

Project 4: Real Estate Database

CMIS 242

Attika Sheikh

May 7, 2020

Enumerated Type

/* Status.java

* Developer: Attiqah Sheikh

* Date: May 7, 2020

* Purpose: enumerated type named status with three enumerated literals

*/

```
public enum Status {  
  
    FOR_SALE, UNDER_CONTRACT, SOLD;}
```

Interface

/*StateChangeable.java

* Developer: Attiqah Sheikh

* Date: May 7, 2020

* Purpose: generic interface, bounded generic type parameter with enumerated type

*/

```
public interface StateChangeable<T extends Enum<T>> {  
  
    void changeState(T t); }
```

Property Class

/*Property.java

* Developer: Attiqah Sheikh

* Date: May 7, 2020

* Purpose: Implements State StateChangeable interface, contains five instance variables

* a changeState method that allows the status to be changed

* and a toString method that prints out a string containing values entered

*/

```

public class Property<T extends Enum<T>> implements StateChangeable<T> {

    private String propertyAddress;

    private int numberOfBedrooms, sqFeet, price;

    private Status saleStatus;

    //Constructor that accepts 4 parameters and sets status as FOR_SALE

    public Property(String propertyAddress, int numberOfBedrooms, int sqFeet, int price) {

        this.propertyAddress = propertyAddress;

        this.numberOfBedrooms = numberOfBedrooms;

        this.sqFeet = sqFeet;

        this.price = price;

        this.saleStatus = Status.FOR_SALE;}

    //Overridden method that allows the status of the property to be changed

    @Override

    public void changeState(T inputStatus) {

        this.saleStatus = (Status)inputStatus;}

    //Overridden method that returns a string of property information

    @Override

    public String toString() {

        return new String("Property Address: " + this.propertyAddress + "\nNumber of Bedrooms: " +

this.numberOfBedrooms

        + "\nSquare Feet: " + this.sqFeet + "\nPrice: $" + this.price + "\nStatus: " + this.saleStatus);

    }}

```

Project4 Class

/*Project4.java

* Developer: Attiqah Sheikh

* Date: May 7, 2020

* Purpose: contains the main method and any instance variable defining

* the database of the property, initializes GUI in JFrame, the JPanel object,

* buttons, and the overall GUI that allows the program to be displayed and run.

* Implements TreeMap as a key and property object as the value.

*/

import javax.swing.*;

import java.awt.*;

import java.awt.event.ActionEvent;

import java.util.TreeMap;

public class Project4 {

 //initializing GUI in mainframe

 public static void main(String[] args) {

 JFrame project4 = new JFrame("Real Estate Database");

 project4.setSize(325,250);

 project4.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

 project4.add(new realEstateDataPanel());

 project4.setVisible(true); }

 //assembles the GUI

 private static class realEstateDataPanel extends JPanel {

 //JLabels for required fields

```

private JLabel transactionLabel = new JLabel("Transaction No:");

private JLabel addressLabel = new JLabel("Address:");

private JLabel bedroomsLabel = new JLabel("Bedrooms:");

private JLabel squareLabel = new JLabel("Square Footage:");

private JLabel priceLabel = new JLabel("Price:");

//JComboBox entries for database

private String[] dataOperations = {"Insert", "Delete", "Find"};

private JComboBox dataList = new JComboBox(dataOperations);

//JComboBox entries for status

private Status[] statuses = {Status.FOR_SALE, Status.UNDER_CONTRACT, Status.SOLD};

private JComboBox statusList = new JComboBox(statuses);

//Text fields for property information and transaction id

private JTextField transactionField = new JTextField("");

private JTextField addressField = new JTextField("");

private JTextField bedroomsField = new JTextField("");

private JTextField squareField = new JTextField("");

private JTextField priceField = new JTextField("");

TreeMap<Integer, Property> propertyData = new TreeMap<>();

//Constructor for panel

public realEstateDataPanel() {

//Sets layout

setLayout(new GridLayout(7,2, 7,10));

this.add(transactionLabel); this.add(transactionField);

this.add(addressLabel); this.add(addressField);

