```java
package project4;

import java.awt.Dialog;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.GregorianCalendar;

import javax.swing.DefaultListModel;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.ListSelectionModel;

/**************************************************************************
 * the GUI for the rental store, uses a jlist to update currently
 * rented dvds and can call save/load functions from RentalStore class
 * @author Gregory Huizenga
 * @version 7/26/2017
 *************************************************************************/
public class RentalStoreGUI extends JFrame implements ActionListener{

    // used for compatibility when saving / loading files
    private static final long serialVersionUID = 1L;

    /**the RentalStore used by the gui */
    private RentalStore rentalStore;

    /** the menu */
    private JMenuBar menu;

    /** the tabs on the menu */
    private JMenu file, action;

    /** the options available from the menu */
    private JMenuItem saveFile, saveText, loadFile, loadText, rentDVD,
                      rentBlueRay, rentGame, turnIn;

    /** the jlist used to display the rented dvds */
    private JList<DVD> rentalList;

    /*************************************************************
     * default constructor for gui
     ************************************************************/
    public RentalStoreGUI(){
        rentalStore = new RentalStore();
        rentalList = new JList<DVD>(rentalStore);
        menu = new JMenuBar();
```

```
58          file = new JMenu("file");
59          action = new JMenu("action");
60          saveFile = new JMenuItem("save (file)");
61          saveText = new JMenuItem("save (text)");
62          loadFile = new JMenuItem("load (file)");
63          loadText = new JMenuItem("load (text)");
64          rentDVD = new JMenuItem("rent DVD");
65          rentBlueRay = new JMenuItem("rent Blue Ray");
66          rentGame = new JMenuItem("rent Game");
67          turnIn = new JMenuItem("return");
68
69          menu.add(file);
70          menu.add(action);
71          file.add(saveFile);
72          file.add(loadFile);
73          file.add(saveText);
74          file.add(loadText);
75          action.add(rentDVD);
76          action.add(rentBlueRay);
77          action.add(rentGame);
78          action.add(turnIn);
79
80          // adds all actionlisteners
81          saveFile.addActionListener(this);
82          loadFile.addActionListener(this);
83          saveText.addActionListener(this);
84          loadText.addActionListener(this);
85          rentDVD.addActionListener(this);
86          rentBlueRay.addActionListener(this);
87          rentGame.addActionListener(this);
88          turnIn.addActionListener(this);
89
90          //adds jlist inside a scrollpane
91          rentalList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
92          JScrollPane scroll = new JScrollPane(rentalList);
93          scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
94
95          setJMenuBar(menu);
96          getContentPane().add(scroll);
97          setDefaultCloseOperation(EXIT_ON_CLOSE);
98          setSize(800, 500);
99          setTitle("Rental Store");
100         setVisible(true);
101
102
103
104     }
105
106     /**************************************************************
107      * main method used to create gui for user
108      * @param args
109      **************************************************************/
110     public static void main(String [] args){
111         RentalStoreGUI GUI = new RentalStoreGUI();
112
113     }
114
```

```
115
116
117     @Override
118     /*****************************************************************
119      * checks for which button was pressed and performs actions
120      *****************************************************************/
121     public void actionPerformed(ActionEvent e) {
122         int index = rentalList.getSelectedIndex();
123         GregorianCalendar temp3 = null;
124
125         // if return button was pressed
126         if (e.getSource() == turnIn){
127             if (index == -1 || index == rentalStore.getSize()){
128                 return;
129             }
130
131             try{
132
133             // converts provided string into a gregoriancalendar date
134             SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy");
135             String returnDate = JOptionPane.showInputDialog("Enter return date (format
    MM/dd/yyyy)", sdf.format(new Date()));
136             Date temp1 = sdf.parse(returnDate);
137             temp3 = new GregorianCalendar(temp1.getYear(), temp1.getMonth(), temp1.getDay());
138
139             }catch(ParseException ex){
140                 JOptionPane.showMessageDialog(null, "Invalid return date added");
141                 return;
142             }
143
144         //creates message dialog telling user the cost of the rental
145         String returnMessage = "Thank you for returning: " + ((DVD)
    rentalStore.getElementAt(index)).getTitle() + "\n";
146         returnMessage += ((DVD) rentalStore.getElementAt(index)).getCost(temp3);
147             JOptionPane.showMessageDialog(null, returnMessage);
148             rentalStore.remove(rentalList.getSelectedIndex());
149             rentalStore.update();
150         }
151
152         //if savefile button was pushed
153         if (e.getSource() == saveFile){
154             String fileName = JOptionPane.showInputDialog("Enter a name for the file to be
    saved to:");
155             if (fileName != null){
156                 rentalStore.saveToFile(fileName);
157             }
158         }
159
160         // if savetext button was pushed, currently unimplemented
161         if (e.getSource() == saveText){
162
163         }
164
165         //if loadfile button was pushed
166         if (e.getSource() == loadFile){
167             String fileName = JOptionPane.showInputDialog("Enter a name for the file to be
    loaded from:");
```

```
168             if (fileName != null){
169                 rentalStore.loadFromFile(fileName);
170                 rentalStore.update();
171             }
172         }
173 
174         //if loadtext button was pushed, currently unimplemented
175         if (e.getSource() == loadText){
176 
177         }
178 
179         //if rentdvd button was pushed
180         if (e.getSource() == rentDVD){
181 
182             //creates dialog prompting user to add fields for dvd
183             RentDVDDialog rent = new RentDVDDialog();
184             rentalStore.add(rent.getRentalDVD());
185             rentalStore.update();
186             rent.dispose();
187         }
188 
189         //if rentBluray button was pushed
190         if (e.getSource() == rentBlueRay){
191 
192             //creates dialog prompting user to add fields for bluray
193             RentDVDDialog rent = new RentDVDDialog();
194             rentalStore.add(rent.getRentalBlueRay());
195             rentalStore.update();
196             rent.dispose();
197         }
198 
199         //if rentgame button was pushed
200         if (e.getSource() == rentGame){
201 
202             //creates dialog prompting user to add fields for game
203             RentGameDialog rent = new RentGameDialog();
204             rentalStore.add(rent.returnGame());
205             rentalStore.update();
206             rent.dispose();
207         }
208 
209 
210 
211 
212         }
213 
214     }
215 
216 
217 
```