**CIS 163, Fall 2016**

**Project 4 – A “RedBox-like” Blue Box program**

**Due Date**

* At the beginning of class/lab, see syllabus for exact day of your lab.

**Before Starting the Project**

* Review Chapters 8 - 10 and Chapters 12, 13, 15, 18 of the CIS163 book
* Read this entire project description before starting, if you have any question please ask the instructor

**Learning Objectives**

After completing this project you should be able to:

* Use inheritance and polymorphism
* Use advanced Swing components like JList and AbstractListModel.
* Use advanced Swing components like JTable and AbstractTableModel.
* Save and restore objects using the Serialization API
* Using simple Date and GregorianCalendar classes

**The made-up fictitious purpose, but often describes real life situations: Our company, the Blue box co. (we may get sued by Red box, we hope not) hired a person from a university (not GVSU) that shall not be named (we may get sued by the Harry Potter group for using that line, we hope not) started to write a game/DVD renting program. The person we hired completed some of the code and then, we fired that person for incompetence. We never thought about the consequences of our action, i.e., firing that person so abruptly…. We were left with imcomplete, half baked code. ☺**

**Your assignment** is to complete the half baked code, your code should simulate a DVD rental store. (The DVD class is the base class and the Game class extends the DVD class). As expected you can rent out a DVD and a Game with your program. The rental store program can also return items (DVD and Game) with a cost calculated based upon the number of days the unit was rented.

**A completed program must have the following functionality:**

* Save and load the rental database with serialized files using JFileChooser
* Save and load the rental database with TEXT files using JFileChooser
* Rent a DVD with a bought date, due date, title and a renters name
* Rent a Game with a bought date, due date, title, renters name and the player used (Xbox One, PS 4, …)
* Returns a DVD and generate a cost for the rental (see below fore details)
* Returns a Game and generate a cost for the rental (see below fore details)
* **This program will be demonstrated in class to show the full functionality of the program, so attending class is very important**.

*Before you turn in your work: use the* [*Java Style Guide*](http://www.cis.gvsu.edu/studentsupport/javaguide) *to document your project. (10 pts)*

*Steps 0 – 7 should be completed first (the ordering is a suggestion). Step 8 – 10 should be completed second (the ordering is a suggestion)*

**Step 0: Create an initial UML model (see rubric) of your proposal solution of your project. Use a tool like dia from the EOS lab to create your class diagram. *DUE: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.* A final UML model that represents your final solution of this project is due upon presentation.**

**Step 1: Create an Eclipse project named “RentalPrj”**

* Create a package named: project4
* Create a class named: DVD implements Serializable
* Create a class named: Game that extends DVD
* Create a class named: RentalStoreGUI extends JFrame implements ActionListener
* Create a classes named: RentDVDDialog extends JDialog
* Create a classes named RentGameDialog extends JDialog
* Create a class named: RentalStore extends AbstractListModel

**Step 2: Implement the DVD (base class) and using the following:**

*public class DVD implements Serializable {*

*private static final long serialVersionUID = 1L;*

*/\*\* The date the DVD was rented \*/*

*protected GregorianCalendar bought;*

*/\*\* The date the DVD is due back \*/*

*protected GregorianCalendar dueBack;*

*/\*\* The title of the DVD \*/*

*protected String title;*

*/\*\* The name of the person who is renting the DVD \*/*

*protected String nameOfRenter;*

*// add constructor*

*// add getter, setter methods*

**Step 3: Game is a derived class by extending DVD and using the following:**

*public class Game extends DVD {*

*/\*\* Represents the type of player \*/*

*private String player; // Xbox 360, PS4, Xbox720.*

*// add constructor*

*// add getter, setter methods*

**Step 4: Implement the class RentalStoreGUI using the following:**

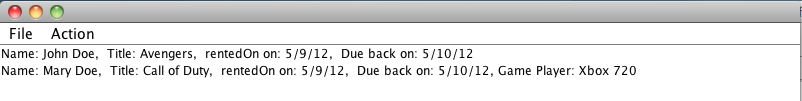
*public class RentalStoreGUI extends JFrame implements ActionListener{*

*// declare GUI components (menu items, buttons, etc.) needed*

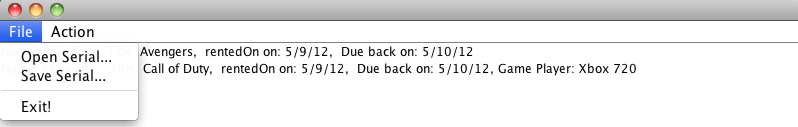
*// constructor method that prepares the GUI*