```

```
this.add(bedroomsLabel); this.add(bedroomsField);
```

```
this.add(squareLabel); this.add(squareField);
```

```
this.add(priceLabel); this.add(priceField);
```

```
    JButton processButton = new JButton(new AbstractAction("Process") {
```

```
        @Override
```

```
        public void actionPerformed(ActionEvent e) {
```

```
            String processOption = String.valueOf(dataList.getSelectedItem());
```

```
            try {
```

```
                switch (processOption) {
```

```
                    case "Insert":
```

```
                        checkForDuplicates(getTransactionId());
```

```
                        propertyData.put(getTransactionId(), getPropertyInfo());
```

```
                        JOptionPane.showMessageDialog(null, "Property successfully stored in database.");
```

```
                        break;
```

```
                    case "Delete":
```

```
                        checkforExisting(getTransactionId());
```

```
                        propertyData.remove(getTransactionId());
```

```
                        JOptionPane.showMessageDialog(null, "Property successfully removed from database.");
```

```
                        break;
```

```
                    case "Find":
```

```
                        checkforExisting(getTransactionId());
```

```
                        Property getProperty = propertyData.get(getTransactionId());
```

```
                        JOptionPane.showMessageDialog(null, getProperty.toString());
```

```
                        break;
```

```

    }

    } catch(NumberFormatException nex) {

        JOptionPane.showMessageDialog(null, "Incorrect format for values entered.");

    } catch(DuplicateProperty dex) {

        JOptionPane.showMessageDialog(null, "Transaction id already exists in database.");

    } catch(PropertyNotFound pex) {

        JOptionPane.showMessageDialog(null, "Transaction id not found in database.");

    }

}

});

```

```

JButton changeButton = new JButton(new AbstractAction("Change Status") {

    @Override

    public void actionPerformed(ActionEvent e) {

        try {

            Status statusOption = (Status) statusList.getSelectedItem();

            checkforExisting(getTransactionId());

            Property changeProperty = propertyData.get(getTransactionId());

            changeProperty.changeState(statusOption);

            propertyData.put(getTransactionId(), changeProperty);

            JOptionPane.showMessageDialog(null, "Property status successfully changed in

database");

        } catch(PropertyNotFound pex) {

            JOptionPane.showMessageDialog(null, "Transaction id not found in database.");

```

```

    } catch(NumberFormatException nex) {

JOptionPane.showMessageDialog(null, "Incorrect format for values entered.");

    }

    }

});

```

```

this.add(processButton); this.add(dataList);

this.add(changeButton); this.add(statusList);

}

```

//reads the property info and throws and exception if incorrect information is entered

```

private Property getPropertyInfo() throws NumberFormatException {

    String address = addressField.getText();

    int bedrooms = getInput(bedroomsField);

    int squareFt = getInput(squareField);

    int price = getInput(priceField);

    return new Property(address, bedrooms, squareFt, price);

}

```

//reads transaction number and throws and exception if it is the wrong format

```

private int getTransactionId() throws NumberFormatException {

    return getInput(transactionField);

}

```

//verifies that property doesn't exist in database before being entered

```

private void checkForDuplicates(int transactionId) throws DuplicateProperty{

    if(propertyData.containsKey(transactionId)) {

```



```

        throw new DuplicateProperty();
    }
}

//verifies that property exists before being deleted

private void checkforExisting(int transactionId) throws PropertyNotFound {

    if(!propertyData.containsKey(transactionId)) {

        throw new PropertyNotFound();

    }

}

//Local method to convert entered Strings to integers

private int getInput(JTextField inputTextField) throws NumberFormatException {

    String inputString = inputTextField.getText();

    return Integer.parseInt(inputString);

}

//Defines custom exception if property already exists

private class DuplicateProperty extends Exception {

    public DuplicateProperty() {

        super();

    }

}

//Defines custom exception if property does not exist

private class PropertyNotFound extends Exception {

    public PropertyNotFound() {

        super(); } } } }

```

Generated GUI

Real Estate ...

