

RentalStore.java

```
1 package project4;
2
3 import java.io.FileInputStream;
4 import java.io.FileNotFoundException;
5 import java.io.FileOutputStream;
6 import java.io.IOException;
7 import java.io.ObjectInputStream;
8 import java.io.ObjectOutputStream;
9 import java.io.Serializable;
10 import java.util.ArrayList;
11
12 import javax.swing.AbstractListModel;
13
14 /*****
15  * rental store that holds the list of rented DVDs,
16  * used in RentalStoreGUI
17  * @author Gregory Huizenga
18  * @version 7/26/2017
19  *****/
20 public class RentalStore extends AbstractListModel implements Serializable{
21
22     /** the list containing the rented dvds */
23     private MyDoubleLinkedList<DVD> storeList;
24
25     /*****
26      * default constructor for RentalStore
27      *****/
28     public RentalStore(){
29         storeList = new MyDoubleLinkedList<DVD>();
30     }
31
32     /*****
33      * add method for dvds
34      * @param item the dvd to be added
35      *****/
36     public void add(DVD item){
37         storeList.add(item);
38     }
39
40     /*****
41      * add method for blue rays
42      * @param item the blue ray to be added
43      *****/
44     public void add(BlueRay item){
45         storeList.add(item);
46     }
47
48     /*****
49      * add method for games
50      * @param item the game to be added
51      *****/
52     public void add(Game item){
53         storeList.add(item);
54     }
55
56     /*****
57      * removes the item at the specified index from the list
```

RentalStore.java

```

58  * @param index the index of the item to be removed
59  * @return temp the element that was removed
60  *****/
61  public DVD remove(int index){
62      DVD temp = (DVD) storeList.get(index);
63      storeList.remove(index);
64      return temp;
65  }
66
67  @Override
68  /**
69   * returns the element stored at the given index
70   * @return index index of item
71   *****/
72  public Object getElementAt(int index) {
73      return storeList.get(index);
74  }
75
76  @Override
77  /**
78   * returns the current size of the list
79   * @return size size of list
80   *****/
81  public int getSize() {
82      return storeList.size();
83  }
84
85  /**
86   * updates the jlist in the gui
87   *****/
88  public void update(){
89      this.fireContentsChanged(this, 0, storeList.size() - 1);
90  }
91
92  /**
93   * saves the current list to a file with specified name
94   * @param filename the name to be saved to
95   *****/
96  public void saveToFile(String filename){
97      try {
98          FileOutputStream outputF = new FileOutputStream(filename);
99          ObjectOutputStream outputO = new ObjectOutputStream(outputF);
100
101          outputO.writeObject(storeList);
102          outputO.close();
103
104      } catch (FileNotFoundException e) {
105          e.printStackTrace();
106      } catch (IOException e) {
107          e.printStackTrace();
108      }
109  }
110
111  /**
112   * loads to the current list from a file with specified name
113   * @param filename the name to be loaded from
114   *****/

```

RentalStore.java

```
115     public void loadFromFile(String filename){
116         try {
117             FileInputStream inputF = new FileInputStream(filename);
118             ObjectInputStream inputO = new ObjectInputStream(inputF);
119
120             storeList = (MyDoubleLinkedList<DVD>) inputO.readObject();
121         } catch (IOException e) {
122             e.printStackTrace();
123         } catch (ClassNotFoundException e) {
124             e.printStackTrace();
125         }
126     }
127
128 }
129
```