Project 1c Navigation

Introduction

Getting the Skeleton Files

Phase 0: The Deque Interface

Phase 1: Basic Palindrome

Phase 2: Generalized Palindrome

Phase 3: OffByN

Phase 4 (extra credit):

Submission

Tips

Frequently Asked Questions

Project 1c: Data Structures Part 3

This is the third part of project 1. The deadline is 2/11.

This project is brand new. Please let us know on Piazza if you spot any bugs or issues.

Introduction

In project 1c, you will build a program that uses the Deque classes to find English words with interesting properties. We will provide the following files for your use:

- `CharacterComparator.java` : An interface for comparing characters.
- `LinkedListSolution.java` : A correct implementation of LinkedList.
- `PalindromeFinder.java` : Class that helps identify cool words in English.

In addition, you should download a list of English words from this link.

For this project, you are allowed to work with a partner, and your partner must be the same as you had for part a and part b. If you wish to dissolve your partnership from part a or part b, please send an email to the course staff for approval.

Unlike projects 1a and 1b, this mini-project is highly-scaffolded in order to maximize the time you spend thinking about core course material (HoFs, interfaces) vs. general system design and debugging.

Project 1c Navigation

Introduction

Getting the Skeleton Files

Phase 0: The Deque Interface

Phase 1: Basic Palindrome

Phase 2: Generalized Palindrome

Phase 3: OffByN

Phase 4 (extra credit):

Submission

Tips

Frequently Asked Questions

Getting the Skeleton Files

As with previous assignments, pull the skeleton using the command `git pull skeleton master`.

If you're using IntelliJ make sure to reimport the project.

For this assignment, you'll be working with Deques. You are welcome to use your LinkedListDeque, your ArrayDeque, or the provided LinkedListDequeSolution.java. If you're not sure if your solution to project 1a is correct, feel free to use LinkedListDequeSolution.java instead.

Phase 0: The Deque Interface

Later in this assignment, you will write methods that return and accept `Deque` objects as inputs. Rather than create separate methods that deal with ArrayDeques and LinkedListDeques, we'll instead have them operate on objects of type Deque, which we'll define as an interface.

Create an interface in `Deque.java` that contains all of the methods that appear in both `ArrayDeque` and `LinkedListDeque`. See the project 1a spec for a concise list.

After creating this interface, modify any `Deque` implementation you intend to use for later parts of this project (`LinkedListDeque`, `ArrayDeque`, or `LinkedListDequeSolution`) so that they implement the `Deque` interface. Add @Override tags to each method that overrides a `Deque` method.

Note: If you're using LinkedListDequeSolution, which relies on some inheritance black magic, your class definition should look like `public class LinkedListDequeSolution<Item> extends LinkedList<Item> implements Deque<Item>`.

Project 1c Navigation

Introduction

Getting the Skeleton Files

Phase 0: The Deque Interface

Phase 1: Basic Palindrome

Phase 2: Generalized Palindrome

Phase 3: OffByN

Phase 4 (extra credit):

Submission

Tips

Frequently Asked Questions

Phase 1: Basic Palindrome

Create a class Palindrome, and implement the two methods shown below:

- `public static Deque<Character> wordToDeque(String word)`
- `public static boolean isPalindrome(String word)`

The `wordToDeque` method should be straightforward. You will simply build a Deque where the characters in the deque appear in the same order as in the word.

The `isPalindrome` method should return true if the given word is a palindrome, and false otherwise. A palindrome is defined as a word that is the same whether it is read forwards or backwards. For example "a", "racecar", and "noon" are all palindromes. "horse", "rancor", and "aaaaab" are not palindromes. *Any word of length 1 or 0 is a palindrome.*

Tip: Search the web to see how to get the ith character in a String.

Tip: Just like how we inserted an int into an `SList<Integer>` , we can insert chars into a `Deque<Character>` .

Tip: I do not recommend writing JUnit tests for `wordToDeque` . Instead, use the `printDeque` method to make sure things look correct.

Tip: Consider recursion. It's a more beautiful approach to this problem IMO.

Just for fun: Uncomment the main method in the provided PalindromeFinder.java class and you'll get a list of all palindromes of length 4 or more in English (assuming you also downloaded the provided words file).

