Programming Assignment 3

Due: M Mar 9 @ 11:59PM CST

This project re-factors your previous code into a Model-View-Controller (MVC) architecture and requires you to use some object-oriented programming. You have already written most of the code for this assignment. The challenge for this project is to put your existing code in the appropriate place.

Section I: Application Design

You will be given a skeleton program organized as an MVC[1]. The view and controller are complete; you must complete the model. This section gives an overview of the MVC design and briefly describes *what* modifications you must make to complete the application. Section II provides more details on *how* to make those modifications.

The application's basic framework consists of a module called `movielib`. This module contains three submodules: `model`, `view`, and `controller`. Each submodule contains its own submodules that provide the basic functionality described below. You should not modify any files in `movielib`. You should instead extend this functionality, as described in Section II. The last page of this document contains a diagram that summarizes the application's module, file, and class relationships.

The Model

The model stores and manipulates the DVD library information. It consists of five basic things:

> **Movies.** A movie is implemented as a class that contains **attributes** (e.g., title, genres, etc.) for a particular film. The movie class provides a dictionary interface where each attribute name corresponds to a dictionary key. You will be provided with a basic `Movie` class. Your job will be to inherit from this class to provide a way to display a movie.
>
> **A schema.** The schema describes *which* attributes a movie maintains. You will not need to modify the schema.
>
> **A database.** A database contains information about lots of movies. It is implemented as a dictionary-like class that maps UPCs to movies. You will be provided with a basic `Database` class. Your job will be to inherit from this class to provide search and sort behaviors.
>
> **A storage manager.** The storage manager shuffles data between disk and memory. You will be provided with an abstract storage manager class. Your job will be to inherit from the abstract storage manager to define a text-based manager.
>
> **Errors.** The model contains a custom exception that is raised whenever the model encounters an error.

The model exists on its own and is not coupled to either the view or the controller.

The View

The view displays the database on a screen. You will be provided with a fully working view. The view exists on its own and is not coupled to either the model or the controller.

The Controller

The controller is the glue that binds the model and the view together. It has direct access to and knowledge of both the model and the view. The controller is responsible for populating the model, creating a view, getting and validating user input, detecting errors, and using the view to display the data. You will be provided with a fully working view.

Section II: Completing the Application

This section describes the basic design in more detail and lists the steps you must take to complete the application. **Download the project files** and unzip them. The project consists of the following files:

> **`MyMovie.py`.** You will modify this file to implement display behavior for movies.
>
> **`MovieText.py`.** You will modify this file to implement the text-based storage manager.
>
> **`MyDatabase.py`.** You will modify this file to implement database searching and sorting.

---

[1] The design of this application is more heavy-handed than is strictly necessary. However, the design is indicative of more complex, real-world applications, and it provides good practice for object-oriented programming in Python.

`pa3.py`. This is the main program. You will not need to modify this file.

`movielib`. This directory contains the skeleton application. You will not need to modify these files, but you may want to take a look at them.

`movies.txt`. The text data for the movie database. You will not need to modify it. The program loads data from this file using your `MovieText` implementation.

Your assignment is to complete the model. The model consists of the files in module `movielib.model`, plus files you must complete: `MyDatabase.py`, `MyMovie.py`, and `MovieText.py`.

The file `MyDatabase.py` contains a class `MyDatabase` that inherits from `movielib.model.Database.Database`. The `Database` class provides dictionary-like behavior for your library. The class defines the following methods:

`loadMovies`. This method uses a storage manager to load movies. You will not need to override this method.

`saveMovies`. This method uses a storage manager to save movies. You will not need to override this method.

`populate`. This method takes a list of `Movie` instances and maps each movie's `upc` attribute to the instance. You will not need to override this method.

`newMovie`. This method creates a new movie instance. You will not need to override this method.

`simpleSearch`. This method takes an attribute name (as a string) and a search string. The method returns a list of movies whose given attribute contains the search string. You must override this method in `MyDatabase` to provide the correct functionality.

`regexSearch`. This method takes an attribute name (as a string) and a regular expresion. The method returns a list of movies whose given attribute contains a substring that matches the regular expression. You must override this method in `MyDatabase` to provide the correct functionality.

`sort`. This method returns a list of movies in the database, sorted by UPC. You must override this method in `MyDatabase` to provide the correct functionality. If you chose to do the sort-by-attribute extra, then the method call can specify the attribute name by which to sort and return a list of movies sorted by that attribute name.

The file `MyMovie.py` contains an empty class `Movie`. You must define an `__str__` method in this class, which should return a string that formats the movie's attributes as described in the previous assignment. You should be able to reuse almost all your code from the previous assignment.

