MIST 4600. Spring 2014. Dr. Janine E. Aronson. Assignment I

As always, your initial file is LastnameFirstname-Assignment*. Download the zipped BlueJ Project start folder, unzip it, and rename the unzipped folder with your Lastname and Firstname. When you finish this assignment, put your name and a comment about what works and what does not at the top of the main class where indicated. Save the BlueJ Project, close the BlueJ Project, zip the folder properly and submit the zipped file properly to eLC Assignment *. Note that if after you submit your file, you may submit an update until the due date. eLC will retain your most recent submission. Note that if the testers do not work, then you earn 0. Your code must interface properly with the testers.

I suggest that before you touch the keyboard, you look at and understand the existing code, and think about the task and how to accomplish it. Also, as soon as you make one or two changes to the code, recompile it.

Add Video Games to the Movie Management System, by creating an inheritance structure. Movie and Game will inherit both fields and methods from Item. Each will have some unique fields and methods. In this Assignment, we will not worry much about methods but will need, in each class (Item, Movie and Game), to have access (get) methods for each field in the class.

Start with the LastnameFirstname-AssignmentI project, from eLC Programming Components. From within the BlueJ workbench, rename your folder to use your own Lastname and Firstname. Make sure to include your name and a Code Comment in the Store class, and the other class(es) in which you do your work. Properly zip and submit your BlueJ project folder to eLC Assignment I.

Scenario:

Start with the LastnameFirstname-AssignmentI BlueJ project, properly renamed. In the initial Java program, there are three classes: a Movie class, a ReadFile class, and a Store class. In the Store class, there is a method to read values from a file and use them to fill an ArrayList (called movies) of objects of the Movie class. There is also a listAllItems method, that prints header information, and then uses a loop to access each movie in the ArrayList and, for each, calls a method in the Movie class to print out part of that movie's information.

Your job is to modify this project to incorporate video Game objects as well as Movie objects. This will involve creating an Item class from which both the existing Movie class and a new Game class will inherit. Class Item will have all fields that are common to both Class Movie and Class Game. Game will be somewhat different from Movie: a Game object should have no "director" or "timesRented", and instead should have a "platform" such as Sega, Gameboy, etc. Other than that, the fields for Game are the same as the fields for Movie, and of course all common fields should be moved to Item, not remaining in Movie or Game.

To see where we are going with this, ultimately most of the code having to do with renting out a Movie or a Game will reside in Item. But that is for another assignment!

In Store, you will need to read in both movie and video game data (each from its own file), creating an appropriate object for each movie or video game, and putting all movie and game objects into a single ArrayList called **items**. (We will no longer use the movies ArrayList in Store.)

You will also need to make sure you can print out the info for each using the listAllInfo method. However, please move the print method to Item since it is almost the same for Movie and Game. This does mean you will not be printing out any info that is unique to Movie (such as director) or to Game (such as platform). We will address that issue later.

It is suggested that you work in the following order, though that is up to you:

1. **Add an Item class and a Game class.** Decide which fields of the Movie Class are common with fields in the Game class and therefore should be included in Item. For the Item class, I suggest starting by copying the code from the Movie class, then pairing it down to what will be needed and common for both the Movie and the new Game class. Then add the new Game class. Again, I suggest starting by copying the code from the Movie class, then paring it down to what is needed for the Game class. Then pare down the code in Movie so there is no duplication with Item. Decide what will happen in the constructor for Movie, for Game, and for Item. (This will involve some calls to the super class constructor for Movie and Game.) Make all those changes. As far as methods, to start with, you might copy all methods from Movie into Game as well. Then decide which of these methods belongs in Item, and which needs to stay in Movie and/or Game. There may be unique methods in Movie or Game. For the print method, for now, put it in Item, recognizing that it may not be able to print any of the unique attributes/fields for each Movie and Game object. Don't worry about that yet.

2. **Create an items ArrayList in Store.** In the Store class, make the ArrayList a collection of Item objects (not Movie objects). (For example, call it **items**, not **movies** and indicate that it holds Item objects, not Game or Movie objects.) Modify the readFromMovieFile() method that reads in movie info so that it now creates Movie objects and puts them into the **items** ArrayList. Add a new method (readFromGameFile()) to read in game info, create Game objects, and put them into the **items** ArrayList. You can do this by copying the readFromMovieFile() method you just modified, and modify readFromGameFile() to read in game data from a file called GamesData.txt. [The attributes/fields are: ID, Name, Platform, price, renterID, dueDate.] Then create Game objects with those values, and put them into the items ArrayList.

3. **Test your program well.** You should be able to read all the movie and game info in, and then list all movies and games by cycling through all objects in the Items ArrayList, using the print() method for each of them.

4. **Submit your work.** Properly name, zip, and submit your project folder to the eLC Assignment H.