Project Report:

Portfolio – Home Inventory Manager

Alejandro Ricciardi

Colorado State University Global

CSC320: Programming 1

Professor Herbert Pensado

June 9, 2024

Contents

Project Report: Portfolio – Home Inventory Manager

This documentation is part of the Portfolio Assignment from CSC320: Programming 1 at Colorado State University Global. This Project Report is an overview of the program's functionality, pseudocode, and detailed testing scenarios including console output screenshots. The program is coded in Java JDK-21; and is named Portfolio (Home Inventory Manager). The program is composed of a Main class, InputValidation class, Home class, and a HomeInventory class.

1- The Assignment Direction

Portfolio Project – Home Inventory Manager Option2
Your Portfolio Project for CSC320 will consist of three components:

- Program corrections: Make the appropriate corrections to all the programming assignments submitted as Critical Thinking assignments from Modules 1-6. You will need to submit the programs along with the carefully outlined corrections needed in order for programs to run correctly.
- Lessons learned reflection: Create a 2-3-page summary that outlines the lessons learned in this Programming I course.
- Final program: Create a final program that meets the requirements outlined below.

Final Program Requirements
Create a home inventory class that will be used by a national builder to maintain inventory of available houses in the country. The following attributes should be present in your home class:

- private int square_feet
- private string address
- private string city
- private string state
- private int zip_code
- private string Model_name
- private string sale_status (sold, available, or under contract)

Your program should have appropriate methods such as:

- constructor
- add a new home
- remove a home
- update home attributes

All methods should include try..catch constructs. Except as noted, all methods should return a success or failure message (failure message defined in "catch").

1. Create an additional class to call your home class (e.g., Main or HomeInventory). Include a try..catch construct and print it to the console.
2. Call home class with parameterized constructor (e.g., "square_feet, address, city, state, zip_code, Model_name, sale_status").
    - Then, call the method to list the values. Loop through the array and print to the screen
3. Call the remove home method to clear the variables:
    - Print the return value.
4. Add a new home.
    - Print the return value.
    - Call the list method and print the new home information to the screen.
5. Update the home (change the sale status).
    - Print the return value.
    - Call the listing method and print the information to the screen.
6. Display a message asking if the user wants to print the information to a file (Y or N).
    - Use a scanner to capture the response. If "Y", print the file to a predefined location (e.g., C:\Temp\Home.txt). Note: you may want to create a method to print the information in the main class.
    - If "N", indicate that a file will not be printed.

Your final program submission materials must include your source code and screenshots of the application executing the application and the results.
Compile your Module 1-6 programs with corrections, lessons learned reflection, and final program course code and application screenshots.

⚠️ My notes:

- I got permission from Professor Pensado for the program to manipulate a file.
- The program utilizes BufferedReader and BufferedWriter classes to read and write a file, the program also uses the ArrayList class the access and modify the homes' data.
- In the HomeInventory class the functionalities of methods getHomeByAddress, removeHomeByAddress, and updateHomeByAddress are not implemented in version 1 of the program. However, they are available for future versions of the program.
- The program handles exceptions by passing them from the Home class to the HomeInventory class, and then to the Main class, where the exceptions and errors are displayed to the user.
- **For the source code please see Main.java, InputValidation. Java, Home.java, and HomeInventory.java files.**

2- Program Corrections

All my Critical Thinking assignments from Modules 1-6 are in the folder "Program Corrections." None of my assignments required corrections, and all achieved a grade of 100%. I have submitted my assignments with no modifications.

3- Lessons Learned Reflection

My lessons learned reflection assignment is in the file called “Programming I Lessons Learned and Reflection Lessons Learned and Reflection.doc.”

4- Git Repository

I use GitHub as my Distributed Version Control System (DVCS), the following is a link to my GitHub, Omegapy.
My GitHub repository that is used to store this assignment is named My-Academics-Portfolio and the link to this specific assignment is: https://github.com/Omegapy/My-Academics-Portfolio/tree/main/Programming-1-CSC320/Portfolio%20Project

5- Classes Description

The Home Class

The Home class creates a home object with various attributes such as id, square feet, address, city, state, zip code, model name, and sale status.

- The class has getters and setters' methods to access and manipulate the Home objects' data.
- It is utilized by the HomeInventory class.

The HomeInventory Class

The HomeInventory class creates a HomeInventory (Inventory) object that manages home data allowing for adding, removing, updating, and saving home data.

- The class can also create, read, write, and delete a home file text containing the home data.
- The class has getters and setters’ methods to access and manipulate the Home and Inventory objects’ data.
- It is utilized by the Main class.

The InputValidation Class

The InputValidation provides methods to validate user inputs for the Home Inventory Manager program.

- It is utilized by the Main class.

The Main Class

The Main class is used to run the Home Inventory Manager program.

- It initializes the HomeInventory object.
- Provides a menu for user interaction.
- Handles user inputs to manage the home inventory.

6- Pseudocode

a- Home Class Pseudocode

Manages a home inventory. The class includes attributes such as ID, square feet, address, city, state, zip code, model name, and sale status. It provides getter and setter methods to access and manipulate home data. It is utilized by the HomeInventory class.

```
public class Home
    Initialize variables :
        private Integer id = null
        private Integer squareFeet = null
        private Integer zipCode = null
        private String address = null
        private String city = null
        private String state = null
        private String modelName = null
        private String saleStatus = null
```

Constructor:

```
Constructor Home(id, squareFeet, address, city, state, zipCode, modelName, saleStatus)
    If id is null:
        Throw Exception "The value of the id is null"
    Else if squareFeet is null:
        Throw Exception "The value of the square feet is null"
    Else if address is null:
        Throw Exception "The value of the address is null"
    Else if city is null:
        Throw Exception "The value of the city is null"
    Else if state is null:
        Throw Exception "The value of the state is null"
    Else if zipCode is null:
        Throw Exception "The value of the zip code is null"
    Else if modelName is null:
        Throw Exception "The value of the model name is null"
    Else if saleStatus is null:
        Throw Exception "The value of the sale status is null"
    Else:
        Assign id to this.id
        Assign squareFeet to this.squareFeet
        Assign address to this.address
        Assign city to this.city
        Assign state to this.state
        Assign zipCode to this.zipCode
        Assign modelName to this.modelName
        Assign saleStatus to this.saleStatus
        Print "The Home object was created successfully!"
    End If
End Constructor
```

Getters:

Get ID Method

```
Method getId()
    If id is null:
        Throw NullPointerException "The value of the id is null!"
    Else:
        Print "The Home id: " + id + " was found!"
        Return id
    End If
End Method
```

Get Square Feet Method

```
Method getSquareFeet()
    If squareFeet is null:
        Throw NullPointerException "The value of the square feet is null!"
    Else:
        Print "The square feet were found!"
        Return squareFeet
    End If
End Method
```

Get Address Method

```
Method getAddress()
    If address is null:
        Throw NullPointerException "The value of the address is null!"
    Else:
        Print "The address was found!"
        Return address
    End If
End Method
```

Get City Method:

```
Method getCity()
    If city is null:
        Throw NullPointerException "The value of the city is null!"
    Else:
        Print "The city was found!"
        Return city
    End If
End Method
```

Get State Method

```
Method getState()
    If state is null:
        Throw NullPointerException "The value of the state is null!"
    Else:
        Print "The state was found!"
        Return state
    End If
End Method
```

Get Zip Code Method

```
Method getZipCode()
    If zipCode is null:
        Throw NullPointerException "The value of the zip code is null!"
```

```
    Else:
        Print "The zip code was found!"
        Return zipCode
    End If
End Method
```

Get Model Name Method

```
Method getModelName()
    If modelName is null:
        Throw NullPointerException "The value of the model name is null!"
    Else:
        Print "The model name was found!"
        Return modelName
    End If
End Method
```

Get Sale Status Method

```
Method getSaleStatus()
    If saleStatus is null:
        Throw NullPointerException "The value of the sale status is null!"
    Else:
        Print "The sale status was found!"
        Return saleStatus
    End If
End Method
```

Setters:

Set Square Feet Method

```
Method setSquareFeet(squareFeet)
    If squareFeet is null:
        Throw NullPointerException "The square feet value is null!"
    Else:
        Assign squareFeet to this.squareFeet
        Print "The home's square feet were set successfully!"
    End If
End Method
```

Set Address Method

```
Method setAddress(address)
    If address is null:
        Throw NullPointerException "The address value is null!"
    Else:
        Assign address to this.address
        Print "The home's address was set successfully!"
    End If
End Method
```

Set City Method

```
Method setCity(city)
    If city is null:
        Throw NullPointerException "The city value is null!"
    Else:
        Assign city to this.city
        Print "The home's city was set successfully!"
    End If
```

```
End Method
```

Set State Method

```
Method setState(state)
    If state is null:
        Throw NullPointerException "The state value is null!"
    Else:
        Assign state to this.state
        Print "The home's state was set successfully!"
    End If
End Method
```

Set Zip Code Method

```
Method setZipCode(zipCode)
    If zipCode is null:
        Throw NullPointerException "The zip code value is null!"
    Else:
        Assign zipCode to this.zipCode
        Print "The home's zip code was set successfully!"
    End If
End Method
```

