Final Project: Library checkout system

Name: Aiyanah "Kaiy" Muhammad

Date: 7/7/2019

Description:

My final project is a library checkout system that uses recursion, searching, sorting, ADTS, polymorphism, and the graphical user interface.

Components:

- public class Final
    - checkOut method greets patron, shows books available for selection, then prompts the user to input the release year of their desired book. Then, after going through a sorting algorithm, the releaseYear array is binary searched for the input year. If the year is found and the title is not unavailable, the title at that index is marked unavailable. After, there is a call to the dueMethod to get a due date that is a user input amount of weeks from the day. This due date is communicated to the user along with a thank you message. Else if the input year is not found or the title is not available, there is a sorry message. Finally, the patron is asked if they want to check out another book.
    - dueMethod method receives the user input amount of 2, 3, or 4 weeks and calls the appropriate Due subclass. If the input is any other number, the user is alerted and the method call itself recursively.

    - main method is the driver. It processes a queue of patrons by having each of them call the checkOut method then be removed from the queue.
- class BinarySearch
    - The BinarySearch class searches for an array element that is equal to the target variable by starting at the middle of the array to can rule out half because the array is ordered. Then is goes to the middle of the potential candidates and rules out half. The search does this recursively until it finds the target or exhaust its search possibilities. If it finds the target, BinarySearch returns the index at which it was found. If not, -1 is returned.
- class Due
    - In this class, weeks is not defined.
    - getDate method adds the value of the integer variable weeks to the current time using ChronoUnits. Then, it displays a message with the due date.
- class DueIn2 extends Due
    - In this class weeks = 2.
- class DueIn3 extends Due
    - In this class weeks = 3.
- class DueIn4 extends Due
    - In this class weeks = 4.

Required Elements:

1. Use of at least three Abstract Data Types

- Abstract data times are used in the main method
    - Lines 122-135

2. Use of inheritance and polymorphism
    - Inheritance and polymorphism are used in the Due class and its subclasses.
        - Lines 164-193
3. Use of recursion
    - Recursion is used inside the BinarySearch class.
        - Lines 154 and 156
    - Also, in dueMethod
        - Line 96
4. At least one sorting algorithm
    - Sorting algorithm is used on releaseYear array.
        - Line 37
5. At least one search algorithm
    - BinarySearch algorithm is used on releaseYear array.
        - Lines 48-48 and 142-161
6. Use of a graphical user interface
    - A GUI is used throughout the program

Big-O Notation Analysis:

- Program has $O(n\log_2 n)$ time complexity

    - Main method—O($n\log_2 n$)— calls checkOut method— O($n\log_2 n$)— which calls dueMethod—O(n)

Difficulties:

- One of my difficulties figuring out how the user would select the books. At the beginning, I used ISBNs (International Standard Book Numbers) because they are essentially book ID numbers. However, after testing and entering 13 digits to select a book each time, I realized that this would be annoying to the user and switched to release years.
- Another difficulty was using three ADTs to satisfy the project requirements because I wasn't entirely sure what constitutes an ADT (if it was an interface or a collection or operations on a collection).
- Determining the time complexity was very confusing because the book doesn't include how to determine complexity some of the situations that my program had (such as multiple class that call each other).
- Lastly, editing the checkOut method so the same book can't be checked out twice was a challenge. Because of the way I had my code, after being checked out, the title would change to title[i] + (unavailable). The title said unavailable, but it could still be checked out. If the user input the corresponding release year, it would be checked out again and the title would become title[i] + (unavailable) + (unavailable). Another (unavailable) would be added every time indefinitely. To solve this I used !title[index].endsWith(" (unavailable)") to make sure that it hasn't been checked out already.