Fatma Zehra Yar 230446404

LINKED LIST-BOOK LIST

* Explain the Big O time complexity for each method (addBook, removeBook, f indBook).

addBook() works for adding a book to the end of the list. It is O(n)

removeBook() works for removing a book by its ID. It isO(n)

findBook() works for searching a book by its ID and returning it. It is O(n)

printList() works for printing all books in the list. It is O(n)

* Compare the performance of a Linked List vs. an ArrayList for these operations. Why is a Linked List suitable (or not) for this task?

Linked list is better because we use ArrayList for adding items continuoulsy fo it wants a big memory. But in this situation, we only add books when we need it.

RETURN REQUEST

In stack, we use these methods for return request;

Push() works for inserting an item to the list. It is O(1) because we insert from the top.

Pop() works for deleting an item to the list. It is O(1) because we delete from the top. We use LIFO.

Peek() works for printing the item on the top. It is O(1) because we print item from the top.

* Why is a Stack suitable for managing return requests?

A stack works better because returns are often handled in reverse order like LIFO, so it is better.

* What issues would arise if a Queue were used instead? Compare their Big O complexities.

If we use FIFO, it would delete some records.

BORROW REQUEST

In queue, we use these methods for borrow request;

Enqueue() works for adding a borrowing request to the queue. O(1)

Dequeue() works for removing and returning a first borrowing request. O(1)

Size() works for returning the size of the queue. O(1)

printRequest() works for printing all requests. O(1)

* What issues would arise if a Stack were used instead? Compare their Big O complexities?

We use FIFO in queue but LIFO in stack. So the last person would get the request, not the first one. It is not suitable.

* Why is a Queue suitable for managing borrowing requests?

It is suitable because we need a line for request. First person need to gets first.

LINKED LIST-BOOK LIST

* Explain the Big O time complexity for each method (addBook, removeBook, f indBook).

addBook() works for adding a book to the end of the list. It is O(n)

removeBook() works for removing a book by its ID. It isO(n)

findBook() works for searching a book by its ID and returning it. It is O(n)

printList() works for printing all books in the list. It is O(n)

* Compare the performance of a Linked List vs. an ArrayList for these operations. Why is a Linked List suitable (or not) for this task?

Linked list is better because we use ArrayList for adding items continuoulsy fo it wants a big memory. But in this situation, we only add books when we need it.