Code Documentation for HashMapPractice Project

By Rachel Hussmann

Overview

The purpose of the HashMapPractice project was to create our own hashmap and hashing algorithms from scratch. This project has four classes.

How It Works

Book

The Book class is a simple class responsible for holding information about a Book object. The main goal for this class was to test if the generic variable for the LinkedList and the RachelSimpleHashMap classes. It has one constructor and one method:

- Book
    - Parameter: String name – The name of the book
    - Functionality – Constructor for the Book class
    - Returns: Nothing
- toString
    - Parameters: None
    - Functionality – Returns a string representative of the object, for this class, it is the name of the book
    - Returns: String – The name of the book

LinkedList<E>

The LinkedList class contains methods and inner classes that assist in the creation and management of a linked list of items. This class has the ability to handle generic cases and has an inner private class called Node. The parameter for this class is E, which is the representation of a generic object of any class. This class has seven methods:

- add
    - Parameter: E item – The generic object to be added to the list
    - Functionality – Accepts a new item and adds it to the linked list
    - Returns: Nothing
- add

    - Parameters: E item – The generic object to be added to the list, int index – The index to add the new item at
    - Functionality – Accepts a new item and adds a new node at the given index
    - Returns: Nothing
- remove
    - Parameters: int index – The index of the Node to be deleted
    - Functionality – Deletes the Node at the specified index
    - Returns: Nothing
- get
    - Parameter: int index – The index of the object the user wants returned
    - Functionality – Finds the Node in the list and returns the data of the Node at that index
    - Returns: E - The generic object at the specified index
- set
    - Parameters: int index – The index of the Node to be changed, E item – The new item to be set at the specified index
    - Functionality – Changes the data within the Node at the specified index
    - Returns: E – The data that has been replaced
- size
    - Parameters: None
    - Functionality – Returns the current number of nodes in the linked list
    - Returns: int – The current number of nodes in the linked list
- Iterator
    - This method is a special case because while it acts as a method, it also declares some methods within it, which will be discussed here.
    - Parameters: None
    - Functionality – Creates a way for the class to use foreach loops and other iteration-based methods
    - Returns: Iterator<E> - The Iterator object for the LinkedList class
        - Within the return statement, some methods are declared for the iterator object
        - hasNext
            - Parameters: None
            - Functionality – Checks to see if the current Node has an object attached to the .next pointer by checking to see if the .next object is null
            - Returns: boolean - True if the list does have a next element, False if the list does not have a next element
        - next
            - Parameters: None

- Functionality – Captures the data from a Node, returns it and moves down the list to the next Node
- Returns: E - The object within the current Node

Node<E>

The Node class is a private inner class within the LinkedList class. It is responsible for creating each collection of data for each part of the linked list. The class contains two variables and one constructor:

- E data
- Node<E> next
- Node
    - Parameter: E dataItem – The data to be held within the node
    - Functionality – Constructor for the Node class
    - Returns: Nothing

RachelSimpleHashMap<E>

The RachelSimpleHashMap contains methods that create hashes from keys and creates an array of linked lists that hold the key and data value pairs. This class has the ability to handle generic cases and has an inner private class called KeyValuePair. The parameter for this class is E, which is the representation of a generic object of any class. The class contains ten methods:

- add
    - Parameters: String key – The key value that will be hashed and used to find the item, E item – The generic object or data to be stored
    - Functionality – Adds a key and an item to the hashmap
    - Returns: Nothing
- get
    - Parameter: String key – The key value for the desired object
    - Functionality – Returns the data associated with the key
    - Returns: E – The generic object associated with the given key
- remove
    - Parameter: String key – The key values that will be hashed and used to find the item
    - Functionality – Removes a key value pair from the hashmap
    - Returns: Nothing
- size

    - Parameters: None
    - Functionality – Returns the number of key value pairs added to the hashmap
    - Returns: int – The total number of key value pairs in the hashmap
- isEmpty
    - Parameters: None
    - Functionality – Checks to see if the hashmap is empty
    - Returns: boolean – True if the hashmap is empty, False if the hashmap is not empty
- checkNumberOfCollisions
    - Parameters: None
    - Functionality – Prints out the structure of the hashmap with the number of collisions next to it
    - Returns: String – The structures of the hashmap with the number of collisions as a string value
- dumbHash
    - Parameter: String key – The key to be hashed
    - Functionality – Takes a key value and hashes it by returning the number of characters in the string
    - Returns: int – The hashed key (The number of characters in the key)
- contains
    - Parameter: String key – The key value to find in the hashmap
    - Functionality – Checks to see if a key exists in the hashmap
    - Returns: boolean – True if the key was found in the hashmap, False if the key was not found
- checkBounds
    - Parameter: int hash – The hashed key value
    - Functionality – Checks to see if the hash is larger than the size of the hashmap
    - Returns: Nothing
- resize
    - Parameters: None
    - Functionality – Dynamically resizes the array when it needs to be expanded
    - Returns: Nothing

KeyValuePair

The KeyValuePair class is a private inner class that is used to store the key and value in one place for the RachelSimpleHashMap class. It has two variables and one method:

- String key
- E data

- KeyValuePair
    - Parameters: String key – The key for the pair, E data – The data associated with the pair
    - Functionality – Constructor for the KeyValuePair class
    - Returns: Nothing

HashMapTester

The HashMapTester class contains the main method and is used to test the methods from the RachelSimpleHashMap class.

Output

Output for creating and acting on an integer RachelSimpleHashMap

```
Adding new items to the hashmap . . .

What value is associated with the key Zero: 0
What value is associated with the key One: 1
What value is associated with the key Three: 3

What is the size of the HashMap: 3

Adding more new items to the hashmap . . .

Linked List at index 3:
[One, 1]
Number of collisions: 0

Linked List at index 4:
[Zero, 0]
[Four, 4]
Number of collisions: 1

Linked List at index 5:
[Three, 3]
[Seven, 7]
Number of collisions: 1
```

```
Does the hashmap contain the key Zero before deleting: true

Deleting key Zero . . .

Does the hashmap contain the key Zero after deleting: false
```

Output for creating and acting on a Book RachelSimpleHashMap

```
Creating new hashmap . . .

Linked List at index 6:
[Austin, Pride and Prejudice]
Number of collisions: 0

Linked List at index 11:
[Shakespeare, Hamlet]
[Shakespeare, Romeo and Juliet]
Number of collisions: 1
```