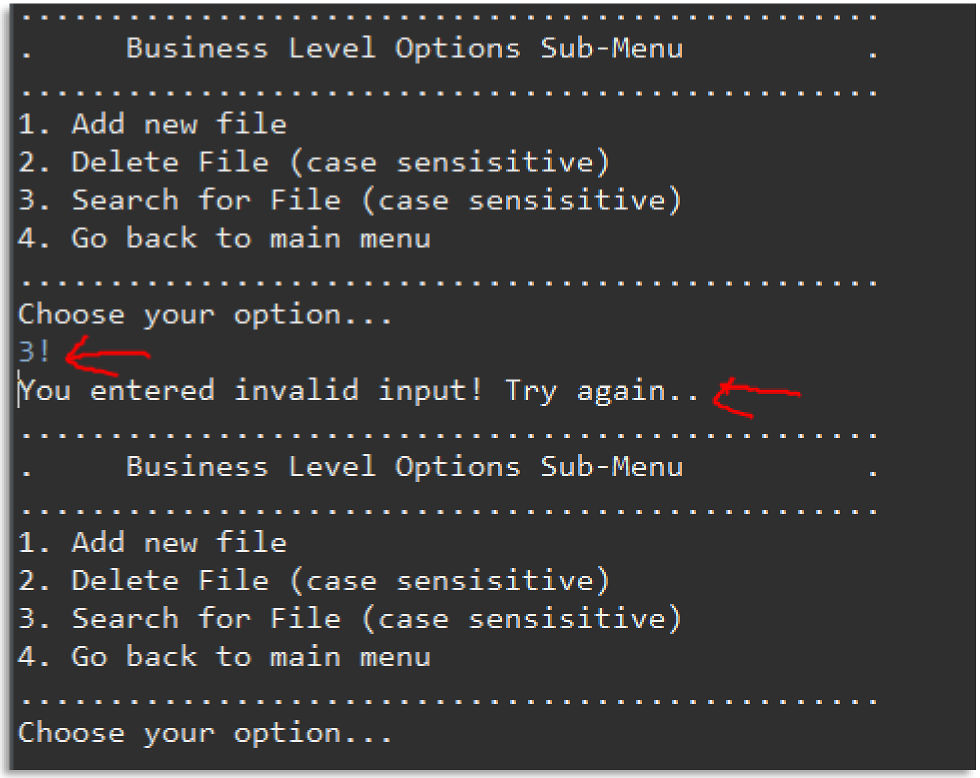
Expected Result: User enters an invalid data for the Business Level Options sub-menu. The program will indicate that the User entered invalid data, displays an error message, displays the menu again, and gives the User another opportunity to make a selection.

Outcome: **Pass - Handled exception as designed**

Test Case Input: 3!

Test Case Output:

Program Console



```
...............................................
.     Business Level Options Sub-Menu         .
...............................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu

...............................................
Choose your option...
3!
You entered invalid input! Try again..
...............................................
.     Business Level Options Sub-Menu         .
...............................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu

...............................................
Choose your option...
```

File Directory Structure: No need to display for this test

Future Improvement Areas

The current implementation does not include the following:

- The debug method `printDirectoryList()` has a noted bug, and will be fixed in a future release. The current implementation does not call this method.
- Further refine the code to use dedicated approach using `java.nio.file` package for all file I/O operations
- Formal JUnit testing
- Debugging logging

Perhaps future versions of this code will add these features later in the course.

Key Takeaways

I truly embraced the following concepts through this project:

- Static class methods
- Exception handling
- File Handling & the `java.nio.file` package (still learning)

GitHub Repository

I have pushed my code and associated documentation to the following GitHub repository:

https://github.com/niakelleyjester/simplilearn-projects/tree/main/Phase%201%20Projects/src/com/simplilearn/project/virtualkey