Project 4: Real Estate Database

CMIS 242

Attiqa Sheikh

May 7, 2020

Enumerated Type

```
/* Status.java
 * Developer: Attiqa 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: Attiqa 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: Attiqa 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: Attiqa 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);
```

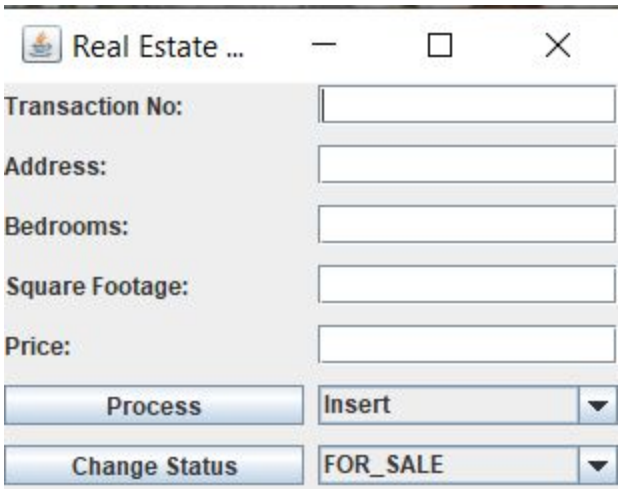
```
this.add(bedroomsLabel); this.add(bedroomsField);
this.add(squareLabel); this.add(squareField);
this.add(priceLabel); this.add(priceField);
        JButton processButton = new JButton(new AbstractAction("Process") {
        @Override
        public void actionPerformed(ActionEvent e) {
        String 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.");
```

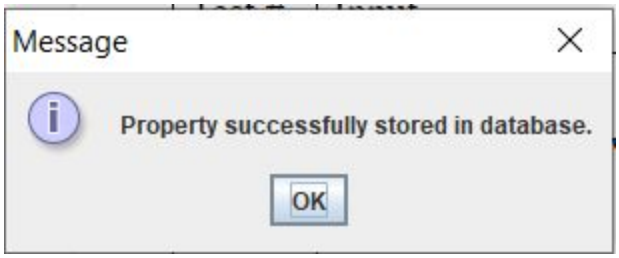
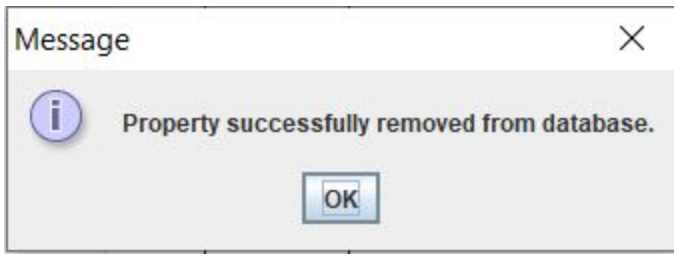
```
} catch(NumberFormatException nex) {
JOptionPane.showMessageDialog(null, "Incorrect format for values entered.");
}
}
});

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)) {
```

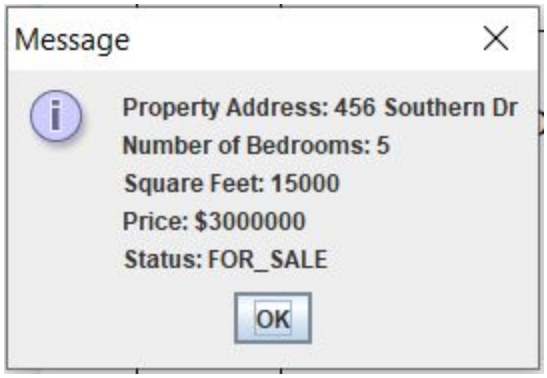
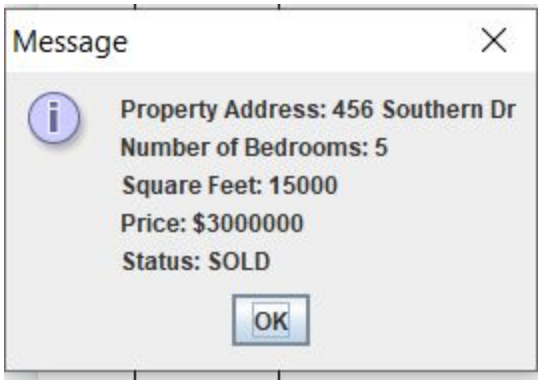
```
                        throw new DuplicateProperty();
        }
        }
        //verifies that property exists before being deleted
        private void checkforExisting(int transactionId) throws PropertyNotFound {
                if(!propertyData.containsKey(transactionId)) {
                        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(); }}}}
```

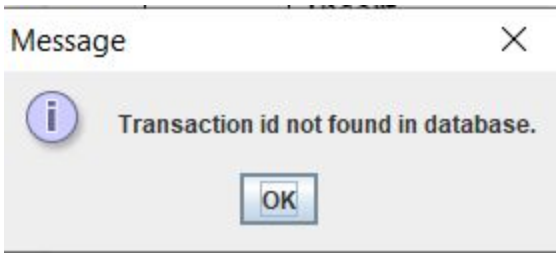
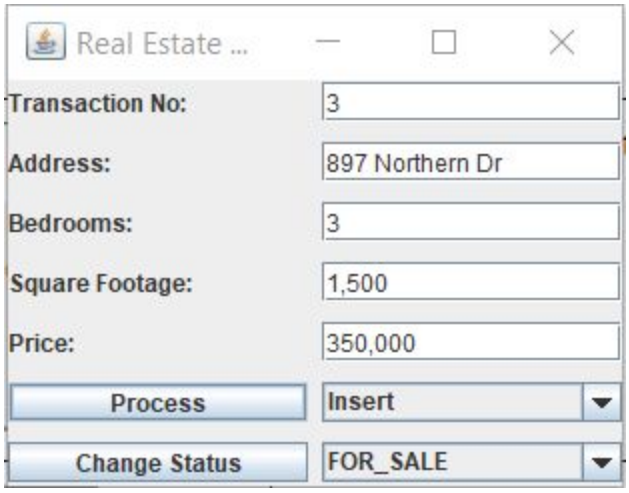
Generated GUI



Test Cases

Test #	Input	Expected Output	Notes
#1	1 123 American Blvd 3 5000 750000	Insert  Find  Remove  	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	FOR_SALE  UNDER_CONTRACT  SOLD  Output for Change Status  	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 × | Property.java | Status.java | Project4.java

```java
/*StateChangeable.java
 * Developer: Attiqa 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);
}
```

StateChangeable.java | Property.java | Status.java × | Project4.java

```java
/* Status.java
 * Developer: Attiqa Sheikh
 * Date: May 7, 2020
 * Purpose: enumerated type named status with three enumerated literals
 */
public enum Status {
    FOR_SALE, UNDER_CONTRACT, SOLD;
}

```

StateChangeable.java | Property.java × | Status.java | Project4.java

```java
/*Property.java
 * Developer: Attiqa 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);
        }
}

```

StateChangeable.java | Property.java | Status.java | Project4.java ×

```java
/*Project4.java
* Developer: Attiqa 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("");
```

```java
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.");
```

```java
                } 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);
                }
```

```java
                //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();
                }
                }
    }
}
```