Set Model Name Method

```
Method setModelName(modelName)
    If modelName is null:
        Throw NullPointerException "The model name value is null!"
    Else:
        Assign modelName to this.modelName
        Print "The home's model name was set successfully!"
    End If
End Method
```

Set Sale Status Method

```
Method setSaleStatus(saleStatus)
    If saleStatus is null:
        Throw NullPointerException "The sale status value is null!"
    Else:
        Assign saleStatus to this.saleStatus
        Print "The home's sale status was set successfully!"
    End If
End Method
```

toString Method

```
Method toString()
    Create a string representation of the Home object
    The format of the string should be:
    "\n---- Home id: " + id + " ----\n" +
    "[ square feet: " + squareFeet + ", Address: " + address +
    ", City: " + city + ", State: " + state +
    ", Zip Code: " + zipCode + ", Model Name: " + modelName +
    ", Sale Status: " + saleStatus + " ]"
    Return the created string
End Method
```

```
End Class
```

b- HomeInventory Class Pseudocode

Manages home data, allowing for adding, removing, updating, and saving home data. The class can also create, read, write, and delete a home file text containing the home data. It is utilized by the Main class.

```
public class HomeInventory
    Initialize variables:
        private static Integer numHome = 0
        private ArrayList<Home> homes
       private String filePath = null
```

Constructor:

```
Constructor HomeInventory(filePath)
    If filePath is null:
        Throw NullPointerException "File name-path is null!"
    Else:
        Assign filePath to this.filePath
        Initialize homes as a new ArrayList
        numHome = loadHomes()
        If numHome is null:
            Set numHome to 0
        Print "The HomeInventory object was created successfully!"
    End If
End Constructor
```

Getters:

Get Path File Method

```
Method getPathFile()
    If filePath is null:
        Throw NullPointerException "File name-path is null!"
    Else:
        Print "The file name-path was found!"
        Return filePath
    End If
End Method
```

Get Home by ID Method

```
Method getHomeById(id)
    For each home in homes:
        If home's ID matches id:
            Print "The home was found!"
            Return home
    End For
    Throw Exception "The provided id does not match a home!"
End Method
```

Get Home by Address Method

```
Method getHomeByAddress(address)
    For each home in homes:
        If home's address matches address:
            Print "The home address was found!"
```

```
            Return home
    End For
    Throw Exception "The provided home address was not found!"
End Method
```

Get Home List Method

```
Method getHomesList()
    If homes is null:
        Throw Exception "The homes list is null"
    Else:
        Print "The homes list was retrieved successfully!"
        Return homes
    End If
End Method
```

Setters:

Add Home Method

```
Method addHome(squareFeet, address, city, state, zipCode, modelName, saleStatus)
    Try:
        Increment numHome
        Create a new Home object with numHome, squareFeet, address, city, state, zipCode,
modelName, saleStatus
        Add home to homes list
        Print "The Home with id " + numHome + " was added successfully to the Homes arrayList."
    Catch Exception e:
        Throw Exception e.getMessage()
    End Try
End Method
```

Remove Home by ID Method

```
Method removeHomeById(id)
    Try:
        Set removed to false
        For each home in homes:
            If home's ID matches id:
                Remove home from homes list
                Set removed to true
                Break
        End For
        If removed:
            Print "The home was removed successfully!"
        Else:
            Throw Exception "The home with the provided ID was not found!"
    Catch Exception e:
        Throw Exception e.getMessage()
    End Try
End Method
```

Remove Home by Address Method

```
Method removeHomeByAddress(address)
    Try:
        Set removed to false
        For each home in homes:
            If home's address matches address:
```

```
                Remove home from homes list
                Set removed to true
                Break
        End For
        If removed:
            Print "The home was removed successfully!"
        Else:
            Throw Exception "The home with the provided address was not found!"
    Catch Exception e:
        Throw Exception e.getMessage()
    End Try
End Method
```

Update Home by ID Method

```
Method updateHomeById(id, updatedHome)
    Try:
        For i from 0 to homes.size() - 1:
            If home's ID at index i matches id:
                Update home at index i with updatedHome
                Print "The home was updated successfully!"
                Return
        End For
        Throw Exception "The home with the provided id was not found!"
    Catch Exception e:
        Throw Exception e.getMessage()
    End Try
End Method
```

Update Home by Address Method

```
Method updateHomeByAddress(address, updatedHome)
    Try:
        For i from 0 to homes.size() - 1:
            If home's address at index i matches address:
                Update home at index i with updatedHome
                Print "The home was updated successfully!"
                Return
        End For
        Throw Exception "The home with the provided address was not found!"
    Catch Exception e:
        Throw Exception e.getMessage()
    End Try
End Method
```

Methods for Saving and Loading Data

Save Homes Method

```
Method saveHomes()
    Try:
        Create BufferedWriter writer for filePath
        For each home in homes:
            Write home's data to file in CSV format
        Print "The home inventory was saved successfully!"
    Catch IOException e:
        Throw IOException "Error saving homes: " + e.getMessage()
```

```
    Finally:
        Try:
            If writer is not null:
                Close writer
        Catch IOException e:
            Throw IOException "An error occurred while closing the writer: " + e.getMessage()
    End Try
End Method
```

Load Homes Method

```
Method loadHomes()
    Try:
        Create BufferedReader reader for filePath
        Set homeIdTemp to null
        While line is read from reader:
            Split line into parts
            Try:
                Create Home object with parts
                Add home to homes list
                Set homeIdTemp to home's ID
                Print "The Home with the id: " + homeIdTemp + " was loaded into the Homes
arrayList."
            Catch Exception e:
                Print "The Home with the id: " + homeIdTemp + " could not be loaded into the
Homes arrayList."
        End While
    Catch IOException e:
        Throw IOException "Error loading homes: " + e.getMessage()
    Finally:
        Try:
            If reader is not null:
                Close reader
        Catch IOException e:
            Throw IOException "An error occurred while closing the reader: " + e.getMessage()
    End Try
    Return homeIdTemp
End Method
```

```
End Class
```

c- InputValidation Class Pseudocode

The InputValidation class provides methods to validate user inputs for the Home Inventory Manager program. It is utilized by the Main class.

```
public class InputValidation
    No Initialize variables
```

Square Feet Validation Method

```
Method squareFeet(scanner, prompt)
    While true:
        Print prompt
        Read input from scanner
        If input matches the regex for a positive integer (\d+):
            Convert input to Integer
            Return the Integer value
        Else:
            Print error message "-- Error: Invalid square feet. Please enter a positive integer."
    End While
End Method
```

Zip Code Validation Method

```
Method zipCode(scanner, prompt)
    While true:
        Print prompt
        Read input from scanner
        If input matches the regex for a 5-digit number (\d{5}):
            Convert input to Integer
            Return the Integer value
        Else:
            Print error message "-- Error: Invalid zip code. Please enter a 5-digit zip code."
    End While
End Method
```

Sale Status Validation Method

```
Method saleStatus(scanner, prompt)
    Define validStatuses as an array containing ["sold", "available", "under contract"]
    While true:
        Print prompt
        Read input from scanner and convert to lowercase
        For each status in validStatuses:
            If input equals status:
                Return input
        End For
        Print error message "-- Error: Invalid sale status. Please enter 'sold', 'available', or
'under contract'."
    End While
End Method
```

Sale yes or noValidation Method

```
Method yesOrNo(scanner, prompt)
    While true:
        Print prompt
        Read input from scanner and convert to lowercase
```

```
        If input is equal to “yes” or “no”
            Return input value
        Else:
            Print error message "Invalid input. Please try again!"
    End While
End Method
```

```
End Class
```

d- Main Class Pseudocode

The Main class is used to run the Home Inventory Manager program. It initializes the HomeInventory object, provides a menu for user interaction, and handles user inputs to manage the home inventory.

```
public class Main
```

Main Method

```
Method main(args)
    Initialize Scanner object for user input
    Declare HomeInventory object named inventory
    Set quitProgram to false
    Set isHomesChangesSaved to true
    Display the program banner

    inventory = Call createInventory(scanner)

    While not quitProgram:
        Display menu options
        Read user choice

        Switch choice:
            Case "1":
                Try:
                    Call addHome(scanner, inventory)
                    Set isHomesChangesSaved to false
                Catch Exception e:
                    Print error message
                End Try
                Break
            Case "2":
                Try:
                    Call removeHome(scanner, inventory)
                    Set isHomesChangesSaved to false
                Catch Exception e:
                    Print error message
                End Try
                Break
            Case "3":
                Try:
                    Call updateHome(scanner, inventory)
```

```
                    Set isHomesChangesSaved to false
                Catch Exception e:
                    Print error message
                End Try
                Break
            Case "4":
                Call listHomes(inventory)
                Break
            Case "5":
                Call displayHome(scanner, inventory)
                Break
            Case "6":
                Try:
                    Call saveHome(inventory)
                    Set isHomesChangesSaved to true
                Catch Exception e:
                    Print error message
                End Try
                Break
            Case "7":
                If isHomesChangesSaved is false:
                    Prompt user to save changes
                    If user chooses to save:
                        Try:
                            Call saveHome(inventory)
                        Catch Exception e:
                            Print error message
                        End Try
                    End If
                End If
                Prompt user to delete the inventory file
                If user chooses to delete:
                    Call deleteFile(inventory.getPathFile())
                End If
                Set quitProgram to true
                Break
            Default:
                Print "Invalid choice. Please try again."
        End Switch
    End While

    Close scanner
    Print "Thank you for using the Home Inventory Manager program!"
End Method
```

