Project 4

Redbox Inventory System

KEY ITEMS: Key items are marked in red. Failure to include or complete key items will incur additional deductions as noted beside the item.

Submission and Grading:

- The pseudocode will be submitted in eLearning as a PDF document and is not accepted late.
- All project source code will be submitted in Zybooks.
- Projects submitted after the due date are subject to the late penalties described in the syllabus.

• Type your name and netID in the comments at the top of all files submitted. (-5 points)

Objectives:

- Implement a binary search tree class in Java.
- Utilize a binary search tree
- Perform simple input validation with string functions

Specification

Problem: You have been hired by Redbox to develop reports for vending kiosk around the US based on rental data transmitted from each unit. A file containing all the transitions (rentals, returns and inventory maintenance) over a given period will be analyzed so that a report can be generated. Your program will process each transaction and create a report listing all DVD titles and their quantities in the kiosk.

Pseudocode: Your pseudocode should describe the following items

- Main.java
    - List functions you plan to create
        - Determine the parameters
        - Determine the return type
        - Detail the step-by-step logic that the function will perform
    - Detail the step-by-step logic of the main function
- Binary search tree functions
    - Insert (return type void)

    - Search (return type generic)
    - Delete (return type void)
    - print_Tree //inorder printing to the console.
    - print_Tree_preOrder // pre_Order printing to The PrintStream passed as an argument.
- A list of at least 10 test cases you will check during testing
    - Specific input is not necessary
    - Describe what you are testing

Core Implementation:

- Read the inventory file and build the tree
    - BST functions must be recursive
- Process sample transaction file
    - Add new title
    - Add copies of existing title
    - Remove copies of existing title without removing node from tree
    - Rent copies
    - Return copies
- Create report
    - In-order to the Console
    - Pre-order to the file `inventory.out`

Zybooks Information:

- You will have multiple source files
    - `Main.java`
    - `BinTree.java`
    - `Node.java`
    - `DVD.java`
- Core implementation has unlimited submissions
    - This will help you make sure the basic actions of your program are working properly in the environment
- Final submission is limited to 10 submissions
- White space will not be checked

Details

- **To avoid potential errors when grading, do not create multiple Scanner objects for System.in. If the Scanner object is needed in multiple functions, please pass the Scanner object into the function.**
- **Do not create a package**

- All classes must be of your own design and implementation.

    - **Do not use the pre-defined Binary Search Tree Java class (-20 points if pre-defined classes used)**
- The binary tree will be seeded with an inventory file
- Once seeded, the program will parse a transaction log to update the inventory
- Use the DVD title as the comparator (alphabetical)
- There are five possible transactions
    - Add new title
    - Add copies for an existing title
    - Remove copies of an existing title
    - Rent a DVD
    - Return a DVD
- Add new title
    - Create a new node and insert it into the tree
- Add copies of an existing title
    - Find the title in the tree and increase the number of available copies by the amount listed
- Remove copies of an existing 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 available amount by one and increase rented amount by one
- Return a DVD
    - Increase available amount by one and reduce rented amount by one

Classes

- DVD – `DVD.java`
    - Must be comparable
    - Members
        - Title (string)
        - Available (integer)
        - Rented (integer)
    - Methods
        - Overloaded constructor
        - Mutators
        - Accessors
        - toString
        - compareTo
- Node – `Node.java`
    - Generic class (-10 points if not)
    - Members

        - Left (node pointer)
        - Right (node pointer)
        - Payload (generic)
    - Methods
        - Default constructor
        - Overloaded constructor
        - Mutators
        - Accessors
- Binary Search Tree – `BinTree.java`
    - Must be generic to use generic Node. Must be Comparable
    - Members
        - Root (node pointer)
    - Methods
        - Default constructor
        - Overloaded constructor
        - Mutator
        - Accessor
        - insert
        - search
        - delete
        - print_Tree
        - print_Tree_preOrder
    - Any additional functions added must be specific to any binary search tree
        - Each of the public methods `insert, search, delete, print_Tree` and `print_Tree_preOrder` must have overloaded private recursive methods.
        - Any functions related to the specific problem should be in main
    - All traversals of the tree will be done recursively (-10 points if not)
        - This includes functions that add, delete, search, print and display the tree

User Interface and Input:

Prompt the user for the files in this order:

- Inventory file
- Transaction log

The program will read inventory file first. This will create the binary tree. Each line (except the last line which may not have a newline character) will be formatted as follows:

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

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

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

You may assume all input will be valid.

Output: At the end of the program, display a formatted report about each title to the console. The titles should be listed in alphabetical (in-order) order (without the double quotes). The report should be arranged in three formatted columns that line up the data nicely:

- Title
- Copies available
- Copies rented

For Example

```
Collateral Beauty              0       3
Doctor Strange                 2       1
Fences                         1       3
Guardians of the Galaxy        0       3
Hacksaw Ridge                  3       1
```

The program should write the preorder traversal to the output file `inventory.out` in the same format.