*// event handlers and other methods needed to build the GUI*

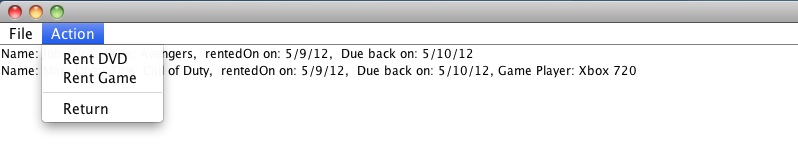
* The RentalStoreGUI class is the class that displays the GUI to the user and allows the user to rent a DVD (and a Game) and return the unit. In addition, the GUI allows users to save and load the database using serialized files. The RentalStoreGUI must handle the following operations shown below (and underlined). There are suggestions on where to place these operations in the GUI menu and these suggestions are in parenthesis. The first screen shot shows the main suggested GUI screen:



* (File Menu) have read and write operations to save or load the database to/from a Serializable file. You must use a JFileChooser to select the file during read/load and write/save operations.
* example:



* **(File Menu) have read and write operations to save or load the database to/from a Text file. You must use a JFileChooser to select the file during read/load and write/save operations. For**
* (Action Menu) have rent DVD and Game operations available and the return (a selected dvd in the list) operation under this menu. For example:



**Step 5: Implement the JDialog classes (RentDVDDialog, RentGameDialog)) using the following:**

These dialog boxes are invoked with the user selects Rent DVD or Rent Game (see above). The only difference between the two dialog boxes is the awareness that the RentGameDialog must ask for the ‘player’ type in addition to the other data collected. When a JDialog box appears, have today’s date in the JTextField for rented on, and have tomorrow’s date in the JTextfield for date due. The following is some coding help with the GregorianCalendar and Date classes.

*Date date = Calendar.getInstance().getTime(); // Today’s date*

Here is a sample screen shot of a DialogRentGame:



To create a dialog box the following code is suggested. Note: The following code is just a start; to fully understand how to create dialog boxes, more details are needed. Examples of other dialog boxes will be presented in class.

*public class RentDVDDialog extends JDialog implements ActionListener{*

*private JTextField titleTxt;*

*private JTextField renterTxt;*

*private JTextField rentedOnTxt;*

*private JTextField DueBackTxt;*

*private JButton okButton;*

*private JButton cancelButton;*

*private boolean closeStatus;*

*private DVD unit;*

*public RentDVDDialog(JFrame parent, DVD d) {*

*unit = d;*

*...*

*}*

*public void actionPerformed(ActionEvent e) {*

*...*

*}*

Finally, to invoke this dialog box from the RentalStoreGUI class, the following code is used.

*if (rentGameItem == e.getSource()) {*

*Game game = new Game();*

*RentGameDialog dialog = new RentGameDialog(this, game);*

*...*

*}*

**Step 6: Implement the class RentalStore using the following:**

This class is used for storing the rental units (DVD and Game) into an LinkedList<DVD>. (Note: DVD is the base class and review chapter 9 of your book). The functionality of this class is similar in concept to code presented in chapter 9, specifically, the staffList array. The main difference is that this class must handle all the operations from the GUI class. That is, rent a DVD, rent a Game, return a rental, store, load, etc. Note: The following code is just a start; to fully understand how to create the RentalStore, more details are needed. Examples of RentalStore classes will be presented in class.

*public class RentalStore extends AbstractListModel {*

*private LinkedList<DVD> listDVDs;*

*// constructor method that initializes the Linkedlist*

*// override these two methods from AbstractListModel class*

*public Object getElementAt(int arg0) {*

*...*

*}*

*public int getSize() {*

*...*

*}*

*// add methods to add, delete, and update.*

*// add methods to load/save accounts from/to a binary file*

*// add other methods as needed*

Notes regarding the RentalStore class:

1. To make updates to the DVDs in the model immediately visible in the JList on your GUI, it is important that the methods in the RentalStore class that modify (add, delete, and update) the DVDs notify the JList immediately after any changes. These notifications can be sent from RentalStore class using one of these methods: fireIntervalAdded(), fireIntervalRemoved(), and fireContentsChanged(). The RentalStore class inherits these methods from the AbstractListModel class.

2. To save and load from a serialized file is not shown in this document, however, a demonstration of this ability will be done in class and code will be provided.

**Step 7: Implementing the return (a selected DVD in the list) function:** *Please allow the user to enter in the return date so I can test your program (8ts).*

* When returning a DVD using the following to calculate cost. For DVDs the cost is 1.2 dollars if returned on or before the due date, and 2 dollar charge if it is late. For Game the cost is 5 dollars if returned on or before the due date, and 10 dollar charge if it is late. Use a JOptionPane.showMessageDialog to output the cost.
  + Important: to accomplish this, create n method in the DVD base class called: pubic double getCost(….) which returns the cost of that DVD unit. This method will be polymorphic since you will have to override this method in the Game subclasses. You can call this method from the GUI class if you wish (or the RentalStore class).

