HOMEWORK - SPRING 2018

HOMEWORK 3 - due Tuesday, March 6th no later than 6:00pm

REMINDERS:

- **Be sure your code follows the coding style for CSE214.**
- **Make sure you read the warnings about academic dishonesty.** ***Remember, all work you submit for homework assignments MUST be entirely your own work. Also, group efforts are not allowed.***
- **Login to your grading account and click "Submit Assignment" to upload and submit your assignment.**
- **You are allowed to use any built-in Java API Data Structure classes to implement this assignment except where noted.**
- **You may use Scanner, InputStreamReader, or any other class that you wish for keyboard input.**

After graduating from Stony Brook, you have landed your dream job at your favorite fruit-branded computer maker. As they are currently trying to modernize their lineup, and realize that they do not have a budget for 3d screens or even built in headphone jacks or SD card slots, they are compromising by adding a back button, much like you find in any modern phone. You will write a prototype for the back button, allowing it to work with such legendary applications as the superior maps app (which never directs drivers into large bodies of water), and the web browser. As an experienced algorist with not one, but two copies of the Algorithm Design Manual (both hardcover), you realize that the underlying data structure for the back button is a stack, so you will implement a simulation that demonstrates how the back button will work in the various applications and will allow the user to switch between applications.

NOTE: All exceptions explicitly thrown in Required Classes except for IllegalArgumentException are custom exceptions that need to be made by you.

Required Classes

Command

Write a fully-documented interface named Command to represent each command entered on the phone app. The Command interface will require the following methods:

- public boolean validCommand(CommandStack stack)
  - Returns whether this Command is valid and can be added to the given CommandStack. Specific behavior will be outlined in the following classes.
- public String toString()
  - Returns the String representation of this Command in long form (for current screen display)
- public String toShortString()

- Returns the String representation of this Command in short form (for stack display)

Map Commands

- **FindPlace implements Command**
  - Write a fully-documented class named FindPlace to represent the "F: Find a place" command for the Maps app. This class requires the following properties (with NO mutators, accessors are allowed) and methods:
    - private String destination
    - public FindPlace(Scanner scanner)
      - Constructs this FindPlace instance accordingly using the Scanner.
    - public boolean validCommand(CommandStack stack)
      - Returns whether or not pushing this FindPlace command will be valid for the given stack.
    - public String toString()
      - Returns the String representation of this Command in long form (for current screen display)
    - public String toShortString()
      - Returns the String representation of this Command in short form (for stack display)
- **PlanRoute implements Command**
  - Write a fully-documented class named PlanRoute to represent the "P: Plan a route" command for the Maps app. This class requires the following properties (with NO mutators, accessors are allowed) and methods:
    - private String source
    - private String destination
    - public PlanRoute(Scanner scanner)
      - Constructs this PlanRoute instance accordingly after reading input from the scanner.
    - public boolean validCommand(CommandStack stack)
      - Returns whether or not pushing this PlanRoute command will be valid for the given stack.
    - public String toString()
      - Returns the String representation of this Command in long form (for current screen display)
    - public String toShortString()
      - Returns the String representation of this Command in short form (for stack display)

- **StartNavigation implements Command**
  - Write a fully-documented class named StartNavigation to represent the "N: Start Navigation" command for the Maps app. This class requires the following properties (with NO accessors and mutators) and methods:
    - private String source
    - private String destination
    - public StartNavigation (CommandStack commandStack)
      - Constructs this StartNavigation instance accordingly after grabbing navigation information from the last instruction
    - public boolean validCommand(CommandStack stack)
      - Returns whether or not pushing this StartNavigation command will be valid for the given stack.
      - This cannot be placed on top of another StartNavigation command or an empty stack
    - public String toString()
      - Returns the String representation of this Command in long form (for current screen display)
    - public String toShortString()
      - Returns the String representation of this Command in short form (for stack display)

Safari Commands

- **GoogleSomething implements Command**
  - Write a fully-documented class named GoogleSomething to represent the "G: Google something" command for the Safari app. This class requires the following properties (with NO accessors and mutators) and methods:
    - private String query
    - public GoogleSomething (Scanner scanner)

