Work on this assignment by yourself.

Write the code for the assignment below and upload the zipped src folder to the learning hub (learn.bcit.ca 🡪 Activities 🡪 Assignments 🡪Assignment 2) before the dropbox due date.

Include your full name at the top of each file, using a Java Javadoc   
/\*\* @author \*/ tag (for example: /\*\* @author Jason Wilder \*/).

In Assignment 2 we are building on top of what we created in the first assignment. This time around we will focus on design by incorporating Inheritance, reading data files, and on providing an interface so users can interact with the program at runtime. Changes and additions below are noted in **blue** font.

| **Class** | **Instance Variables** | **Methods** |
| --- | --- | --- |
| Address | **Unit number** (String: must be one to four characters)  **Street number** (int must be 0 to 999999)  **Street name** (String: must be one to twenty characters)  **Postal code** (String: must be either length 5 or length 6)  **City** (String: must be one to thirty characters) | *Get* methods for all instance variables  toString() to display the state of Address objects |
|  |  |  |
| Property | **Price in USD** (double, must be positive)  **Address** (Address, not null)  **Type** (String: must be one of: “residence”, “commercial”, or “retail”)  **Property ID** (String: must be one to six characters) | Property will become a supertype that will hold common fields and behaviours of the subtype  Property will no longer support public int getNumberOfBedrooms() nor public boolean hasSwimmingPool() nor the associated fields  *Get* methods for all instance variables  *Set* method for price  toString() to display the state of Property objects |
| Residence extends Property | **numberOfBedrooms**(int: must not be less than 1)  **swimmingPool** (boolean: indicates presence of a pool)  **strata** (boolean: indicates if part of a strata) | Subtype of Property  *Get* methods for all fields  toString() for displaying the state of Residence types |
| Commercial extends Property | **loadingDock** (boolean: indicates if it has a loading dock)  **highwayAccess** (boolean: indicates if easy access to a highway) | Subtype of Property  *Get* methods for all fields  toString() for displaying the state of Commercial types |
| Retail extends Property | **squareFootage** (int: the amount of floor space available)  **customerParking** (boolean: indicates if customer parking is available) | Subtype of Property  *Get* methods for all fields  toString() for displaying the state of Retail types |
| AddressReader |  | **public static ArrayList<Address> readAddressData(File file) throws FileNotFoundException**: reads “address\_data.txt” (provided) and adds Address objects to an ArrayList<Address> and returns it. |
| PropertyReader |  | **public static ArrayList<String> readPropertyData(File file) throws FileNotFoundException**: reads “property\_data.txt” (provided) and adds Strings (for each line) to an ArrayList<String> and returns it. |
| Agency | **Name** (String, 1 to 30 characters)  **Properties** (HashMap of properties; key is property id, value is a Property) | **addProperty(property):** adds the (non-null) property to the HashMap  **removePropery(propertyId):** removes the property whose ID matches the parameter, from the HashMap  **getProperty(propertyId):** returns the property whose ID matches the parameter, from the HashMap (or null if there is no match)  **getTotalPropertyValues()**: returns the total amount in USD of all Properties  **getPropertiesWithPools()**: This method needs to be modified to return an ArrayList<Residence> type properties since only that type will now have swimming pools  **getPropertiesBetween(minUsd, maxUsd)**: returns an array of properties whose price falls in the range specified by the parameters…or null if there are none  **getPropertiesOn(streetName)**: returns an ArrayList of addresses which are on the specified street…or null if there are none  **getPropertiesWithBedrooms(minBedrooms, maxBedrooms)**: returns a HashMap of Residences (key is property id, value is the Residence) whose number of bedrooms falls in the range specified by the parameters…or null if there are none. NOTE only Residence type have bedrooms so you will have to modify this method to look at Residence types only.  **getPropertiesOfType(propertyType)**: this existing method needs to be modified to return an ArrayList<Property> that hold the subtype specified in the parameter.  **getPropertiesWithLoadingDock()**: returns an ArrayList<Commercial> that holds only Commercial properties that have a loading dock available  **getPropertiesWithHighwayAccess()**:returns an ArrayList<Commercial> that holds only Commercial properties that have highway access  **getProperties WithSquareFootage(int squareFootage)**: returns an ArrayList<Retail> that holds properties where square footage is at least the parameter value.  **getPropertiesWithCustomerParking()**:returns an ArrayList<Retail> that holds properties where customer parking is available.  **getPropertiesWithStrata():** returns ArrayList<Residence> that hold only the Residences that are in a strata. |