Create Inventory Method

```
Method createInventory(scanner)
    While true:
        Try:
            Prompt user to enter the inventory file path-name
            Read filePath from scanner

            Create a File object with filePath
            If file is a new file:
```

```
                While true:
                    Prompt user to create a file populated with fake data or an empty file
                    Read choiceFakeData from scanner
                    If choiceFakeData equals "y":
                        Create HomeInventory object with filePath
                        Add fake homes to inventory
                        Save homes to file
                        Print success message
                        Return inventory
                    Else If choiceFakeData equals "n":
                        Create HomeInventory object with filePath
                        Print success message
                        Return inventory
                    Else:
                        Print "Invalid Entry! Please try again."
                End While
            Else:
                Print "The file already exists!"
                Load homes data from file into inventory
                Return inventory
        Catch Exception e:
            Print error message
            Print "Please try again!"
        End Try
    End While
End Method
```

Data Manipulation Methods

Add Home Method

```
Method addHome(scanner, inventory)
    Try:
        Prompt user for square feet and validate input
        Prompt user for address
        Prompt user for city
        Prompt user for state
        Prompt user for zip code and validate input
        Prompt user for home model
        Prompt user for sale status and validate input

        Add home to inventory
        Print success message
        Get and display the id of the newly added home
    Catch Exception e:
        Throw new Exception("Error adding home: " + e.getMessage())
    End Try
End Method
```

Update Home Method

```
Method updateHome(scanner, inventory)
    Try:
        Prompt user for home id
        Get home to update by id
```

```
        If home is found:
            Set continueUpdating to true
            While continueUpdating:
                Display update options
                Read update choice

                Switch update choice:
                    Case "1":
                        Prompt user for new square feet and validate input
                        Set new square feet to home
                        Break
                    Case "2":
                        Prompt user for new address
                        Set new address to home
                        Break
                    Case "3":
                        Prompt user for new city
                        Set new city to home
                        Break
                    Case "4":
                        Prompt user for new state
                        Set new state to home
                        Break
                    Case "5":
                        Prompt user for new zip code and validate input
                        Set new zip code to home
                        Break
                    Case "6":
                        Prompt user for new model name
                        Set new model name to home
                        Break
                    Case "7":
                        Prompt user for new sale status and validate input
                        Set new sale status to home
                        Break
                    Case "8":
                        Set continueUpdating to false
                        Break
                    Default:
                        Print "Invalid choice. Please try again."
                End Switch
            End While

            Update home in inventory by id
            Display updated home data
        Else:
            Print "Home not found."
    Catch Exception e:
        Throw new Exception("Error updating home: " + e.getMessage())
    End Try
End Method
```

List Homes Method

```
Method listHomes(inventory)
    Try:
        Get homes list from inventory
        If homes list is empty:
            Print "No homes in inventory."
        Else:
            For each home in homes list:
                Print home
        End If
    Catch Exception e:
        Print "Error listing homes: " + e.getMessage()
        Print "Please try again."
    End Try
End Method
```

Display Data Methods

Display Home Method

```
Method displayHome(scanner, inventory)
    Try:
        Prompt user for home id
        Get home by id from inventory
        If home is found:
            Print home
        Else:
            Print "Home not found."
    Catch Exception e:
        Print "Error displaying home: " + e.getMessage()
        Print "Please try again."
    End Try
End Method
```

Display Home Using ID Method

```
Method displayHomeUsingId(id, inventory)
    Try:
        Get home by id from inventory
        If home is found:
            Print home
        Else:
            Print "Home not found."
    Catch Exception e:
        Print "Error displaying home: " + e.getMessage()
    End Try
End Method
```

File Manipulation Methods

Delete File Method

```
Method deleteFile(filePath)
    Create a File object with filePath
    Try:
        Attempt to delete the file
        Print success or failure message based on the result
```

```
    Catch SecurityException se:
        Print "Permission denied cannot delete the file: " + se.getMessage()
    Catch Exception e:
        Print "An error occurred cannot delete the file: " + e.getMessage()
    End Try
End Method
```

Save Home Method

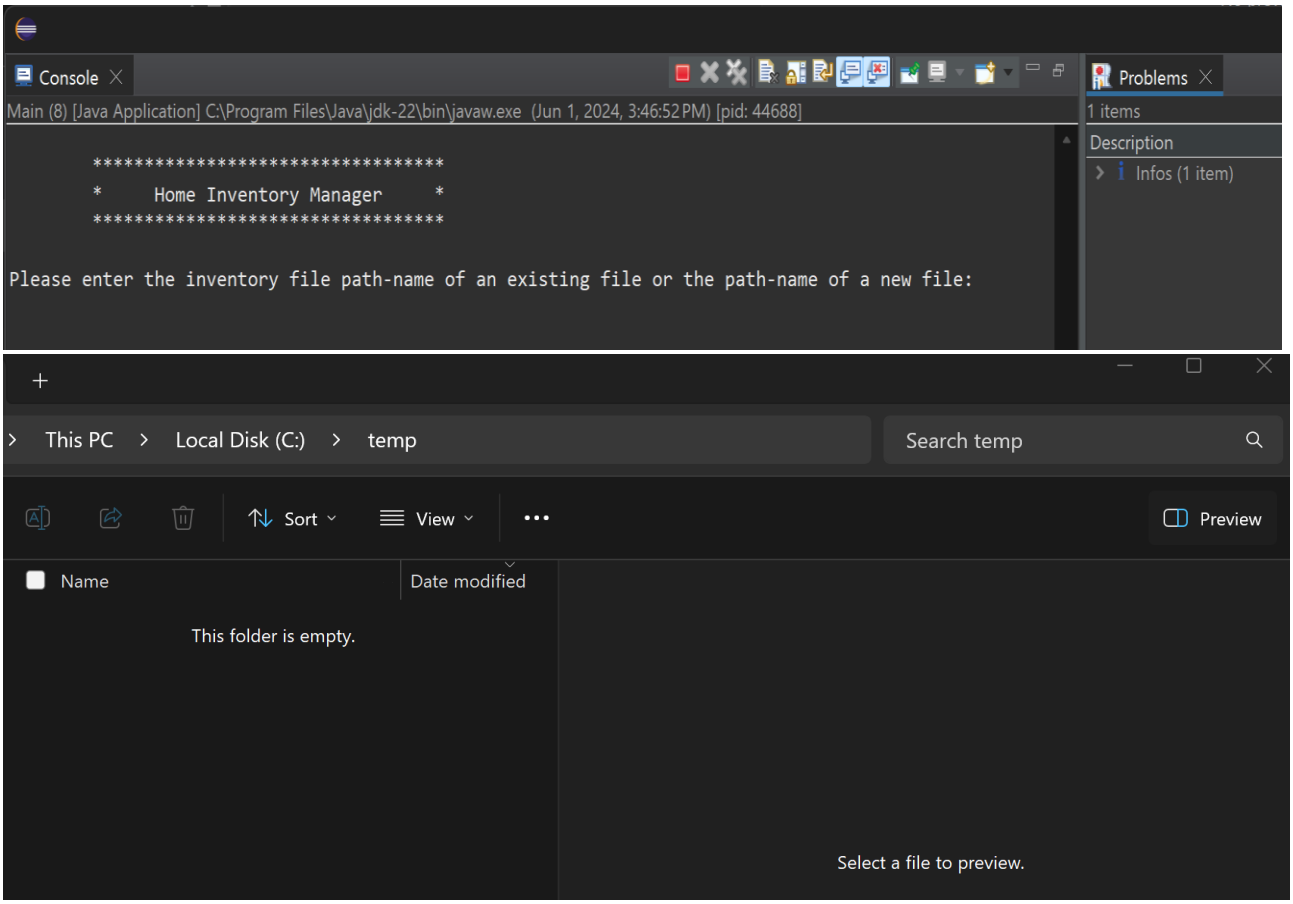
```
Method saveHome(inventory)
    Try:
        Save homes data from inventory to file
    Catch Exception e:
        Throw new Exception("Error saving homes: " + e.getMessage())
    End Try
End Method

End Class
```