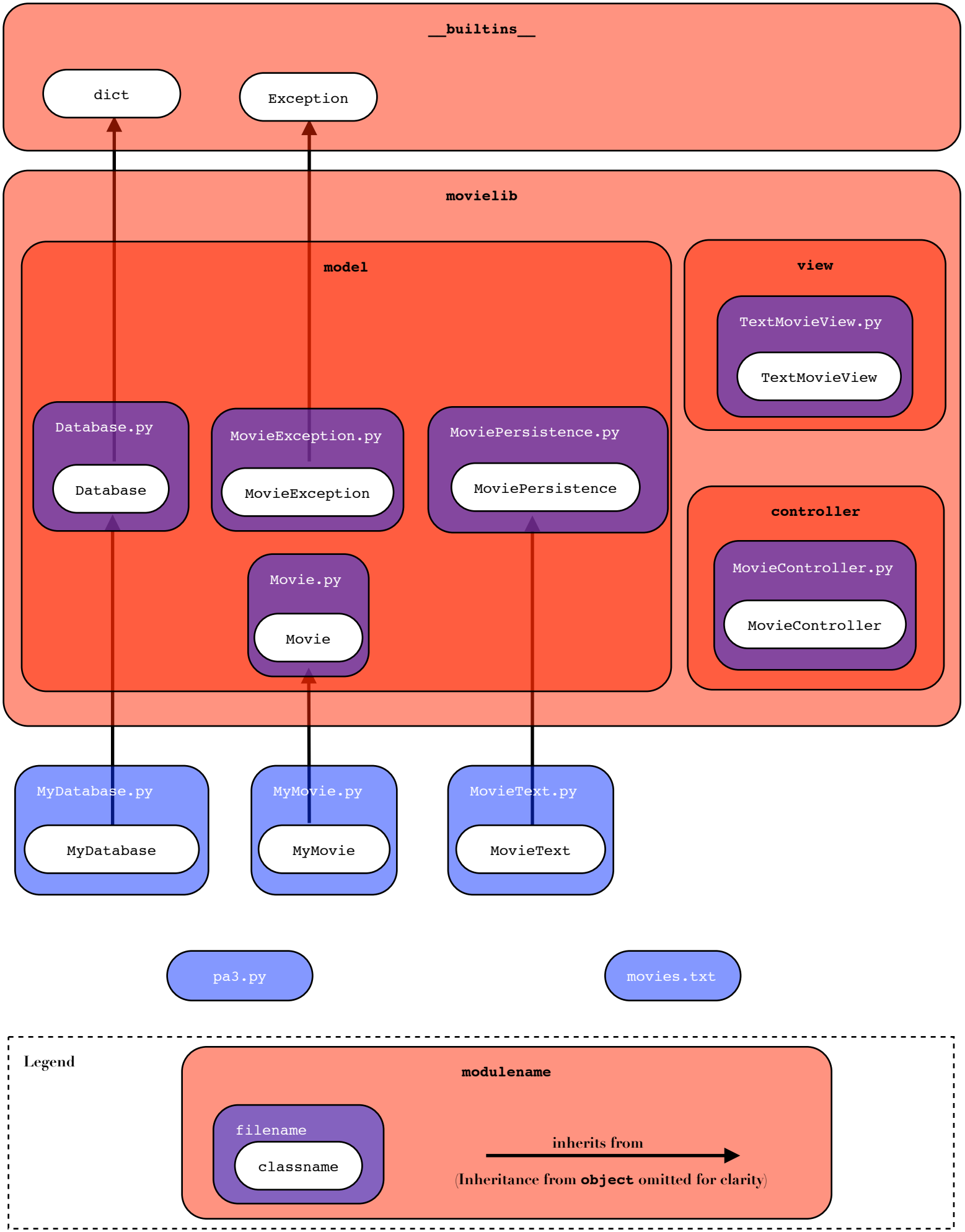
The file `MovieText.py` contains a class `MovieText` that inherits from `movielib.model.MoviePersistence.MoviePersistence`. The `MoviePersistence` class loads movie data from and saves movie data to disk. The class constructor takes a file name from which to load and/or save movies. The class defines the following methods:

`loadMovies`. This method loads the movie data from disk. The method must convert the disk format to a list of `MyMovie` instances and return this list. You must override this method in `MovieText` to convert the text file to a list of `MyMovie` instances. You should be able to reuse almost all of your code from the previous assignment.

`saveMovies`. This method takes a list of `Movie` instances and saves them to disk. You must override this method in `MovieText` to convert a list of `Movie` instances to file format and save the file to disk. You should be able to reuse almost all of your code from the previous assignment.

What To Turn In

Turn in all the files you modified. If you don't do any extras these files are: `MyDatabase.py`, `MyMovie.py`, and `MovieText.py`. You must also turn in a project report in the file `report.txt`. Each submitted file should have both team members' names and EIDs in a comment at the top of the file. Be sure to read the **Programming Assignment Overview** for instructions on how to prepare your report and submit your work. The *`homework-name`* for this assignment is: `pa3`.


__builtins__
dict
Exception
movielib
model
view
TextMovieView.py
TextMovieView
Database.py
Database
MovieException.py
MovieException
MoviePersistence.py
MoviePersistence
Movie.py
Movie
controller
MovieController.py
MovieController
MyDatabase.py
MyDatabase
MyMovie.py
MyMovie
MovieText.py
MovieText
pa3.py
movies.txt
Legend
modulename
filename
classname
inherits from
(Inheritance from object omitted for clarity)