| Assignment2  (driver class) | **Agency**: reference to the Agency class | **public void init() throws FileNotFoundException**: This method gets the ArrayList,<Address> and ArrayList<String> form AddressReader and PropertyReader, and uses them to create subtype Objects and adds them to the Agency.HashMap<String, Property> properties  **public void doSearches()**: This method provides the primary user interface through command prompts that will allow the user to choose which search operations to perform. See the **Sample Interface** below. Each search will display results to the console.  **public static void main(String[] args) throws FileNotFoundException**: Will create an instance of Assignment2 and use that to call init() and then doSearches(). |
| Sample Interface. |  | Welcome to our Property search.  Choose one of the following options:   1. General Queries – will present the General Queries menu 2. Residence Queries – will present the Residence Queries menu 3. Commercial Queries – will present the Commercial Queries menu 4. Retail Queries – will present the Retail Queries menu 5. Exit – will exit the program   ------------------------------------------------------------------  General Queries   1. By Property ID – will further prompt to ender Property ID, then call Agency.getProperty(propertyId) and display the result to the console 2. By Price – will further prompt to enter both min and max ranges, call Agency.getPropertiesBetween(minUsd, maxUsd) and display results to the console 3. By Street –will further prompt to enter the Street name, call Agency.getPropertiesOn(streetName) and display results to the console 4. By Type – will further prompt to enter the property type (residence, commercial, retail), call Agency.getPropertiesOfType(propertyType) and display results to the console 5. Back – will take you back to the main menu   ------------------------------------------------------------------  Residence Queries   1. By Bedroom – will further prompt to enter the min and max ranges, call Agency.getPropertiesWithBedrooms(minBedrooms, maxBedrooms) 2. By Pool – will call Agency.getPropertiesWithPools() and display only the residences that have swimming pools 3. By Strata – will call Agency.getPropertiesWithStrata() and display the results to the console 4. Back – will take you back to the main menu   ------------------------------------------------------------------  Commercial Queries   1. By Loading Dock – will call Agency.getPropertiesWithLoadingDock() and display results to the console 2. By Highway Access – will call Agency.getPropertiesWithHighwayAccess() and display the results to the console 3. Back – will take you back to the main menu   ------------------------------------------------------------------  Retail Queries   1. By Square Footage – will further prompt for min square footage value, call Agency.getProperties WithSquareFootage(int squareFootage) and display results to the console 2. By Customer Parking – will call Agency.getPropertiesWithCustomerParking() and display only Retail properties where customer parking is available 3. Back – will take you back to the main menu |
| Sample Program Run |  | Welcome to our Property search. Choose one of the following options:  1.General Queries  2.Residence Queries  3.Commercial Queries  4.Retail Queries  5.Exit  2  General Queries:  1.By Pool  2.By Bedroom  3.By Strata  4.Back  2  Enter minimum number of bedrooms:  2  Enter maximum number of bedrooms:  4  Residence [numberOfBedrooms=3, swimmingPool=false, strata=false, toString()=Property [priceUsd=1600000.0, address=Address [unitNumber= , streetNumber=333, streetName=elm street, postalCode=90111, city=los angeles], type=residence, propertyId=9000a]]  Residence [numberOfBedrooms=3, swimmingPool=false, strata=false, toString()=Property [priceUsd=740100.0, address=Address [unitNumber=b, streetNumber=711, streetName=country road, postalCode=v8h5f5, city=maple ridge], type=residence, propertyId=mr6789]]  Residence [numberOfBedrooms=2, swimmingPool=false, strata=true, toString()=Property [priceUsd=499000.0, address=Address [unitNumber=1a, streetNumber=777, streetName=56th avenue, postalCode=v7n2m8, city=surrey], type=residence, propertyId=abc123]]  General Queries:  1.By Pool  2.By Bedroom  3.By Strata  4.Back  4  Welcome to our Property search. Choose one of the following options:  1.General Queries  2.Residence Queries  3.Commercial Queries  4.Retail Queries  5.Exit  5  Goodbye for now! |