        - Constructs this GoogleSomething instance accordingly after reading input from the scanner.
    - public boolean validCommand(CommandStack stack)
        - Returns whether or not pushing this GoogleSomething command will be valid for the given stack.
    - public String toString()
        - Returns the String representation of this Command in long form (for current screen display)
    - public String toShortString()
        - Returns the String representation of this Command in short form (for stack display)
- **GoToBookmark implements Command**
    - Write a fully-documented class named GoToBookmark to represent the “F: Go to favorite/bookmark” command for the Safari app. This class requires the following properties (with NO accessors and mutators) and methods:
        - private String bookmark
        - public GoToBookmark (Scanner scanner)
            - Constructs this GoToBookmark instance accordingly after reading input from the scanner.
        - public boolean validCommand(CommandStack stack)
            - Returns whether or not pushing this GoToBookmark command will be valid for the given stack.
        - public String toString()
            - Returns the String representation of this Command in long form (for current screen display)
        - public String toShortString()
            - Returns the String representation of this Command in short form (for stack display)

- **FollowLink implements Command**
    - Write a fully-documented class named FollowLink to represent the “L: FollowLink” command for the Safari app. This class requires the following properties (with NO accessors and mutators) and methods:
        - private String link
        - public FollowLink (Scanner scanner)
            - Constructs this FollowLink instance accordingly after reading input from the scanner.
        - public boolean validCommand(CommandStack stack)
            - Returns whether or not pushing this FollowLink command will be valid for the given stack.
            - This cannot be placed on top of an empty stack
        - public String toString()
            - Returns the String representation of this Command in long form (for current screen display)
        - public String toShortString()
            - Returns the String representation of this Command in short form (for stack display)

CommandStack

Write a fully-documented class named CommandStack. For this homework assignment, you are allowed to use built-in Java libraries for your stack. You may choose to extend a stack (e.g. LinkedList<Command>) or implement your own (using an array, ArrayList or LinkedList). CommandStack must **only** use stack methods (push, pop, isEmpty, and peek), with the exception of printing.

- public void push(Command command)
    - Pushes command onto the top of the backing data structure.
    - Throws:
        - InvalidCommandException if the Command is invalid given the current state of this CommandStack.
            - You can use the command's built in method to check
            - Make sure safari commands aren't placed on top of map commands and vice versa
        - If you are extending LinkedList, you may rename this pushCommand(Command command) if your IDE is complaining about

throwing a InvalidCommandException
- public Command pop()
  - Removes the topmost Command from the stack and returns it.
  - Throws:
    - EmptyStackException if the stack was empty.
    - If you are extending LinkedList, you may rename this popCommand() if your IDE is complaining about throwing a EmptyStackException
- public Command peek()
  - Returns the topmost Command from the stack without removing it. The stack should be unchanged as a result of this method.
  - Throws:
    - EmptyStackException if the stack was empty
- public boolean isEmpty()
  - Returns true if the stack is empty, false otherwise.
- public String getScreenCommand()
  - Returns a String representation of the Command that will be displayed on the screen.
- public String toString()
  - Returns a String representation of this CommandStack. See the sample I/O.

Application
Write a fully-documented class named Application. This class will have the following properties and methods:
Note: you may make this abstract and have two implementing classes Maps and Safari. You may make all private methods and variables protected in this case.
- private CommandStack stack;
- public void readCommand(Scanner scanner)
  - Reads in input from the scanner to construct a Command and add it to the CommandStack.
  - After determining command type, use the command's built in method to read from scanner to populate data fields
  - Throws:
    - InvalidCommandException if the Command is invalid given the current state of the stack
- public void goBack()
  - Returns the application to the state it was before the most recent Command.
  - Throws:
    - EmptyStackException if there was no Command entered

iCatchUp
Write a fully-documented class named iCatchup. This class will have the following properties and methods:
- public static void main(String[] args)
  - The main method runs a menu driven application which allows the user to create two instances of the Application class and then prompts the user for input based on which screen it is currently in (Home, Maps, or Safari). The required information for each command is then requested from the user based on the selected operation.

InvalidCommandException
An Exception class that is thrown when trying to push a Command that is not valid for the given CommandStack.

EmptyStackException
An Exception class that is thrown when trying to pop from a stack with no elements.

Note on Exceptions: all exceptions should be handled gracefully - they should be caught in the main, and the user should be notified by a nice printout. Your messages will not be graded for creativity, but they should clearly indicate what the problem is (bad index, full list, negative number, etc.). The program should continue to run normally after an exception is encountered. We will not be checking Input Mismatch cases.

Pretty Printing: Here is a tutorial on how to print tables neatly using printf in java. It is highly encouraged that you print the output neatly, as it makes grading much easier.

https://docs.oracle.com/javase/tutorial/java/data/numberformat.html Here's a friendly reference page: https://alvinalexander.com/programming/printf-format-cheat-sheet

---

General Recommendations

You might want to use the toString() method for CommandStack to make debugging and printing easier. You do not have to do this, but it will help you.

You can feel free to add extra methods and variables if you need.