7- Screenshots: Program Functionality and Testing Scenarios

a- Functionality Figures 1 To 10

Figure 1

Welcome Screen and Temp Folder

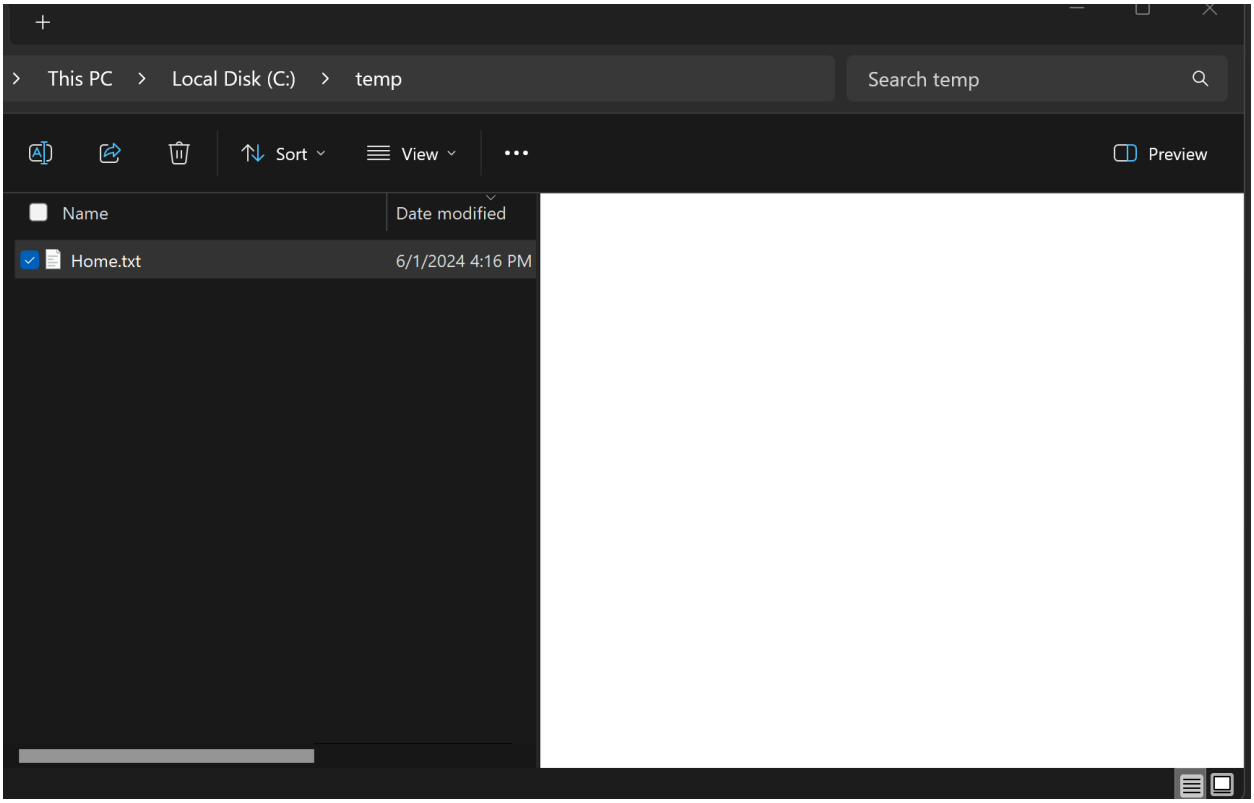


Note: Eclipse Console output from the program and Problem window showing no items.

Figure 2

Creating a New Empty File, Temp Folder, and File Preview

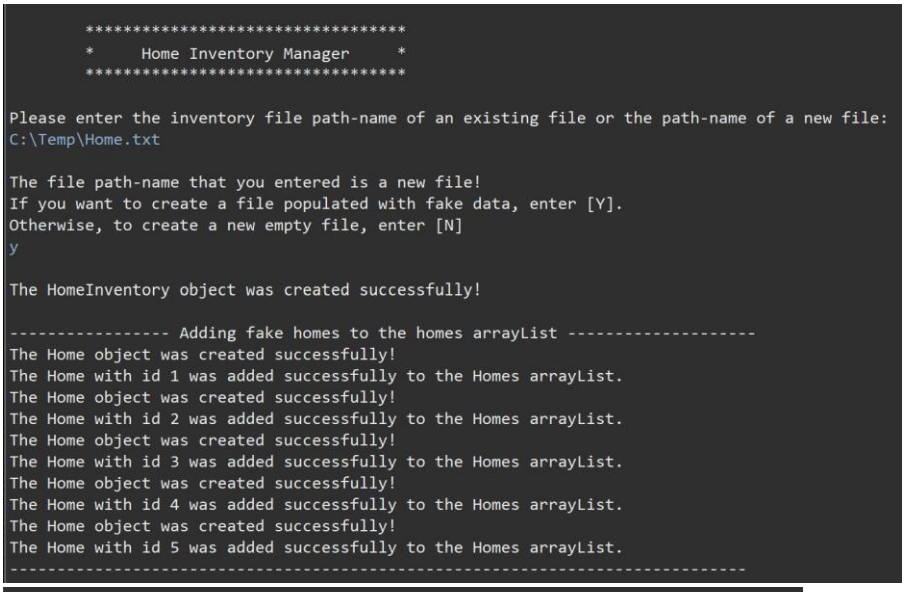
```
        **********************************
        *     Home Inventory Manager     *
        **********************************

Please enter the inventory file path-name of an existing file or the path-name of a new file:
C:\Temp\Home.txt

The file path-name that you entered is a new file!
If you want to create a file populated with fake data, enter [Y].
Otherwise, to create a new empty file, enter [N]
n

The HomeInventory object was created successfully!
The file C:\Temp\Home.txt was successfully created!

       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
```



Note: File preview is empty

Figure 3

Creating a New File Populated with Fake Data and Exiting Program

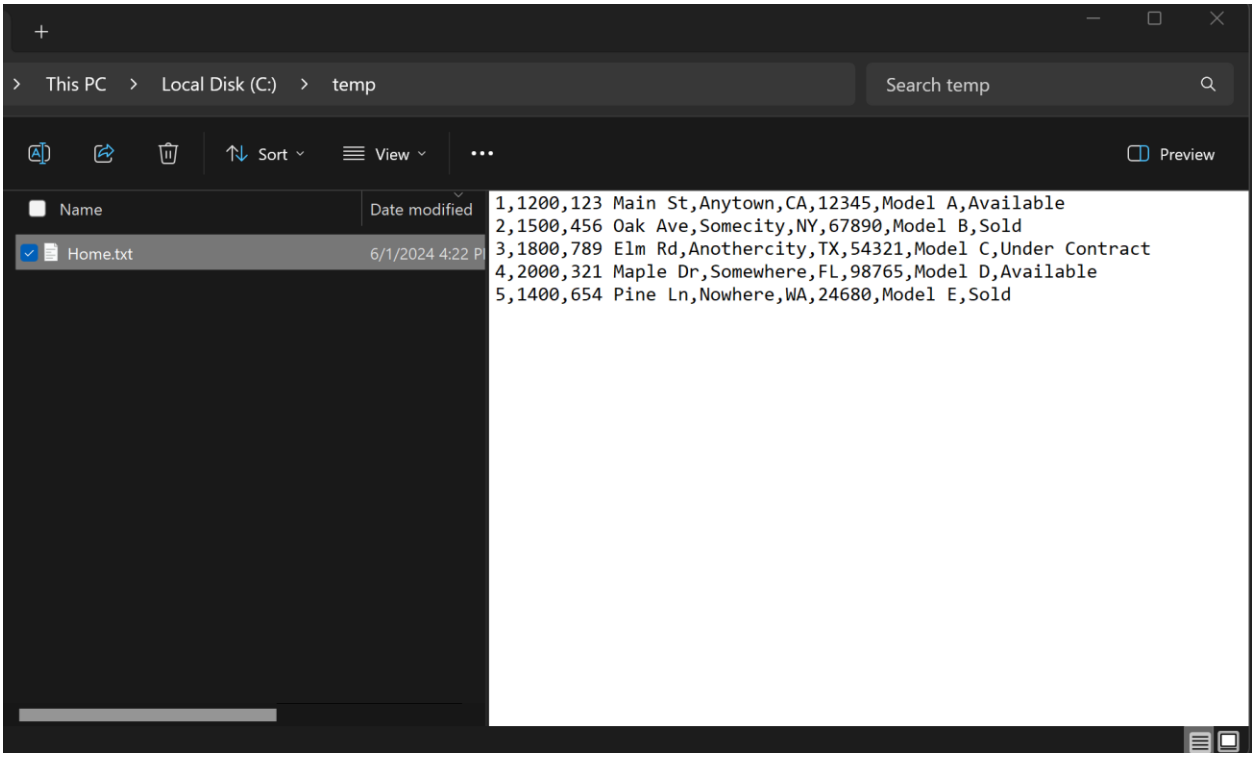


```
        *********************************
        *     Home Inventory Manager    *
        *********************************

Please enter the inventory file path-name of an existing file or the path-name of a new file:
C:\Temp\Home.txt

The file path-name that you entered is a new file!
If you want to create a file populated with fake data, enter [Y].
Otherwise, to create a new empty file, enter [N]
y

The HomeInventory object was created successfully!

----------------- Adding fake homes to the homes arrayList --------------------
The Home object was created successfully!
The Home with id 1 was added successfully to the Homes arrayList.
The Home object was created successfully!
The Home with id 2 was added successfully to the Homes arrayList.
The Home object was created successfully!
The Home with id 3 was added successfully to the Homes arrayList.
The Home object was created successfully!
The Home with id 4 was added successfully to the Homes arrayList.
The Home object was created successfully!
The Home with id 5 was added successfully to the Homes arrayList.
-------------------------------------------------------------------------------
```

```
------------------ Saving Homes arrayList to file ---------------------------
The Home id: 1 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 2 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 3 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 4 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 5 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!

The home inventory was saved successfully!
The file C:\Temp\Home.txt was successfully created and populated with fake data!
-------------------------------------------------------------------------------
```

Continue next page

```
      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
7
Do you want to delete the inventory file? [Y/N]: n

Thank you for using the Home Inventory Manager program!
```



Note: Preview data in the preview window and existing program without deleting created file.