Project 1c Navigation

Introduction

Getting the Skeleton Files

Phase 0: The Deque Interface

Phase 1: Basic Palindrome

Phase 2: Generalized Palindrome

Phase 3: OffByN

Phase 4 (extra credit):

Submission

Tips

Frequently Asked Questions

Phase 2: Generalized Palindrome

In this part, you will generalize your isPalindrome method by adding a new method:

- `public static boolean isPalindrome(String word, CharacterComparator cc)`

The method will return true if the word is a palindrome according to the character comparison test provided by the CharacterComparator passed in as argument `cc` . A character comparator is defined as shown below:

```
/** This interface defines a method for determin
public interface CharacterComparator {
    /** Returns true if characters are equal by
    boolean equalChars(char x, char y);
}
```

In addition to adding the method above to `Palindrome.java` , you should also create a class called `OffByOne.java` , which should implement `CharacterComparator` such that `equalChars` returns true for letters that are different by one letter. For example the following calls to obo should return true. Note that characters are delineated in Java by single quotes, in contrast to Strings, which use double quotes.

```
OffByOne obo = new OffByOne();
obo.equalChars('a', 'b')
obo.equalChars('r', 'q')
```

However, the two calls below should return false:

```
obo.equalChars('a', 'e')
obo.equalChars('z', 'a')
```

A palindrome is a word that is the same when read forwards and backwards. To allow for odd length palindromes, we do not check the middle character for equality with itself. So character off from itself.

Project 1c Navigation

Introduction

Getting the Skeleton Files

Phase 0: The Deque Interface

Phase 1: Basic Palindrome

Phase 2: Generalized Palindrome

Phase 3: OffByN

Phase 4 (extra credit):

Submission

Tips

Frequently Asked Questions

Tip: Make sure to include `@Override` when implementing equalChars. While it has no effect on the function of your program, it's a good habit for the reasons detailed in lecture.

Tip: To calculate the difference between two chars, simply compute their difference. For example `'d' - 'a'` would return `-3` .

Just for fun: Try printing out all off-by-one palindromes of length 4 or more in English (assuming you also downloaded the provided dictionary) by modifying PalindromeFinder.java. For example "flake" is an off-by-1 palindrome since f and e are one letter apart, and k and l are one letter apart.

Phase 3: OffByN

In this final phase of the project, you will implement a class OffByN, which should implement the `CharacterComparator` interface, as well as a single argument constructor which takes an integer. In other words, the callable methods and constructors will be:

- OffByN(int N)
- equalChars(char x, char y)

The OffBYN constructor should return an object whose `equalChars` method returns true for characters that are off by N. For example the call to equal chars below should return true, since a and f are off by 5 letters.

```
        OffByN offby5 = new OffByN(5);
        offBy5.equalChars('a', 'f')
```

Just-for-fun: Try modifying PalindromeFinder so that it outputs a list of offByN palindromes for the N of your choosing.

in English? What is the longest offByN palindrome for any N?

Project 1c Navigation

Introduction

Getting the Skeleton Files

Phase 0: The Deque Interface

Phase 1: Basic Palindrome

Phase 2: Generalized Palindrome

Phase 3: OffByN

Phase 4 (extra credit):

Submission

Tips

Frequently Asked Questions

Phase 4 (extra credit):

Pull from skeleton, which will create a proj1d directory.

Fill out the Project 1 Survey. You will be given a secret word.

Add this secret word to MagicWord1D.java in the proj1d directory, and submit to the Project 1d (extra credit) autograder.

Submission

Submit `Deque.java` , `Palindrome.java` , `OffByOne.java` , `OffByN.java` and any supporting files you require, including `ArrayDeque.java` and `LinkedListDeque.java` or `LinkedListDequeSolution.java` . Do not submit .class files.

Tips

None yet.

Frequently Asked Questions

LinkedListDequeSolution won't compile.

Make sure your class definition is `public class LinkedListDequeSolution<Item> extends LinkedList<Item> implements Deque<Item>` .

My implementation of LinkedListDeque or ArrayDeque won't compile.

Deque<Item> .

Project 1c Navigation

Introduction

Getting the Skeleton Files

Phase 0: The Deque Interface

Phase 1: Basic Palindrome

Phase 2: Generalized Palindrome

Phase 3: OffByN

Phase 4 (extra credit):

Submission

Tips

Frequently Asked Questions