Transaction No:

Address:

Bedrooms:

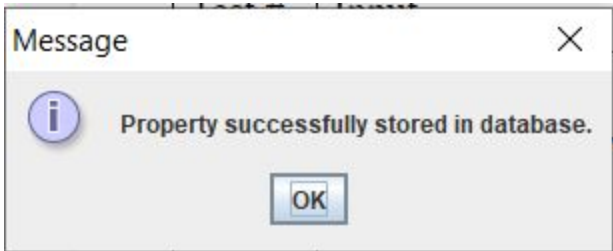

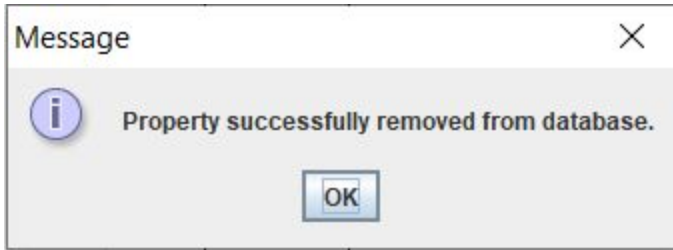
Square Footage:





Price:


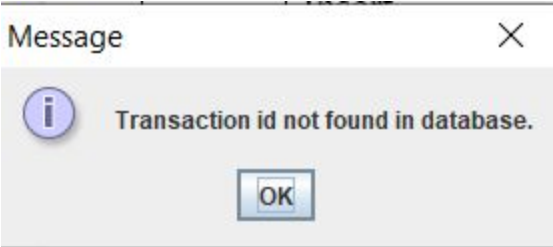
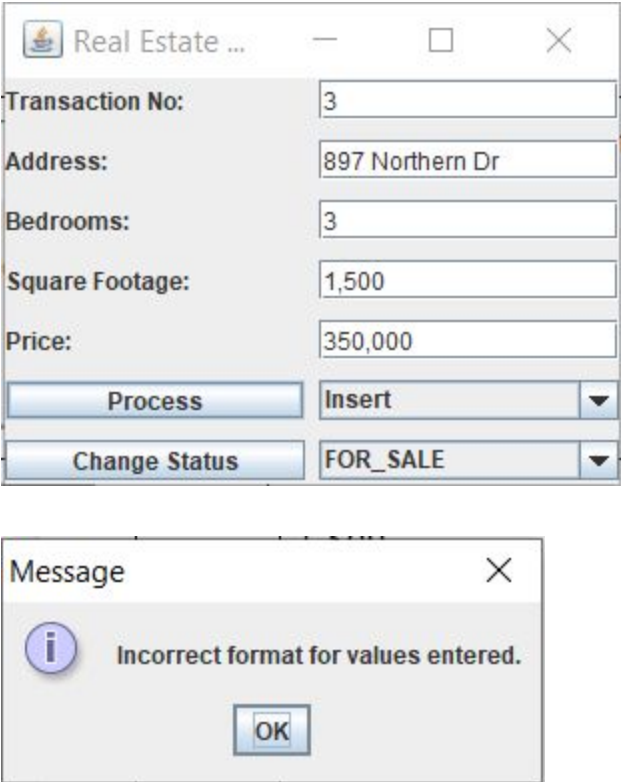
Process

Change Status

Test Cases

Test #	Input	Expected Output	Notes
#1	1 123 American Blvd 3 5000 750000	<u>Insert</u>  <u>Find</u>  <u>Remove</u> 	Required fields are entered. Property is first inserted to database, then found in database to display report, then removed from the database.

#2	2 456 Southern Dr 5 15000 3000000	<p>FOR_SALE</p> <div data-bbox="521 191 1057 558"><p>Message</p><p> Property Address: 456 Southern Dr Number of Bedrooms: 5 Square Feet: 15000 Price: \$3000000 Status: FOR_SALE</p><p>OK</p></div> <p>UNDER_CONTRACT</p> <div data-bbox="521 648 1057 1016"><p>Message</p><p> Property Address: 456 Southern Dr Number of Bedrooms: 5 Square Feet: 15000 Price: \$3000000 Status: UNDER_CONTRACT</p><p>OK</p></div> <p>SOLD</p> <div data-bbox="521 1104 1057 1472"><p>Message</p><p> Property Address: 456 Southern Dr Number of Bedrooms: 5 Square Feet: 15000 Price: \$3000000 Status: SOLD</p><p>OK</p></div> <p>Output for Change Status</p> <div data-bbox="521 1562 1198 1806"><p>Message</p><p> Property status successfully changed in database</p><p>OK</p></div>	Status changed of the property from FOR_SALE, UNDER_CONTRACT, SOLD
-----------	---	--	--

#3	2 897 Northern Dr 3 1500 350000 Insert		Enter invalid entries to prompt an error message.
#4	3 Delete		Insert a transaction id that is not in the database.
#5	3 897 Northern Dr 3 1,500 350,000 Insert		Insert incorrect format in database to prompt an error message.

