P09 Movie Catalog

Overview

This assignment involves the implementation of a simple Movie Catalog using Binary Search Trees (BST). A movie catalog is usually used for streaming service and review websites. It is mainly dedicated to storing, retrieving, and making movie information available to the audiences. Our BST will store a set of movies where the lookup key information is the year of production and the rate from reviewers. You are going to learn how BSTs can be used to facilitate insertion and retrieval operations to and from a collection of ordered elements, in an easy and elegant way.

Grading Rubric

5 points	**Pre-Assignment Quiz:** The P09 pre-assignment quiz is accessible through Canvas before having access to this specification by **11:59PM on Sunday 04/18/2021**. Access to the pre-assignment quiz will be unavailable passing its deadline.
20 points	**Immediate Automated Tests:** Upon submission of your assignment to Gradescope, you will receive feedback from automated grading tests about whether specific parts of your submission conform to this write-up specification. If these tests detect problems in your code, they will attempt to give you some feedback about the kind of defect that they noticed. Note that passing all of these tests does NOT mean your program is otherwise correct. To become more confident in this, you should run additional tests of your own.
20 points	**Additional Automated Tests:** When your final grade feedback appears on Gradescope, it will include the feedback from these additional automated grading tests which will be run after the HARD DEADLINE of 11:59PM on April 22nd.
5 points	**Manual Grading Feedback:** After the deadline for an assignment has passed, the course staff will begin manually grading your submission. We will focus on looking at your algorithms, use of programming constructs, and the style and readability of your code. This grading usually takes about a week from the hard deadline, after which you will find feedback on Gradescope.

Learning Objectives

The goals of this assignment include:

- Implement common Binary Search Tree (BST) operations.
- Gain more Practice in recursive problem-solving.
- Gain more experience with developing unit tests.

Assignment Requirements and Reminders

- **BEFORE STARTING the implementation of the MovieTree's methods, MAKE SURE to define your test scenarios and IMPLEMENT the TEST methods FIRST. Your tester methods should help you assessing the correctness of your implementation, and locate defects if any. It MUST be also able to detect bugs in any broken implementation (not necessarily yours).**
- **DO NOT submit** the provided `Movie.java` and `BSTNode.java` source files on Gradescope and DO NOT make any change to their implementations.
- You ARE NOT allowed to add any additional import statements to the provided source files except `java.util.ArrayList` and `java.util.NoSuchElementException` classes.
- You ARE NOT allowed to add any fields either instance or static, and any public methods either static or instance to your `MovieTree` class.
- You CAN define local variables that you may need to implement the methods defined in this program.
- You CAN define private methods to help implement the different public methods defined in this programming assignment, if needed.
- ALL your test methods MUST be implemented in your `MovieTreeTester` class.
- In addition to the required test methods, we HIGHLY recommend (not require) that you develop your additional own unit tests (**public static methods that return a boolean**).
- Ensure that your code for every assignment is styled in conformance to CS300 Course Style Guide.
- You have adhered to the Academic Conduct Expectations and Advice. If you're feeling the strain of this incredibly difficult Spring 2021 semester and panicking, please don't hesitate to contact your instructor. We understand, we're feeling it too. We want you to succeed in this course, but not at the expense of your mental or physical health, or your academic integrity. We can always help you work something out.

1 Getting Started

Start by creating a new Java Project in eclipse. You may call it P09 Movie Catalog, for instance or any other name at your convenience. As the previous assignments, make sure that your new project uses Java 11, by setting the "Use an execution environment JRE:" drop down setting to "JavaSE-11.0.X" within the new Java Project dialog box. Then, download and add these provided Movie.java, and BSTNode.java source files to your project.

Read carefully through the provided source codes details. The `Movie` class represents a single movie in our catalog. For simplicity, we consider only three data fields in that class (the year of production, the review rating and the name). We are going to use two of them (year and rating) as lookup keys in our Movie Catalog. Notice also that the `Movie` class implements the `java.util.Comparable` interface. You can call the instance method `.compareTo()` to compare two movies with respect to their three instance fields.

The provided generic `BSTNode` class models the binary nodes which will be used to build and implement our Binary Search Tree (BST) called `MovieTree`. Please read trough this provided source file carefully to make sure you understand each field and method in the class.

2 MovieTree and MovieTreeTester Classes

We provide you in the following with the templates for the MovieTree and MovieTreeTester classes.