Figure 4

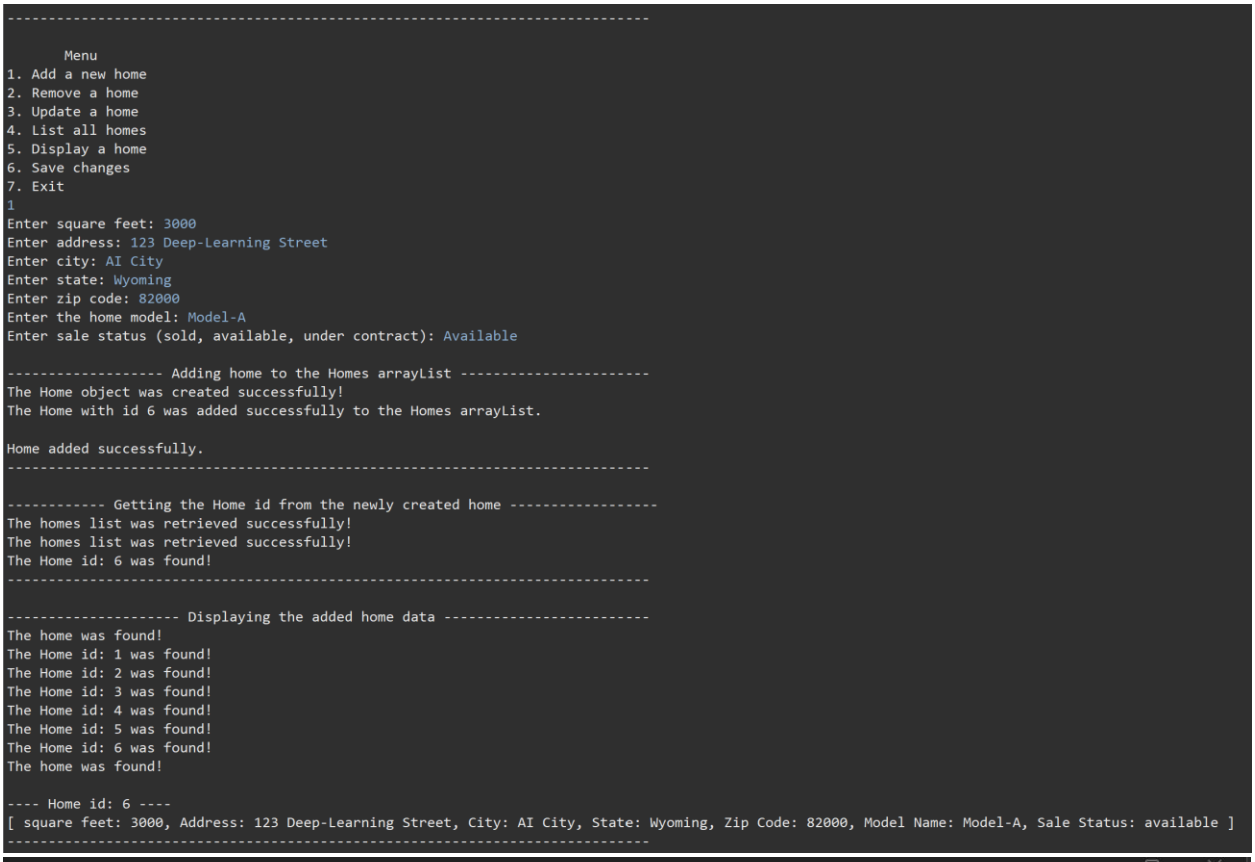
Opening an Existing File and Display the File Data

```
        **********************************
        *     Home Inventory Manager     *
        **********************************

Please enter the inventory file path-name of an existing file or the path-name of a new file:
C:\Temp\Home.txt
The file C:\Temp\Home.txt already exists!

---------------------------- Loading File Data ------------------------------
The Home object was created successfully!
The Home with the id: 1 was loaded into the Homes arrayList.
The Home object was created successfully!
The Home with the id: 2 was loaded into the Homes arrayList.
The Home object was created successfully!
The Home with the id: 3 was loaded into the Homes arrayList.
The Home object was created successfully!
The Home with the id: 4 was loaded into the Homes arrayList.
The Home object was created successfully!
The Home with the id: 5 was loaded into the Homes arrayList.

The HomeInventory object was created successfully!
--------------------------------------------------------------------------------
```

```
      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
4

-------------------- Displaying the list of homes -------------------------
The homes list was retrieved successfully!

---- Home id: 1 ----
[ square feet: 1200, Address: 123 Main St, City: Anytown, State: CA, Zip Code: 12345, Model Name: Model A, Sale Status: Available ]

---- Home id: 2 ----
[ square feet: 1500, Address: 456 Oak Ave, City: Somecity, State: NY, Zip Code: 67890, Model Name: Model B, Sale Status: Sold ]

---- Home id: 3 ----
[ square feet: 1800, Address: 789 Elm Rd, City: Anothercity, State: TX, Zip Code: 54321, Model Name: Model C, Sale Status: Under Contract ]

---- Home id: 4 ----
[ square feet: 2000, Address: 321 Maple Dr, City: Somewhere, State: FL, Zip Code: 98765, Model Name: Model D, Sale Status: Available ]

---- Home id: 5 ----
[ square feet: 1400, Address: 654 Pine Ln, City: Nowhere, State: WA, Zip Code: 24680, Model Name: Model E, Sale Status: Sold ]

--------------------------------------------------------------------------------

      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
```

Figure 5

Add a Home



Note: No change were made on the file, to registrer the new house to the file the user needs to save the changes.

Figure 6

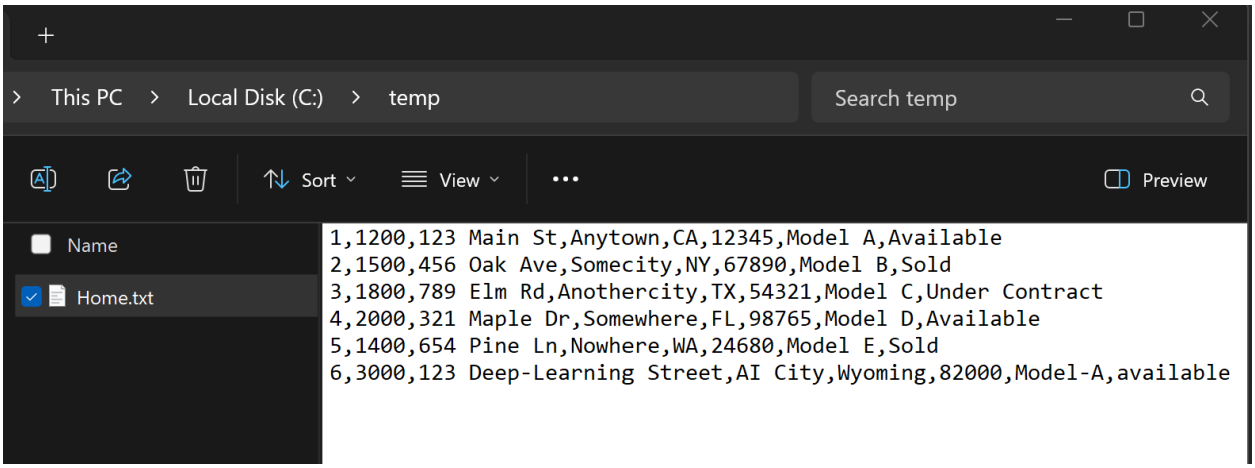
Saving Changes

```
      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
6
```

```
------------------ Saving Homes arrayList to file ---------------------------
The Home id: 1 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 2 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 3 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 4 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 5 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 6 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!

The home inventory was saved successfully!
--------------------------------------------------------------------------------
```

Continue next page.

```
--------------------------------------------------------------------------------

      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
```



Note: The new home was added to the file

Figure 7

Updating a Home and Saving Changes to File

```
      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
3
Enter the id of the home to update: 2

------------------------- Finding home to update -----------------------------
The home was found!
The Home id: 1 was found!
The Home id: 2 was found!
The home was found!
--------------------------------------------------------------------------------
```