Java Code

```
StateChangeable.java x Property.java Status.java Project4.java
1 /*StateChangeable.java
2  * Developer: Attiga Sheikh
3  * Date: May 7, 2020
4  * Purpose: generic interface, bounded generic type parameter with enumerated type
5  */
6 public interface StateChangeable<T extends Enum<T>> {
7     void changeState(T t);
8 }
```

```
StateChangeable.java Property.java Status.java x Project4.java
1 /* Status.java
2  * Developer: Attiga Sheikh
3  * Date: May 7, 2020
4  * Purpose: enumerated type named status with three enumerated literals
5  */
6 public enum Status {
7     FOR_SALE, UNDER_CONTRACT, SOLD;
8 }
9
```

```
StateChangeable.java Property.java x Status.java Project4.java
1 /*Property.java
2  * Developer: Attiga Sheikh
3  * Date: May 7, 2020
4  * Purpose: Implements State StateChangeable interface, contains five instance variables
5  * a changeState method that allows the status to be changed
6  * and a toString method that prints out a string containing values entered
7  */
8 public class Property<T extends Enum<T>> implements StateChangeable<T> {
9     private String propertyAddress;
10    private int numberOfBedrooms, sqFeet, price;
11    private Status saleStatus;
12    //Constructor that accepts 4 parameters and sets status as FOR_SALE
13    public Property(String propertyAddress, int numberOfBedrooms, int sqFeet, int price) {
14        this.propertyAddress = propertyAddress;
15        this.numberOfBedrooms = numberOfBedrooms;
16        this.sqFeet = sqFeet;
17        this.price = price;
18        this.saleStatus = Status.FOR_SALE;
19    }
20    //Overridden method that allows the status of the property to be changed
21    @Override
22    public void changeState(T inputStatus) {
23        this.saleStatus = (Status)inputStatus;
24    }
25    //Overridden method that returns a string of property information
26    @Override
27    public String toString() {
28        return new String("Property Address: " + this.propertyAddress + "\nNumber of Bedrooms: " + this.numberOfBedrooms
29        + "\nSquare Feet: " + this.sqFeet + "\nPrice: $" + this.price + "\nStatus: " + this.saleStatus);
30    }
31 }
32
```


StateChangeable.java Property.java Status.java Project4.java x

```
1  /*Project4.java
2  * Developer: Attiga Sheikh
3  * Date: May 7, 2020
4  * Purpose: contains the main method and any instance variable defining
5  * the database of the property, initializes GUI in JFrame, the JPanel object,
6  * buttons, and the overall GUI that allows the program to be displayed and run.
7  * Implements TreeMap as a key and property object as the value.
8  */
9  import javax.swing.*;
10 import java.awt.*;
11 import java.awt.event.ActionEvent;
12 import java.util.TreeMap;
13
14 public class Project4 {
15     //initializing GUI in mainframe
16     public static void main(String[] args) {
17         JFrame project4 = new JFrame("Real Estate Database");
18         project4.setSize(325,250);
19         project4.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20         project4.add(new realEstateDataPanel());
21         project4.setVisible(true);
22     }
23     //assembles the GUI
24     private static class realEstateDataPanel extends JPanel {
25         //JLabels for required fields
26         private JLabel transactionLabel = new JLabel("Transaction No:");
27         private JLabel addressLabel = new JLabel("Address:");
28         private JLabel bedroomsLabel = new JLabel("Bedrooms:");
29         private JLabel squareLabel = new JLabel("Square Footage:");
30         private JLabel priceLabel = new JLabel("Price:");
31         //JComboBox entries for database
32         private String[] dataOperations = {"Insert", "Delete", "Find"};
33         private JComboBox dataList = new JComboBox(dataOperations);
34         //JComboBox entries for status
35         private Status[] statuses = {Status.FOR_SALE, Status.UNDER_CONTRACT, Status.SOLD};
36         private JComboBox statusList = new JComboBox(statuses);
37         //Text fields for property information and transaction id
38         private JTextField transactionField = new JTextField("");
39         private JTextField addressField = new JTextField("");
40         private JTextField bedroomsField = new JTextField("");
41         private JTextField squareField = new JTextField("");
```

```

42     private JTextField priceField = new JTextField("");
43     TreeMap<Integer, Property> propertyData = new TreeMap<>();
44     //Constructor for panel
45     public realEstateDataPanel() {
46         //Sets layout
47         setLayout(new GridLayout(7,2, 7,10));
48         this.add(transactionLabel); this.add(transactionField);
49         this.add(addressLabel); this.add(addressField);
50         this.add(bedroomsLabel); this.add(bedroomsField);
51         this.add(squareLabel); this.add(squareField);
52         this.add(priceLabel); this.add(priceField);
53
54         JButton processButton = new JButton(new AbstractAction("Process") {
55             @Override
56             public void actionPerformed(ActionEvent e) {
57                 String processOption = String.valueOf(dataList.getSelectedItem());
58                 try {
59                     switch (processOption) {
60                         case "Insert":
61                             checkForDuplicates(getTransactionId());
62                             propertyData.put(getTransactionId(), getPropertyInfo());
63                             JOptionPane.showMessageDialog(null, "Property successfully stored in database.");
64                             break;
65                         case "Delete":
66                             checkForExisting(getTransactionId());
67                             propertyData.remove(getTransactionId());
68                             JOptionPane.showMessageDialog(null, "Property successfully removed from database.");
69                             break;
70                         case "Find":
71                             checkForExisting(getTransactionId());
72                             Property getProperty = propertyData.get(getTransactionId());
73                             JOptionPane.showMessageDialog(null, getProperty.toString());
74                             break;
75                     }
76                 } catch (NumberFormatException nex) {
77                     JOptionPane.showMessageDialog(null, "Incorrect format for values entered.");