Example output:



*----------- Do not start step 8 until the above is completed ----------*

**Step 8: Change the DVD class and using the following (**property player is an enum not a String**):**

*public class Game extends DVD {*

*/\*\* Represents the type of player \*/*

*private PlayerType player; // Xbox 360, PS4, Xbox720.*

*// add constructor*

*// add getter, setter methods*

(Note: PlayerType is an enum class with values of: *Xbox360, PS4, XboxOne, PS3, Xbox720*

Create a new game dialog box with a dropDown Combo box for player type. Here is a sample screen shot of a RentGameDialog:



**Enum type**

**(Type of Player would be nice to use a dropDown (Combo) box)**

To get access to different types with in a enum type, the following piece of code may help you:

*// Create the correct size array*

*PlayerType[] gameNames = new PlayerType[PlayerType.values().length];*

*for (PlayerType p: PlayerType.values()) // select each value;*

*gameNames[i++] = p;*

Other commands that may help (also, good examples found on google):

*PlayerType p = PlayerType.valueOf(“PS4”); // converts String to PlayerType*

*String temp = PlayerType.PS4.toString(); // converts PlayerType to String*

**Step 9: Complete and comprehensive error checking….**

]

**Step 10: Add on the following functionality to RentalStore class.**

Looking at the RentalStore class properties, you will see:  *“private LinkedList<DVD> listDVDs;”.* Your task in this step is to convert LinkedList class into your own MyDoubleLinkList class that you create: *private MyDoubleLinkedList<DVD> listDVDs*. Important, MyDoubleLinkedList has a DNode, and DNode have next and previous pointers (references).

There several methods that you will need to create so that the RentalStore class will again function correctly with your new class. You must create the following methods:

*public class MyDoubleLinkedList<E> implements Serializable*

*{*

*private DNode<E> top; //* ***DNode is a double linked list node.***

*private DNode<E> tail;*

*public int size;*

*public MyLinkedList()*

*public int size()*

*public void clear ()*

*public void add(E s) // add at the end.*

*Public void addFirst (E s) // add at the top.*

*public E remove(int index) // remove first occurrence.*

*public E get(int index)*

*public boolean removeAll(E) // return true if at least one item is removed*

*public int find (E s) // return -1 if not found, otherwise return index*

***Extra credit worth 15 pts on project… Implement:***

***public class MyDoubleLinkedList<E> implements Serializable, List***

***use Collections.Sort to sort the linked list.***

**Step 11: Convert Jlist to a JTable and use a AbstractTableModel (not a AbstractListModel)**

***Extra credit worth 15 pts on project… Implement:***

***public class MyDoubleLinkedList<E> implements Serializable, List***

***use Collections.Sort to sort the linked list.***

--------------------------- YOU’RE DONE ☺ -------------------------------

**Some additional grading criteria**

There is a 70% penalty on programming projects if your solution does not compile.

* Stapled cover with your name and signed pledge. (-5 pts if missing)

**Late Policy**

Projects are due at the START of the class period and NO late submission this time, since this is the last project.

**Turn In**

A professional document is stapled with an attractive cover page.

* Cover page - Your project must have a cover page that includes your name, a title, an interesting graphic or photograph related to the project topic and the following signed pledge: "I pledge that this work is entirely mine, and mine alone (except for any code provided by my instructor). " You are responsible for understanding and adhering to the [School of CIS Guidelines for Academic Honesty](http://www.cis.gvsu.edu/Academics/Honesty/).

**CIS 163 – Computer Science II**

**Project 4: “Blue Box” Rental Program**

|  |  |
| --- | --- |
| Student Name |  |
| Date Submitted, Days Late, Late Penalty |  |

|  |  |  |
| --- | --- | --- |
| **Graded Item** | **Pts** | **Points Secured / Comments** |
| Javadoc Comments and Coding Style/Technique  (<http://www.cis.gvsu.edu/studentsupport/javaguide>)   * Code Indentation (auto format source code in IDE) * Naming Conventions (see Java style guide) * Proper access modifiers for fields and methods * Use of helper (private) methods * Using good variable names * Header/class comments * Every method uses @param and @return (1 sentence after) * Every method uses a /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* separator * Overall layout, readability, No text wrap * Using /\*\* … / for each Instance variable * Has many inner “inner” comments | 10 |  |
| **Steps 1 – 7: Basic Functionality**   * DVD and Game classes * RentalStore class * RentalStoreGUI class * RentDVDDialog and RentGameDialog classes * Rent DVD (input via RentDVDDialog) * Rent Game (input via RentGameDialog) * Return DVD/Game * Menu item to save items in store as a serialized file * Menu item to load items from a serialized file | 50 |  |
| **Step 8 – 10:**   * Step 8: Enum * Complete error checking, MISC too * Step 10: Linked list (double, and top,tail) | 4  9  15 |  |
| **Step 11:**   * Convert JList to JTable * Convert | 12 |  |
| ***Extra credit worth 15 pts on project… Implement:*** *public class MyDoubleLinkedList<E> implements Serializable, List*  *use Collections.Sort to sort the linked list.* |  | ***This is lots of work….*** |

**Comments: (extra credit)**