Continue next page.

```
        Update Home Menu
Choose the information to update:
1. Square Feet
2. Address
3. City
4. State
5. Zip Code
6. Home Model
7. Sale Status
8. Exit update menu
1
Enter new square feet: 0000
The home's square feet were set successfully!

        Update Home Menu
Choose the information to update:
1. Square Feet
2. Address
3. City
4. State
5. Zip Code
6. Home Model
7. Sale Status
8. Exit update menu
2
Enter new address: 123 Ozone Avenue
The home's address was set successfully!

        Update Home Menu
Choose the information to update:
1. Square Feet
2. Address
3. City
4. State
5. Zip Code
6. Home Model
7. Sale Status
8. Exit update menu
3
Enter new city: Paris
The home's city was set successfully!
```

```
        Update Home Menu
Choose the information to update:
1. Square Feet
2. Address
3. City
4. State
5. Zip Code
6. Home Model
7. Sale Status
8. Exit update menu
4
Enter new state: Texas
The home's state was set successfully!

        Update Home Menu
Choose the information to update:
1. Square Feet
2. Address
3. City
4. State
5. Zip Code
6. Home Model
7. Sale Status
8. Exit update menu
5
Enter new zip code: 74321
The home's zip code was set successfully!
Zip code updated successfully.

        Update Home Menu
Choose the information to update:
1. Square Feet
2. Address
3. City
4. State
5. Zip Code
6. Home Model
7. Sale Status
8. Exit update menu
7
Enter sale status (sold, available, under contract): available
The home's sale status was set successfully!
Sale status updated successfully.
```

Continue next page.

```
        Update Home Menu
Choose the information to update:
1. Square Feet
2. Address
3. City
4. State
5. Zip Code
6. Home Model
7. Sale Status
8. Exit update menu
8

------------------------- Finding home to update ----------------------------
The Home id: 1 was found!
The Home id: 2 was found!
The home was updated successfully!
-------------------------------------------------------------------------------

--------------------- Displaying the updated home data ------------------------
The home was found!
The Home id: 1 was found!
The Home id: 2 was found!
The home was found!

---- Home id: 2 ----
[ square feet: 0, Address: 123 Ozone Avenue, City: Paris, State: Texas, Zip Code: 74321, Model Name: Model B, Sale Status: available ]
-------------------------------------------------------------------------------

      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
```

```
      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
6
```

Continue next page.

```
------------------- Saving Homes arrayList to file -----------------------------
The Home id: 1 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 2 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 3 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 4 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 5 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 6 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!

The home inventory was saved successfully!
--------------------------------------------------------------------------------
```

Continue next page.

```
      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
```

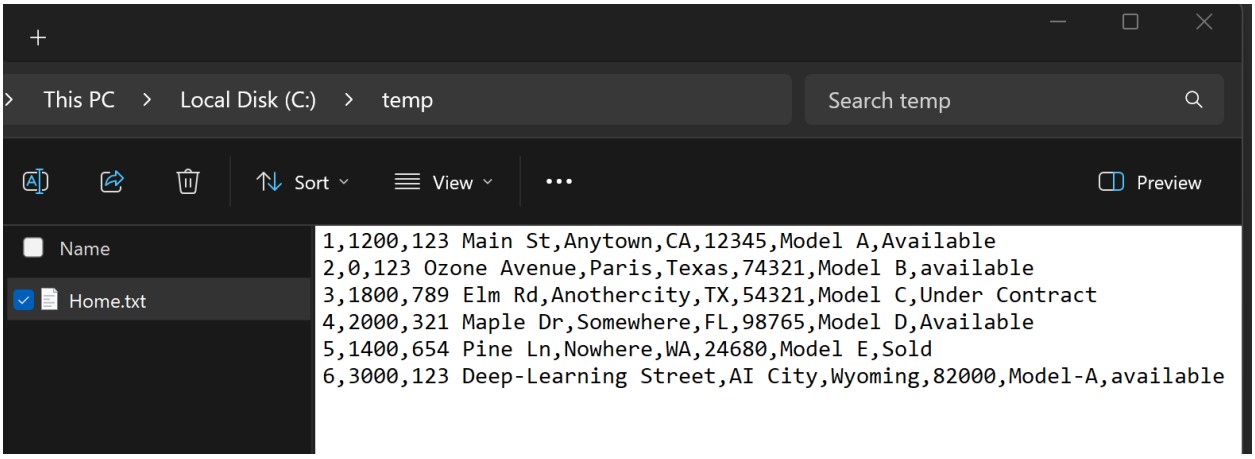


Figure 8

Removing a Home and Saving Change to File

```
      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
2
Enter the ID of the home to remove: 2

------------------------- Finding home to remove ----------------------------
The Home id: 1 was found!
The Home id: 2 was found!
The Home id: 3 was found!
The Home id: 4 was found!
The Home id: 5 was found!
The Home id: 6 was found!
The home was removed successfully!
-------------------------------------------------------------------------------
```

Continue next page.

```
------------------- Saving Homes arrayList to file ----------------------------
The Home id: 1 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 3 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 4 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 5 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!
The Home id: 6 was found!
The square feet were found!
The address was found!
The city was found!
The state was found!
The zip code was found!
The model name was found!
The sale status was found!

The home inventory was saved successfully!
-------------------------------------------------------------------------------
```

Continue next page.

```
      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
```

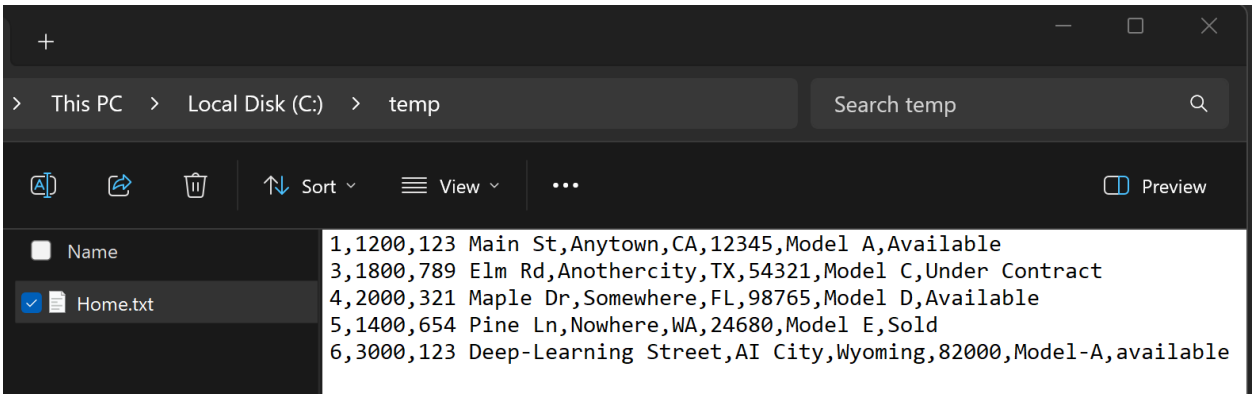


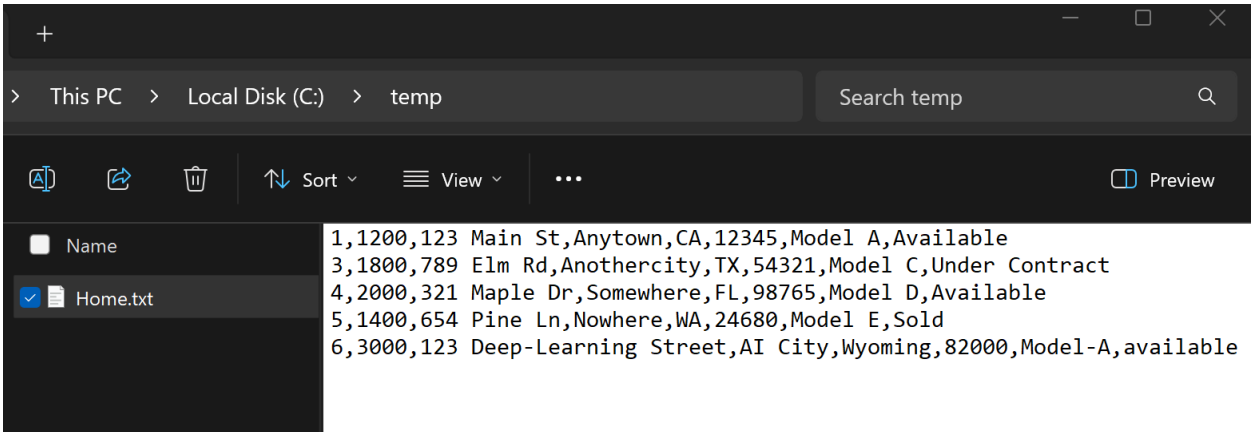
Figure 9

Updating Square Feet and Existing the Program Without Saving Changes

```
      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
3
Enter the id of the home to update: 4

------------------------- Finding home to update -----------------------------
The home was found!
The Home id: 1 was found!
The Home id: 3 was found!
The Home id: 4 was found!
The home was found!
--------------------------------------------------------------------------------
```

Continue next page.

```
---------------------- Displaying the updated home data ------------------------
The home was found!
The Home id: 1 was found!
The Home id: 3 was found!
The Home id: 4 was found!
The home was found!

---- Home id: 4 ----
[ square feet: 1111, Address: 321 Maple Dr, City: Somewhere, State: FL, Zip Code: 98765, Model Name: Model D, Sale Status: Available ]
--------------------------------------------------------------------------------

       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
7
Do you want to save the changes? [Y/N]: n
Do you want to delete the inventory file? [Y/N]: n

Thank you for using the Home Inventory Manager program!
```



