Class: HomeownerBO

- HomeownerBO class.
- This class calls HomeownerDAO methods.
- There should be no instance variables in this class.

Methods: **getHomeownerInfo**

- getHomeownerInfo - This method takes as an argument an integer quoteId and calls the getHomeowner method inside HomeownerDAO class.
- Be sure to pass quoteId as an argument.
- getHomeownerInfo method returns an Homeowner object.
- In case of an exception, throws and catch HomequoteSystemException and throw explicitly HomequoteBusinessException.

**==
===== ==**

Class: LocationBO

- LocationBO class This class calls LocationDAO methods.
- There should be no instance variables in this class.

Methods: saveHomeLocation

- This method takes as an argument a Location object and calls the saveLocation method inside of LocationDAO.
- Be sure to pass the Location object as an argument.
- saveHomeLocation method returns an integer which is the quoteId referring to this location.
- In case of an Exception, throws and catch HomequoteSystemException and throw explicitly HomequoteBusinessException.

Class: getHomeLocation

- This method takes as an argument an integer quoteId and calls the getLocation method inside of LocationDAO.
- Be sure to pass the integer quoteId as an argument. getHomeLocation method returns a Location object which contains information about one location.
- In case of an Exception, throws and catch HomequoteSystemException and throw explicitly HomequoteBusinessException.

Method: getQuoteIds

- This method takes as an argument a String userName and calls the getQuoteIds method inside of LocationDAO.
- Be sure to pass the String userName as an argument.
- getQuoteIds method returns a list of Location object which contains information about which location of one user.

- In case of an Exception, throws and catch HomequoteSystemException and throw explicitly HomequoteBusinessException.

== ===

Class: LoginBO class
- This class calls LoginDAO methods.
- There should be no instance variables in this class.

Methods: getUser
- This method takes as an argument a String userName and calls the getUser method inside of LoginDAO.
- Be sure to pass the String userName as an argument.
- getUser method returns a User object which contains information about one user.
- In case of an Exception, throws and catch HomequoteSystemException and throw explicitly HomequoteBusinessException.

Method: registerUser
This method takes as an argument a User object and calls the saveUser method inside of LoginDAO.
Be sure to pass the User object as an argument. registerUser does not returns, it just save a new user.
In case of an Exception, throws and catch HomequoteSystemException and throw explicitly HomequoteBusinessException.

== ==
Class: PolicyBO
- This class calls PolicyDAO methods.
- There should be no instance variables in this class.

Methods: getPolicies
- This method takes as an argument a String userName and calls the getPolicies method inside of PolicyDAO.
- Be sure to pass the String userName as an argument.
- getPolicies method returns a list of Policy object which contains information about policies of one user.
- In case of an Exception, throws and catch HomequoteSystemException and throw explicitly HomequoteBusinessException.

Methods:cancelPolicy

- This method takes as an argument a String policyKey and calls the cancelPolicy method inside of PolicyDAO.
- Be sure to pass the String policyKey as an argument.
- cancelPolicy method returns a Policy object which contains information about one policie of one user.
- In case of an Exception, throws and catch HomequoteSystemException and throw explicitly HomequoteBusinessException.

Method:renewPolicy

- method takes as an argument a String policyKey and calls the renewPolicy method inside of PolicyDAO.
- Be sure to pass the String policyKey as an argument.
- renewPolicy method returns a Policy object which contains information about the policy just updated.
- In case of an Exception, throws and catch HomequoteSystemException and throw explicitly HomequoteBusinessException.

Method: getDateAfterOneYear

- This method takes as an argument a String policyEffDate.
- Create a SimpleDateFormat object with the format "yyyy-MM-dd" and with the help of the Calendar interface increment the year by one.
- In case of an Exception, throws and catch HomequoteSystemException and throw explicitly HomequoteBusinessException.

==
===== ==

Class PropertyBO:

- This class calls PropertyDAO methods.
- There should be no instance variables in this class.

Method getProperty:

- This method takes as an argument an integer quoteId and calls the getProperty method inside of PropertyDAO.
- Be sure to pass the integer quoteId as an argument.
- getProperty method returns a Property object which contains information about one Property.
- In case of an Exception, throws and catch HomequoteSystemException and throw explicitly HomequoteBusinessException.

Method saveProperty:

- This method takes as an argument an integer quoteId and calls the getProperty method inside of PropertyDAO.
- Be sure to pass the integer quoteId as an argument.
- getProperty method does not returns but it saves a Property object which contains information about one Property.
- In case of an Exception, throws and catch HomequoteSystemException and throw explicitly HomequoteBusinessException.

==
===== ==

Class QuoteBO:

- This class calls PropertyDAO methods.
- There should be no instance variables in this class.

Method: processPremiumWithLoc

- This method takes two arguments, a Location object and a double availablePremium.
- This method return a premium result of type double.
- The premium is calculated with the following instruction:

  if the residence type passed inside the Location object equals: -Single-Family Home then:

  premium equals to -> availablePremium (availablePremium * 0.05 / 100)
- Condo or Duplex or Apartment then:

  premium equals to -> availablePremium (availablePremium * 0.06 / 100)
- Townhouse or Rowhouse then:

  premium equals to -> availablePremium (availablePremium * 0.07 / 100)

Method: getHomeValue:

- This method takes two arguments, a Property object and a integer numYearsOld.
- A variable homeValue is given inside this method.
- Calculate the real home value following the these instructions:

  if numYearOld is less than 5 then the real home value equals to homeValue - (homeValue * 10/100)

  if numYearOld is greater than 5 and less than 10 then the real home value equals to -> homeValue - (homeValue * 20/100)

  if numYearOld is greater than 10 and less than 20 then the real home value equals to -> homeValue - (homeValue * 30/100)

if numYearOld is greater than 20 then the real home value equals to -> homeValue - (homeValue * 50/100)

- After, return the real home value.

