Movie Store

1 Introduction

This assignment is about polymorphism. You will refactor an existing system written in a non-object-oriented style to use polymorphic classes.

2 Problem Description

A movie store sells comedies and dramas and wants to expand to sell sci-fi and action movies. You've been hired to update their shopping cart software to accomodate the new genres. The existing system is written in a non-object-oriented style: adding the new genres will require updates to classes that have nothing to do with movie genres. You endeavor to apply the lessons you have learned in this course to refactor the shopping cart system to use polymorphic movie classes instead of conditional logic and genre tags.

3 Solution Description

Updating the existing software to an object-oriented design will require the following changes:

- Make the `Movie` class abstract and remove its use of the `Genre` enum.
- Add an abstract `public int getPrice()` method to `Movie`.
- Make the following concrete subclasses of `Movie` to represent the genres: `ComedyMovie`, `DramaMovie`, `ScifiMovie`, and `ActionMovie`.
  - Comedy movies are priced at $3.
  - Drama movies are priced at $4.
  - Sci-fi movies are priced at $5.
  - Action movies are priced at $2.
- Update the `Cart` class's `totalPrice()` method to use the polymorphic classes listed above, removing the smelly `switch` statement.
- Update the main method in `Shopper` to use the polymorphic classes listed above and add examples of each genre to the cart.

In addition to the changes above, add a `toString` method to your `Movie` class(es) that returns the name of the movie and an `equals` method that considers two movies equal if they have the same name, regardless of genre. These methods must properly override the corresponding methods in `java.lang.Object`. Do not write more `toString` or `equals` methods than you need.

Here's the output of running the `Shopper` class as provided:

```
$ java Shopper
Movie                          Price
Movie@677327b6                $    3
Movie@14ae5a5                 $    3
Movie@7f31245a                $    4
Total:                        $   10
```

Here's how the output of running `Shopper` might look after making your changes:

```
$ java Shopper
Movie                          Price
Super Troopers                $    3
Office Space                  $    3
Fight Club                    $    4
The Matrix                    $    5
True Lies                     $    2
Total:                        $   17
```

4 Tips

- You won't need the `Genre` enum after you make your changes.
- You will make several small classes.
- Pay attention to the specifications of methods you're to override form `java.lang.Object`. Don't write more overriding methods than you need.

5 Checkstyle

You must run checkstyle on your submission. The checkstyle cap for this assignment is **10** points. Review the Style Guide and download the Checkstyle jar. Run Checkstyle on your code like so:

```
$ java -jar checkstyle-6.2.1.jar *.java
Audit done. Errors (potential points off):
0
```

The message above means there were no Checkstyle errors. If you had any errors, they would show up above this message, and the number at the end would be the points we would take off. The Java source files we provide contain no Checkstyle errors. For this assignment, there will be a maximum of **10** points lost due to Checkstyle errors (1 point per error). In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

6 Using the Submission Tool

Included with this timed lab is a handy-dandy submission tool. You will not submit to T-Square, but will use this instead. The commands are:

- `ant` or `ant compile` compiles your Java source files.
- `ant test` runs the provided JUnit tests. For this timed lab, the output will be your grade. Of course, we reserve the right to make changes to the grading rubric if necessary, but if you pass these tests, you've done it correctly.
- `ant checkstyle` is a convenient way of running checkstyle without typing that obnoxious command.
- `ant run` will run the code.
- `ant submit` will actually submit your code to the GT Github repo. This will prompt you for your user name and password each time, but you can run it as many times as you want if you want to keep resubmitting as you make minor changes.

7 Confirm Your Submission

Your submission was pushed to a git repository on github.gatech.edu. After running `ant submit`, a new browser window should open at the repository, at https://github.gatech.edu/[your-username]/[your-username]-timedlab2