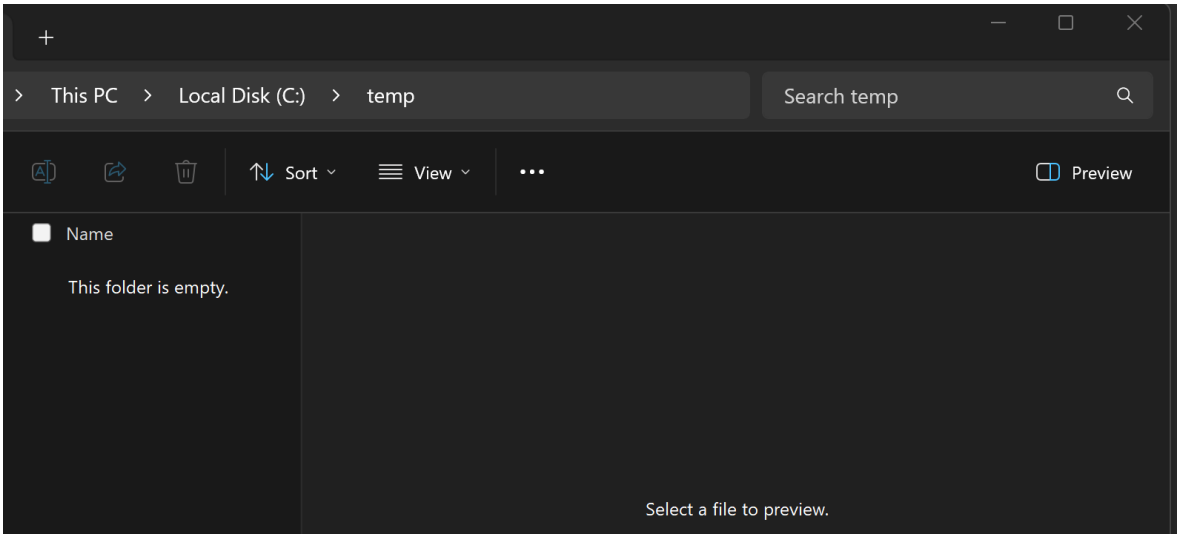

```
1,1200,123 Main St,Anytown,CA,12345,Model A,Available
3,1800,789 Elm Rd,Anothercity,TX,54321,Model C,Under Contract
4,2000,321 Maple Dr,Somewhere,FL,98765,Model D,Available
5,1400,654 Pine Ln,Nowhere,WA,24680,Model E,Sold
6,3000,123 Deep-Learning Street,AI City,Wyoming,82000,Model-A,available
```

Note: The changes to the square feet of the home with id 4 were not saved to the file. Additionally, when exiting the program, the user was prompted to save the changes; however, the user entered 'n'. When compared to Figure 3, no changes to the data were made (in Figure 3), and the user was not prompted to save the changes when exiting the program.

Figure 10

Deleting the File When Exiting the Program

```
        *********************************
        *     Home Inventory Manager     *
        *********************************

Please enter the inventory file path-name of an existing file or the path-name of a new file:
C:\Temp\Home.txt
The file C:\Temp\Home.txt already exists!

--------------------------- Loading File Data ------------------------------
The Home object was created successfully!
The Home with the id: 1 was loaded into the Homes arrayList.
The Home object was created successfully!
The Home with the id: 3 was loaded into the Homes arrayList.
The Home object was created successfully!
The Home with the id: 4 was loaded into the Homes arrayList.
The Home object was created successfully!
The Home with the id: 5 was loaded into the Homes arrayList.
The Home object was created successfully!
The Home with the id: 6 was loaded into the Homes arrayList.

The HomeInventory object was created successfully!
-------------------------------------------------------------------------------

       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
7
Do you want to delete the inventory file? [Y/N]: y
The file name-path was found!
File deleted successfully.

Thank you for using the Home Inventory Manager program!
```



b- Input Validation Figures 11 To 15

Figure 11

File Path-name Validation

```
        *********************************
        *     Home Inventory Manager     *
        *********************************

Please enter the inventory file path-name of an existing file or the path-name of a new file:
Temp\Home.txt
-- An error occurred: The system cannot find the path specified
Please try again!
Please enter the inventory file path-name of an existing file or the path-name of a new file:
C:\Temp\Home.txt

The file path-name that you entered is a new file!
If you want to create a file populated with fake data, enter [Y].
Otherwise, to create a new empty file, enter [N]
iii
-- Invalid Entry! Please try again.

The file path-name that you entered is a new file!
If you want to create a file populated with fake data, enter [Y].
Otherwise, to create a new empty file, enter [N]
1
-- Invalid Entry! Please try again.

The file path-name that you entered is a new file!
If you want to create a file populated with fake data, enter [Y].
Otherwise, to create a new empty file, enter [N]
Y

The HomeInventory object was created successfully!
```

...

```
The home inventory was saved successfully!
The file C:\Temp\Home.txt was successfully created and populated with fake data!
--------------------------------------------------------------------------------

       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
```

Figure 12

Main Menu Input Validation

```
       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
-1
Invalid choice. Please try again.

       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
0
Invalid choice. Please try again.

       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
8
Invalid choice. Please try again.

       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
aa
Invalid choice. Please try again.

       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
```

Figure 13

Add New Home Input Validation

```
       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
1
Enter square feet: -111
-- Error: Invalid square feet. Please enter a positive integer.
Enter square feet: hjj
-- Error: Invalid square feet. Please enter a positive integer.
Enter square feet: 1234
Enter address: 123 Ai Street
Enter city: Robot city
Enter state: Wyoming
Enter zip code: -12345
-- Error: Invalid zip code. Please enter a 5-digit zip code.
Enter zip code: 1234
-- Error: Invalid zip code. Please enter a 5-digit zip code.
Enter zip code: 123456
-- Error: Invalid zip code. Please enter a 5-digit zip code.
Enter zip code: 1qw2
-- Error: Invalid zip code. Please enter a 5-digit zip code.
Enter zip code: 12345
Enter the home model: Model A
Enter sale status (sold, available, under contract): 123
-- Error: Invalid sale status. Please enter 'sold', 'available', or 'under contract'.
Enter sale status (sold, available, under contract): we
-- Error: Invalid sale status. Please enter 'sold', 'available', or 'under contract'.
Enter sale status (sold, available, under contract): Sold

------------------- Adding home to the Homes arrayList ----------------------
The Home object was created successfully!
The Home with id 6 was added successfully to the Homes arrayList.

Home added successfully.
--------------------------------------------------------------------------------
```

...

```
---- Home id: 6 ----
[ square feet: 1234, Address: 123 Ai Street, City: Robot city, State: Wyoming, Zip Code: 12345, Model Name: Model A, Sale Status: sold ]
--------------------------------------------------------------------------------

      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
```

Figure 14

Remove Home Input Validation

```
       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
2
Enter the ID of the home to remove: -1
Invalid id, the id needs to be a possitive integer.
Please try again!

       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
2
Enter the ID of the home to remove: jjj
Invalid id, the id needs to be a possitive integer.
Please try again!

       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
2
Enter the ID of the home to remove: 1jjj
Invalid id, the id needs to be a possitive integer.
Please try again!
```

Continue next page.

```
       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
2
Enter the ID of the home to remove: 1234

-------------------------- Finding home to remove ----------------------------
The Home id: 1 was found!
The Home id: 2 was found!
The Home id: 3 was found!
The Home id: 4 was found!
The Home id: 5 was found!
-- An error occurred: Error removing home: The home with the provided ID was not found!
Please try again.

       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
2
Enter the ID of the home to remove: 1

-------------------------- Finding home to remove ----------------------------
The Home id: 1 was found!
The Home id: 2 was found!
The Home id: 3 was found!
The Home id: 4 was found!
The Home id: 5 was found!
The home was removed successfully!
--------------------------------------------------------------------------------

       Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
```

⚠️ Note:
Home ID validation was demonstrated in Figure 14, “Remove Home Input Validation.” Data home validation such as square feet, address, city, state, zip code, model name, and sale status were demonstrated in Figure 13, “Add New Home Input Validation.” The main menu choices ‘Display a home’ and ‘Update a home’ utilize the same methods to validate their inputs as those used by ‘Remove a home’ and ‘Add a new home’ to validate their user inputs; therefore, their input validation was already tested.