```

```

78         } catch(DuplicateProperty dex) {
79             JOptionPane.showMessageDialog(null, "Transaction id already exists in database.");
80         } catch(PropertyNotFound pex) {
81             JOptionPane.showMessageDialog(null, "Transaction id not found in database.");
82         }
83     }
84 });
85
86 JButton changeButton = new JButton(new AbstractAction("Change Status") {
87     @Override
88     public void actionPerformed(ActionEvent e) {
89         try {
90             Status statusOption = (Status) statusList.getSelectedItem();
91             checkforExisting(getTransactionId());
92             Property changeProperty = propertyData.get(getTransactionId());
93             changeProperty.changeState(statusOption);
94             propertyData.put(getTransactionId(), changeProperty);
95             JOptionPane.showMessageDialog(null, "Property status successfully changed in database");
96         } catch(PropertyNotFound pex) {
97             JOptionPane.showMessageDialog(null, "Transaction id not found in database.");
98         } catch(NumberFormatException nex) {
99             JOptionPane.showMessageDialog(null, "Incorrect format for values entered.");
100        }
101    }
102 });
103
104 this.add(processButton); this.add(dataList);
105 this.add(changeButton); this.add(statusList);
106 }
107 //reads the property info and throws an exception if incorrect information is entered
108 private Property getPropertyInfo() throws NumberFormatException {
109     String address = addressField.getText();
110     int bedrooms = getInput(bedroomsField);
111     int squareFt = getInput(squareField);
112     int price = getInput(priceField);
113     return new Property(address, bedrooms, squareFt, price);
114 }

```



```

115 //reads transaction number and throws and exception if it is the wrong format
116 private int getTransactionId() throws NumberFormatException {
117     return getInput(transactionField);
118 }
119 //verifies that property doesn't exist in database before being entered
120 private void checkForDuplicates(int transactionId) throws DuplicateProperty{
121     if(propertyData.containsKey(transactionId)) {
122         throw new DuplicateProperty();
123     }
124 }
125 //verifies that property exists before being deleted
126 private void checkForExisting(int transactionId) throws PropertyNotFound {
127     if(!propertyData.containsKey(transactionId)) {
128         throw new PropertyNotFound();
129     }
130 }
131 //Local method to convert entered Strings to integers
132 private int getInput(JTextField inputTextField) throws NumberFormatException {
133     String inputString = inputTextField.getText();
134     return Integer.parseInt(inputString);
135 }
136 //Defines custom exception if property already exists
137 private class DuplicateProperty extends Exception {
138     public DuplicateProperty() {
139         super();
140     }
141 }
142 //Defines custom exception if property does not exist
143 private class PropertyNotFound extends Exception {
144     public PropertyNotFound() {
145         super();
146     }
147 }
148 }
149 }

```