Project 3 – A "RedBox-like" program

Due Date

- At the beginning of the lab; see the schedule, last page of the syllabus

Before Starting the Project

- Review Chapters 8 - 10, 12, and 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`
- Save and restore objects using the Serialization API
- Using simple Date and GregorianCalendar classes

Program description: Your assignment is to create a program that simulates 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
- Rent a DVD with a bought date, due date, title and a renter's name
- Rent a Game with a bought date, due date, title, renters name and the player used (Xbox, PS4, …)
- Returns a DVD and generate a cost for the rental (see Step 7 for details)
- Returns a Game and generate a cost for the rental (see Step 7 for 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** 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. You do not have to include any "uses" relationships in your diagram; focus on "has-a" and "is-a" relationships. Use a tool like dia from the EOS lab to create your class diagram. *DUE: Monday, 6/25.* A final UML model that represents your final solution of this project is due upon presentation of Project 4.

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 class named: RentDVDDialog extends JDialog
- Create a class 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 PlayerType player;   // Xbox360, PS4, Xbox720, PS3, GameCube

   // add constructor
   // add getter, setter methods
```

Note: PlayerType is an enum class with values of: *Xbox360, XBox1, PS4, WiiU, NintendoSwitch*

Some commands that may help as you develop your project (in addition to info found on Google):

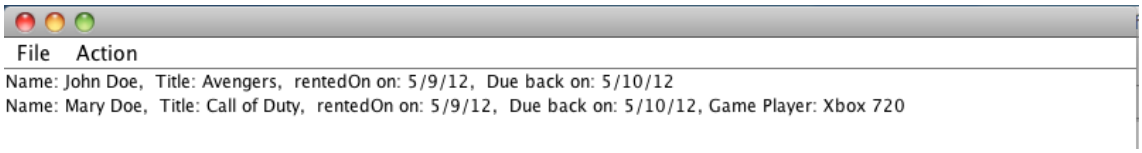
```
PlayerType p = PlayerType.valueOf("PS4"); // converts String to PlayerType
String temp = PlayerType.PS4.toString();  // converts PlayerType to String
```

Step 4: Implement the class RentalStoreGUI using the following:

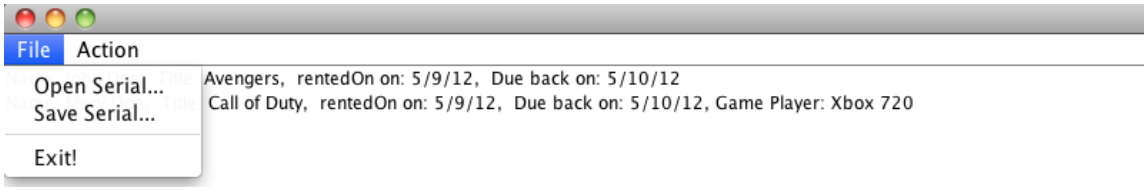
```
public class RentalStoreGUI extends JFrame implements ActionListener{

   // declare GUI components (menu items, buttons, etc.) needed
   // constructor method that prepares the GUI
   // event handlers and other methods needed to build the GUI
```

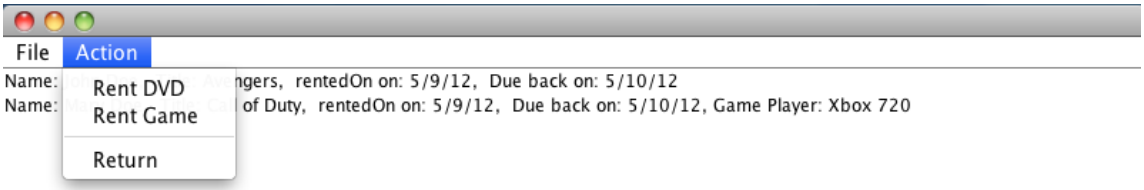
- The RentalStoreGUI class is the class that displays the GUI to the user and allows the user to rent a DVD (and a Game) and return the unit. In addition, the GUI allows users to save and load the database using serialized files. The RentalStoreGUI must handle the following operations shown below (and 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. For example:



- (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 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 RentGameDialog:

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 custom dialog boxes are provided in the skeleton code.

```
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()) {
      DVD dvd = new DVD();
      RentDVDDialog dialog = new RentDVDDialog(this, dvd);
      ...
      }
```

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. An example of a RentalStore class is provided in the skeleton code.

```
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 or Game in the list) function:

Note: Please allow the user to enter the return date when returning a DVD or Game so that your program can be thoroughly tested. Do NOT assume the return date by hard coding values or using the current date.

- When returning a DVD using the following to calculate cost. For DVDs the cost is $1.20 if returned on or before the due date, with an additional $2.00 charge if it is late. For Game the cost is $5.00 if returned on or before the due date, with an additional $10.00 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:

Step 8: TOTALLY error checked your program. For example: Due Date was before rented date; improper date such as: "abc/abc/abc"; etc.

---------- Do not start Step 9 until the above is completed ----------

Step 9: Add on the following functionality to the GUI (modify other classes as needed).

In the Action menu create new JMenuItem that when selected opens another custom JDialog that accepts a Date. Then, only the units that would be considered late if returned on the given Date and the number of days late are shown. For example, if the title of the rental is "Hunger Games" and it is due back on 6/20/2018 and you enter in 6/22/2018 (in the JDialog box) then, this used would be displayed with 2 days late. How you display the list is up to the group except that you CANNOT use System.out.println().

CIS 163 – Computer Science II
Project 3: "RedBox" 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/java-coding-style-guide/) • 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 – 8: Basic Functionality • Initial UML class diagram • DVD and Game, RentalStore, RentalStoreGUI 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	5 20 20 10 4 6	
Step 9: • Step 9: search	10	
Total	**100**	

Comments: (extra credit)