CS 2336 – PROJECT 4 – Redbox Inventory System (Standalone Project 2)

Pseudocode Due: 3/26 by 11:59 PM

Project Due: 4/10 by 11:59 PM

Submission and Grading:

- All project deliverables are to be submitted in eLearning.
- The pseudocode should be submitted as a Word or PDF document and is not accepted late.
- Zip all of your source files into a single zip file for submission.
- Projects submitted after the due date are subject to the late penalties described in the syllabus.
- Programs must compile and run with JDK 8.
- Each submitted program will be graded with the rubric provided in eLearning as well as a set of test cases. These test cases will be posted in eLearning after the due date. Each student is responsible for developing sample test cases to ensure the program works as expected.
- **Type your name and netID in the comments at the top of all files submitted.**
- **Your main class file must be named Main.java**
    - **Keep Main.java in the default package**
- **The binary search tree classes must be in a package named BSTree**
- **Zip the contents of the src subdirectory into a single zip file (not .rar, not .tar, not .7z). *Do not zip the src directory, only its contents.***
- **Projects that are not submitted properly will receive a 10 point deduction from the grade.**

Objectives: Create and manipulate a binary search tree in Java. Utilize String functions to validate input.

Problem: Redbox is in need of a program to track inventory and to generate a report for their DVD rental kiosks. Given a log of transactions including renting and returning DVDs as well as adding and removing DVD titles, the program will need to process each transaction and create a report after all transactions have been processed. The generated report will list all DVD titles stored in the kiosk as well as how many of each disc are in the kiosk.

Details

- The inventory will be held in a binary search tree
- Use the DVD title to determine node placement in the tree
- The binary tree will be seeded with an inventory file
- Once seeded, the program will parse a transaction log to update the inventory
- There are five possible transactions
    - Add new title
    - Add copies of title
    - Remove copies of a title
    - Rent a DVD
    - Return a DVD
- Add new title
    - Create a new node and insert it into the tree
- Add copies of title

    - Find the title in the tree and increase the number of available copies by the amount listed
- Remove copies of a title
    - Find the title in the tree and reduce the number of available copies by the amount listed
    - If number available is zero and no copies are rented out, delete the node from the tree
    - There will not be more copies removed than available.
- Rent a DVD
    - Reduce amount by one and increase rented amount by one
- Return a DVD
    - Increase amount by one and reduce rented amount by one

Classes

- Node
    - Members
        - Title (string)
        - Available (integer)
        - Copies rented (integer)
        - Left (node pointer)
        - Right (node pointer)
    - Methods
        - Overloaded constructor
        - Mutators
        - Accessors
- Binary Search Tree
    - Members
        - Root (node pointer)
    - Methods
        - Mutator
        - Accessor
        - Other methods that are necessary to interact with a binary search tree
            - Remember methods should be generic enough to be used on a binary tree regardless of the problem.
            - **Any method that traverses the tree must be recursive.**

User Interface and Input: There is no user interface for this program.

Input will be given in two files, `inventory.dat` and `transaction.log`. The program will read `inventory.dat` first. This will create the binary tree. Each line (except the last line which will not have a newline character) in `inventory.dat` will be formatted as follows:

`"<title>",<quantity available>,<quantity rented><\n>`

After processing the inventory file, begin processing `transaction.log`. Each line of the file should follow one of the following formats (Note: the last line will not end with a newline character):

- `add "<title>",<number to add><\n>`
- `remove "<title>",<number to remove><\n>`
- `rent "<title>"<\n>`
- `return "<title>"<\n>`

The transaction file may contain errors due to network disruptions from the main server. For each line in the transaction log, validate that it follows one of the formats listed above. If it is the correct format, process the transaction. If the line is invalid, write the line to an error file (as described below). All numbers are expected to be integers. In order to be valid, the line must follow the format exactly.

Output: A file named `error.log` will be created if any lines of `transaction.log` are invalid. `Error.log` will contain all invalid entries of the transaction file.

At the end of the program, create a formatted report to display each title, the number of copies available to rent for that title as well as the number of copies that are currently rented. The titles should be listed in alphabetical order (without the double quotes). The report should be arranged in three columns:

- Title
- Copies available
- Copies rented

Write the report to a file named `redbox_kiosk.txt`.