Figure 15

Exit and Save Changes Input Validation

```
      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
7
Do you want to save the changes? [Y/N]: uuuer
Invalid input. Please try again!
Do you want to save the changes? [Y/N]: 0
Invalid input. Please try again!
Do you want to save the changes? [Y/N]: 1
Invalid input. Please try again!
Do you want to save the changes? [Y/N]: y
```

…

```
The home inventory was saved successfully!
--------------------------------------------------------------------------------
Do you want to delete the inventory file? [Y/N]: 2
Invalid input. Please try again!
Do you want to delete the inventory file? [Y/N]: dsa
Invalid input. Please try again!
Do you want to delete the inventory file? [Y/N]: d
Invalid input. Please try again!
Do you want to delete the inventory file? [Y/N]: n

Thank you for using the Home Inventory Manager program!
```

Note:
The ‘Exit’, ‘Save Changes’, and ‘populate a file with fake data’ utilize the *yesOrNo()* method from the *InputValidation class*.

c- Exceptions Figures 16 To 20

In this section, screenshots of exception handling are shown. The program handles exceptions by passing them from the *Home class* to the *HomeInventory class*, and then to the *Main class*, where the exceptions and errors are displayed to the user. This is done by the methods in the *Home class* and *HomeInventory class* catching exceptions and errors and passing them to the class utilizing the method functionality using the '*trow*' statements in the '*try*' and '*catch*' blocks.

For time seek, I will show only how the *getId()* method in *Home Class* and the *updateHomeById()* method in the *HomeInventory class* handle exceptions, also other getters and setters method handle the exception in the same way. Additionally, I will show how the program handles IOException when manipulating the file.

Figure 16

Handling Exception getId() From the Home Class

```java
/**
 * Returns the unique identifier of the home.
 *
 * @return the unique identifier of the home
 * @throws NullPointerException if the id is null
 */
public Integer getId() throws NullPointerException {
    try {
        id = null;  // <--- testing exception handling
        if (id == null) {
            throw new Exception("The value of the id is null!");
        }
        System.out.println("The Home id: " + id + " was found!");  // success message
        return id;
    } catch (Exception e) {
        throw new NullPointerException(e.getMessage());   // failure message
    }
}
```

```
      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
2
Enter the ID of the home to remove: 3

------------------------- Finding home to remove ----------------------------
-- An error occurred: Error removing home: The value of the id is null!
Please try again.

      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
```

Note: In this example the exception was passed from the *Home.getId()* to *HomeInventory. removeHomeById()* to *Main.removeHome()*.

Figure 17

Handling Exception updateHomeById () From the HomeInventory Class

```java
/**
 * Updates a home in the inventory by its id.
 *
 * @param id          the ID of the home to update
 * @param updatedHome the updated home data
 * @throws Exception if the home with the specified ID is not found or an error occurs
 */
public void updateHomeById(Integer id, Home updatedHome) throws Exception {
    try {
        id = -1;  // <--- testing exception handling
        for (int i = 0; i < homes.size(); i++) {
            if (homes.get(i).getId() == id) {
                homes.set(i, updatedHome);
                System.out.println("The home was updated successfully!");  // success message
                return;  // exit the method after updating
            }
        }
        throw new Exception("The home with the provided id was not found!");
    } catch (Exception e) {
        throw new Exception(e.getMessage());   // failure message
    }
}
```

```
      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
3
Enter the id of the home to update: 4

------------------------- Finding home to update ----------------------------
The home was found!
The Home id: 2 was found!
The Home id: 3 was found!
The Home id: 4 was found!
The home was found!
-------------------------------------------------------------------------------

        Update Home Menu
Choose the information to update:
1. Square Feet
2. Address
3. City
4. State
5. Zip Code
6. Home Model
7. Sale Status
8. Exit update menu
1
Enter new square feet: 1234
The home's square feet were set successfully!
```

```
        Update Home Menu
Choose the information to update:
1. Square Feet
2. Address
3. City
4. State
5. Zip Code
6. Home Model
7. Sale Status
8. Exit update menu
8

------------------------- Finding home to update ----------------------------
The Home id: 2 was found!
The Home id: 3 was found!
The Home id: 4 was found!
The Home id: 5 was found!
-- An error occurred: Error updating home: The home with the provided id was not found!
Please try again.

      Menu
1. Add a new home
2. Remove a home
3. Update a home
4. List all homes
5. Display a home
6. Save changes
7. Exit
```

Note: In this example the exception was passed from *HomeInventory*. *updateHomeById()* to *Main.removeHome()*. The *Home.getId()* was usded to find the home initialy.

Figure 18

Invalid path IOException

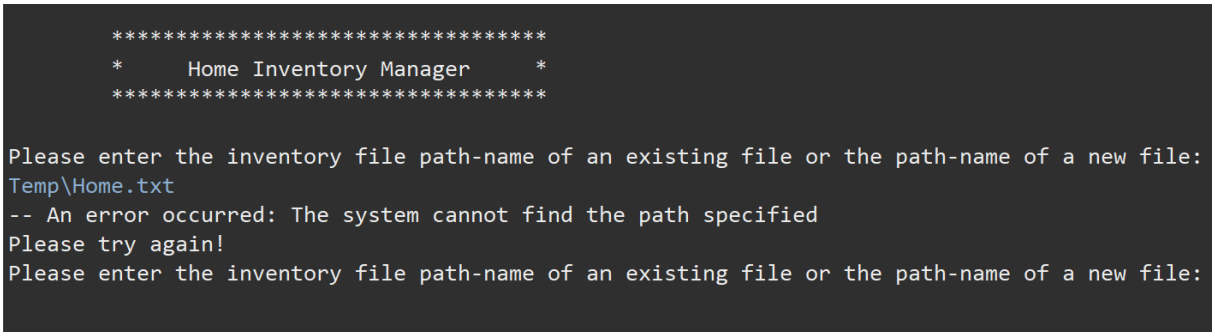


Figure 19

The File is Open in Other Application

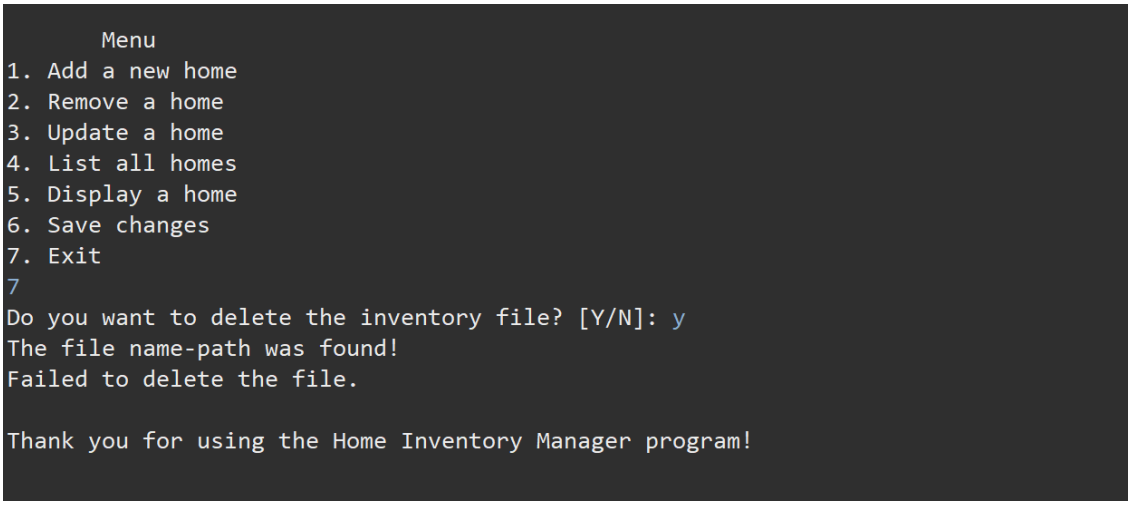
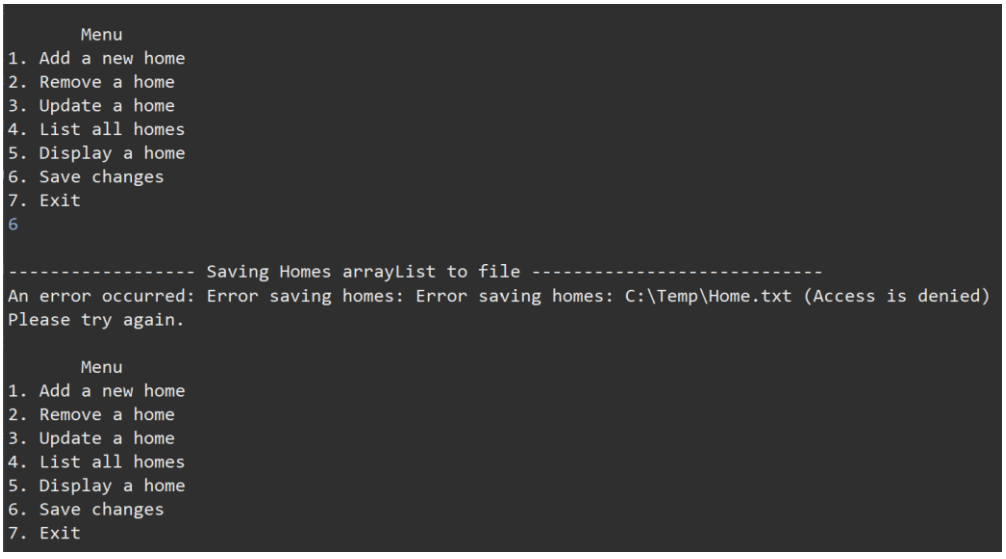


Figure 20

The File is Read Only



As shown in Figure 1 through Figure 20 the tests run without any issues, displaying the correct output as expected, inputs were validated, and exceptions/errors were caught.