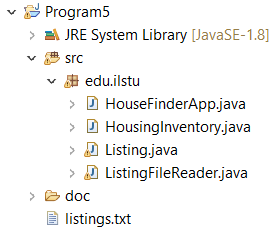
Program 5

Housing Inventory Using Arrays and Aggregate Classes

## In this assignment you will be creating an application to work with data from the local “For Sale By Owner” (FSBO) organization. You have been given a text data file that will be read into a data structure using an array of Listing class data type. This inventory file will be read into the system upon launching the program. Good news – the class file that does the heavy lifting is provided. It will be up to you to look at the structures it creates and learn how to manipulate them – sort it, report on it, search it. Like Program 4 it will be primarily menu-driven. Make sure that you follow good practices when you ask for input. The choice from a menu should always be validated. If it is not a valid input, the user should be asked to try again until they reply with a valid choice.

## Step #1 – Setup the project and import the files:

Your first step will be to import the class files that have been provided into your project. These files are incomplete and you will be adding to them. Remember to import/copy the text data file which is usually placed in the root of the project folder for simplicity. Here’s how your initial project setup should look.



## Step #2 – Test setup:

The provided code is operational. Run the HouseFinderApp class to confirm you’ve correctly import all resources. You should get one of the following 2 messages:

If your files are correctly placed:

If your files are not correctly placed:



## Step #3 – Add aggregation:

The HousingInventory class contains an array of Listings. The files that you received will import the text file and create a structure of Listing objects that are stored into a HousingInventory array; however, the implementation of the required aggregation is left to you. The Listing class needs to be updated to have a "has a" relationship with the Lot class and to have a "has a" relationship with the House class. This will require modifying the file reading method. This will also require you to create a Lot class file and a House class file and add to the instance variables in the Listing class.

You may want to run the program line by line in the Eclipse debugger just to be sure you understand the array of objects that has been provided for you to write your program.

## Step #4 – Build out the User Interface:

Use the sample runs given in the provided document for formatting and prompts. Look at the sample runs carefully for examples of types of input to be validated and what the program flow should specifically look like. You should attempt to make your program look the same.

Here is what your start menu should look like:

Welcome to your local FSBO - Let's find your next home

(1) Display the entire housing inventory

(2) Display the entire housing inventory - sorted by year built

(3) Display homes with 4 or more bedrooms

(4) Display homes listed at $150k and below

(5) Output a list of properties with >= 0.25 acres

(6) Find home by property id

(7) Quit

Enter your choice (1-7):

## Step #5 – Complete the assignment:

Unlike prior assignments, all of the details of the assignment will not be provided for you. Here are a few hints to help you:

1. In addition to the Project structure listed above, you can expect to have two additional classes (Lot and House).
2. The HousingInventory data structure contains an array that is instantiated to a size of 100. The file has 32 records in it. That means that 68 elements of the array will be null. You will need to come to a design decision on how you will address these null values when working with the array (e.g. using a separate size variable, resizing the array, or a combination of both).
3. Get started early and work methodically, one step at a time, testing along the way.

**HouseFinderApp class**

This is your driver class and will handle your interactions with the user of the application. All other work should be delegated to the service classes (ListingFileReader, HousingInventory, Listing, etc.).

**ListingFileReader class**

This class will be used to read the real estate listings file into the HousingInventory class. Note: You will need to enhance the readInventory() method as you add aggregation to the Listing class.

**HousingInventory class**

Keeps track of all of the real estate listings

Special note: the sorting of the array of listings, the inventory, will need to be done using a sort algorithm. The book describes a simple sort algorithm called the **selection** **sort** which should be adequate for this. You will need to adapt the book’s integer sorting example to sort on the House year built. In addition, your sort should not change the order of the underlying inventory array.

* Instance Variables
  + inventory - This will be an array of Listing class data type.
  + size - The number of filled spots in the array will also need to be tracked.
  + Constant for the max array size of 100
* Methods
  + Getters and setters for instant variables you need to access.
  + Other methods suggested are below in Java Doc format

* + - **getInventory**

public edu.ilstu.Listing[] getInventory()

**Returns:**

An unsorted array of all Listing objects in the housing inventory

* + - **getInventorySortedByYear**

public edu.ilstu.Listing[] getInventorySortedByYear()

**Returns:**

A sorted array of all Listing objects in the housing inventory (sorted ascending by year)

* + - **getInventoryByBedrooms**

public edu.ilstu.Listing[] getInventoryByBedrooms(int minBedrooms)

**Parameters:**

minBedrooms - The minimum number of bedrooms desired for a listing

**Returns:**

an unsorted array of Listing objects of >= minBedrooms

* + - **getInventoryByPrice**

public edu.ilstu.Listing[] getInventoryByPrice(double maxPrice)

**Parameters:**

maxPrice - The maximum price desired for a listing

**Returns:**

An unsorted array of Listing objects of <= maxPrice

* + - **getInventoryByLotSize**

public edu.ilstu.Listing[] getInventoryByLotSize(double minAcreage)

**Parameters:**

minAcreage - The minimum acreage desired for a listing

**Returns:**

An unsorted array of Listing objects of >= minAcreage

* + - **getListing**

public edu.ilstu.Listing getListing(java.lang.String propertyId)

**Parameters:**

propertyId - The unique identifier of a Listing in the housing inventory.

**Returns:**

The Listing object for the propertyId, null if property id is not found.

**Listing class**

Keeps track of the information about a real estate listing.

* Instance Variables
  + Study the ListingFileReader class to understand which instance variables you will need.
* Methods
  + Listing—two different constructors, one that will accept 1 parameter and one that accepts 4 parameters, including the aggregate Lot and House instance variables
  + Getter methods for instance variables that you need to access
  + toString method—the format of this should be setup to use for a tabular list layout like given in the sample run document
  + equals method—this should accept a Listing object as a parameter and should return a true or false based on whether or not the Property Ids match.

**Lot class**

Keeps track of a real estate listing’s lot information (address and lot size)

* Constant Values
  + The Lot class stores the lot size in square feet; however, the lot size is output in acres. Use a constant to store the conversion rate. Note: There are 43,560 square feet in 1 acre.
* Instance Variables
  + Study the ListingFileReader class to understand which instance variables you will need.
* Methods
  + Lot – two different constructors
  + Getters for instant variables you need to access.
  + toString, that will print the information for the address and acreage (check the sample run for the format)

**House class**

Keeps track of a real estate listing’s house information (# or bedrooms, # of bathrooms, year built, finished square feet, etc.).

* Instance Variables
  + Study the ListingFileReader class to understand which instance variables you will need.
* Methods
  + Getters for instant variables you need to access.

**Design Requirements**

* Create a class diagram for 4 classes – Housing Inventory, Listing, Lot, and House.
* The classes should be placed in a **package** with the name **edu.ilstu**

**Submission**

1. UML class diagram for the Housing Inventory, Listing, Lot, and House classes.
2. Zip your .java source files for all of the classes together into a zip file.
3. Name the zip file with **your initials** and Program 5
   * Example format: KES-Program5.zip