- Download those two files and add them to your project. Notice that we added default return statements to the incomplete methods to simply let the code compile. You are going to complete the implementation of all the methods defined in these classes and including the `TODO` tag with respect to the details provided in their javadoc method headers. **Make sure to remove the TODO tags once you complete the implementation of each method.**

- Recall that you ARE NOT allowed to add any additional fields either instance or static fields to the `MovieTree` class. Besides, NO additional public methods must be added to this class. Read carefully all the details provided in the javadoc method headers. If a method is described to be a *recursive helper* method, you are not allowed to implement it using iteration (for or while loops). They MUST be designed and implemented using recursion.

- Duplicate Movies with exactly the same years, ratings and names at the same time are not allowed in this system. Note also that in our `MovieTree` binary search tree, the increasing order of movies records goes from the earliest year to the latest year. If there is a tie of years, it goes from the lowest rating to the highest rating. If there is still a tie in terms of ratings, the sequence will follow increasing alphabetical order.

- Finally, we highlight that you are responsible for testing thoroughly your implementation of the `MovieTree` public methods. You MUST define and implement at least the FIVE unit test methods defined in the `MovieTreeTester` class. Make sure to test every method with at least 3 different scenarios. Hints are provided in the provided javadocs method headers.

Illustrative Example

In order to provide you with a better understanding on how to use the implemented classes, demo.txt contains an example of source code. You can find in the following its expected output. Do not use this source code to test your methods. Rely on the implementation of your own test scenarios.

```
Size: 0 Height: 0
Catalog:

============================================================
Size: 2 Height: 2
Catalog:
[(Year: 1988) (Rate: 9.5) (Name: Best)]
[(Year: 2018) (Rate: 6.5) (Name: Airplanes)]

============================================================
Size: 8 Height: 5
Catalog:
[(Year: 1988) (Rate: 9.5) (Name: Best)]
[(Year: 2015) (Rate: 8.5) (Name: Grand Parents)]
[(Year: 2017) (Rate: 5.5) (Name: Dogs)]
[(Year: 2018) (Rate: 6.0) (Name: Flights)]
[(Year: 2018) (Rate: 6.0) (Name: Yes)]
[(Year: 2018) (Rate: 6.5) (Name: Airplanes)]
[(Year: 2018) (Rate: 7.5) (Name: Earth)]
[(Year: 2018) (Rate: 8.5) (Name: Cats)]

This catalog contains (2018, 7.5, Earth): true
This catalog contains (2016, 8.4, Flowers): false

Best movie: [(Year: 2018) (Rate: 8.5) (Name: Cats)]

Lookup query: search for the movies of 2018 rated 6.5 and higher
[[(Year: 2018) (Rate: 6.5) (Name: Airplanes)], [(Year: 2018) (Rate: 8.5) (Name: Cats)],
[(Year: 2018) (Rate: 7.5) (Name: Earth)]]
```

```
Lookup query: search for the movies of 2018 with rated 8.0 and higher
[[(Year: 2018) (Rate: 8.5) (Name: Cats)]]

Lookup query: search for the movies of 2015 with rated 9.0 and higher
No search results.
```

3 Assignment Submission

Congratulations on finishing this CS300 assignment! After verifying that your work is correct, and written clearly in a style that is consistent with the CS300 Course Style Guide, you should submit your final work through gradescope.com. The only 2 files that you must submit include: `MovieTree.java` and `MovieTreeTester.java`. Your score for this assignment will be based on your **"active"** submission made prior to the hard deadline of **11:59PM on April 22**nd. The second portion of your grade for this assignment will be determined by running that same submission against additional offline automated grading tests after the submission deadline. Finally, the third portion of your grade for your submission will be determined by humans looking for organization, clarity, commenting, and adherence to the CS300 Course Style Guide.

4 Assignment Submission

Congratulations on finishing this CS300 assignment! After verifying that your work is correct, and written clearly in a style that is consistent with the CS300 Course Style Guide, you should submit your final work through gradescope.com. The only 2 files that you must submit include: `MovieTree.java` and `MovieTreeTester.java`. Your score for this assignment will be based on your **"active"** submission made prior to the hard deadline of **11:59PM on April 22**nd. The second portion of your grade for this assignment will be determined by running that same submission against additional offline automated grading tests after the submission deadline. Finally, the third portion of your grade for your submission will be determined by humans looking for organization, clarity, commenting, and adherence to the CS300 Course Style Guide.

©**Copyright:** This write-up is a copyright programming assignment. It belongs to UW-Madison. This document should not be shared publicly beyond the CS300 instructors, CS300 Teaching Assistants, and CS300 Spring 2021 fellow students. Students are NOT also allowed to share the source code of their CS300 projects on any public site including github, bitbucket, etc.