Method getQuote:

- This method takes as an argument an integer quoteId and calls the getQuote method inside of QuoteDAO.
- Be sure to pass the integer quoteId as an argument.
- getQuote method returns a Quote object which contains information about one quote.
- In case of an Exception, throws and catch HomequoteSystemException and throw explicitly HomequoteBusinessException.

Method saveQuote:

- This method takes as an argument a Quote Object and calls the saveQuote method inside of QuoteDAO.
- Be sure to pass the Quote Object as an argument. saveQuote method does not returns, it just saves the information contained in Quote object.
- In case of an Exception, throws and catch HomequoteSystemException and throw explicitly HomequoteBusinessException.

== ===== ===

Class: HomeownerDAO

- This class will attempt to connect to the database and perform queries according to business roles.
- When preparing statements be sure to use the constants variables inside SQLQueries.java class, create variables when needed.

Method: saveHomeowner

- This method takes as an argument an Homeowner object and saves the object's properties to the HomeownerInfo table.
- When a SQLException occurs, explicitly throw the HomequoteSystemException class with the error message.
- Be sure to close connection to the database inside the finally block.

Method: getHomeowner

- This method takes as an argument an integer quoteId and queries the HomeownerInfo for a home with matching quoteId.
- The method returns an Homeowner object with all the information retrieve from the HomeownerInfo table.

===
===== ==

Class LocationServlet:

- This class handles the request made from the location.jsp page

Method doPost:

- This methods handles the request coming from location.jsp.
- This methods checks if the is not an attribute in the session named "location".
- If this is true, then get all the parameter and store it inside a Location object.
- With the help of the LocationBO, save this location object by passing it to the saveHomeLocation method.
- Since this saveHomeLocation return a quoteId of type integer, be sure to set it to the location as well.
- Finally, set the location object to the session with a key named "location".

After performing the below instruction, dispatch to the homeowner info page.

In case of an Exception, set attribute to the request object with key named "message" with the error message. After dispatch to the error page.

Be sure to use the constants variables inside the class HomeInsuranceConstants when retrieving and dispatching.

===
===== ==

Database Requirements:

Table: **- Login**
Columns -USER_NAME (Cannot be Null, it must be unique, must be less than 20 characters long). -USER_PWD (Cannot be Null, must be less than 20 characters long).

Table: - Locations -Columns --QUOTE_ID (No changes required) -RESIDENT_TYPE (Cannot be Null, check that only contains one of these values): 'Single-Family Home', 'Condo', 'Townhouse', 'Rowhouse', 'Duplex', 'Apartment' --ADDRESS_LINE_1 (Cannot be Null) --ADDRESS_LINE_2 (Defaults to Null) --CITY (Cannot be Null) --LOCATION_STATE (Cannot be Null) --ZIP (Cannot be Null) --RESIDENCE_USE (Cannot be Null), check that only contains one of these values: ‘'Primary', 'Secondary', 'Rental Property' --USERNAME (Cannot be Null)

Tables: HomeownerInfo
Columns: --QUOTE_ID (Cannot be Null) --FIRST_NAME- (Cannot be Null, must be less than 31 characters long) --LAST_NAME - (Cannot be Null, must be less than 31 characters long) --DOB - (Cannot be Null, must be less than 11 characters long) --IS_RETIRED - (Cannot be Null) --SSN -(Cannot be Null, must be less than 10 characters long) --EMAIL_ADDRESS - (Cannot be Null, must be less than 51 characters long).

Quote - Columns --QUOTE_ID -(Cannot be Null, must be unique) --PREMIUM (Cannot be Null) --DWELLING_COVERAGE - Cannot be Null --DETACHED_STRUCTURE - Cannot be Null --PERSONAL_PROPERTY - Cannot be Null --ADDNL_LIVING_EXPENSE - Cannot be Null --MEDICAL_EXPENSE - Cannot be Null --DEDUCTIBLE - Cannot be Null

Property - Columns --QUOTE_ID (Cannot be Null) --MARKET_VALUE - (Cannot be Null) --YEAR_BUILT - (Cannot be Null) --SQUARE_FOOTAGE - (Cannot be Null) --DWELLING_STYLE - (Cannot be Null) --ROOF_MATERIAL - (Cannot be Null) --GARAGE_TYPE - (Cannot be Null) --NUM_FULL_BATHS - (Cannot be Null) --NUM_HALF_BATHS - (Cannot be Null) --HAS_SWIMMING_POOL - (Cannot be Null, Defaults to string 'FALSE')

Policies - Columns --POLICY_KEY - (Cannot be Null) --QUOTE_ID - (Cannot be Null) --POLICY_EFFECTIVE_DATE - (Cannot be Null) --POLICY_END_DATE - (Cannot be Null) --POLICY_STATUS - (Cannot be Null) --
**Make one constraint that checks if POLICY_EFFECTIVE_DATE < POLICY_END_DATE