UI Required Functions

Note: please make sure that the menu is NOT case sensitive (so selecting A should be the same as selecting a).

- Home
  - S - Safari
  - M - Maps
  - Q - Quit
- Maps
  - F - Find a place
  - P - Plan a route
  - N - Start Navigation
- Safari
  - G - Google something
  - F - Go to a favorite/bookmark
  - L - Follow a link
- Both Safari and Maps
  - B - Back
  - S - Switch
  - H - Home

Program Sample

```
Welcome to the iPhony pocket telegraph simulator. You are on the home
screen.

Home Options:
    S) Safari
    M) Maps
    Q) Quit
Please select an option: M
Stack Debug:
[Home->MapsHome
Current Screen: Maps Home
Maps Options:
     F) Find a place
     P) Plan a route
     N) Start Navigation
     H) Home Screen
     S) Switch to Safari
     B) Back
Please select an option: N
No route or destination!

Stack Debug:
[Home->MapsHome
Current Screen: Maps Home
Maps Options:
     F) Find a place
     P) Plan a route
     N) Start Navigation
     H) Home Screen
     S) Switch to Safari
     B) Back
Please select an option: F
```

```
Please enter a location: Microsoft Store

Stack Debug:
[Home->MapsHome->F:Microsoft Store
Current Screen: Showing results for Microsoft Store
Maps Options:
     F) Find a place
     P) Plan a route
     N) Start Navigation
     H) Home Screen
     S) Switch to Safari
     B) Back
Please select an option: N
Maps Options:
     F) Find a place
     P) Plan a route
     N) Start Navigation
     H) Home Screen
     S) Switch to Safari
     B) Back
Current Screen: Navigating to Microsoft Store
Stack Debug:
[Home->SafariHome->F:Microsoft Store->N:Microsoft Store
Please select an option: B

Stack Debug:
[Home->MapsHome->F:Microsoft Store
Current Screen: Showing results for Microsoft Store
Maps Options:
     F) Find a place
     P) Plan a route
     N) Start Navigation
     H) Home Screen
     S) Switch to Safari
     B) Back
Please select an option: P
Please enter a source: Target
Please enter a destination: Walmart

Stack Debug:
[Home->MapsHome->F:Microsoft Store->P:Target-Walmart
Current Screen: Planning route from Target to Walmart
Maps Options:
     F) Find a place
     P) Plan a route
     N) Start Navigation
     H) Home Screen
     S) Switch to Safari
     B) Back
Please select an option: N

Stack Debug:
[Home->MapsHome->F:Microsoft Store->P:Target-Walmart->N:Target-
Walmart
Current Screen: Navigating from Target to Walmart
Maps Options:
     F) Find a place
     P) Plan a route
     N) Start Navigation
     H) Home Screen
     S) Switch to Safari
     B) Back
Please select an option: S

Stack Debug:
[Home->SafariHome
```

```
Current Screen: Safari Home
Safari Options:
     G) Google Something
     F) Go to a favorite (bookmark)
     L) Follow a link
     H) Home Screen
     S) Switch to Maps
     B) Back
Please select an option: G
Please enter a query: Best web browser

Stack Debug:
[Home->SafariHome->G:Best web browser
Current Screen: Google: Best web browser
Safari Options:
     G) Google Something
     F) Go to a favorite (bookmark)
     L) Follow a link
     H) Home Screen
     S) Switch to Maps
     B) Back
Please select an option: L
Please enter a link: chrome.com

Stack Debug:
[Home->SafariHome->G:Best web browser->L:chrome.com
Current Screen: chrome.com
Safari Options:
     G) Google Something
     F) Go to a favorite (bookmark)
     L) Follow a link
     H) Home Screen
     S) Switch to Maps
     B) Back
Please select an option: B

Stack Debug:
[Home->SafariHome->G:Best web browser
Current Screen: Google: Best web browser
Safari Options:
     G) Google Something
     F) Go to a favorite (bookmark)
     L) Follow a link
     H) Home Screen
     S) Switch to Maps
     B) Back
Please select an option: S

Stack Debug:
[Home->MapsHome->F:Microsoft Store->P:Target-Walmart->N:Target-
Walmart
Current Screen: Navigating from Target to Walmart
Maps Options:
     F) Find a place
     P) Plan a route
     N) Start Navigation
     H) Home Screen
     S) Switch to Safari
     B) Back
Please select an option: B

Stack Debug:
[Home->MapsHome->F:Microsoft Store->P:Target-Walmart
Current Screen: Planning route from Target to Walmart
Maps Options:
     F) Find a place
```

```
     P) Plan a route
     N) Start Navigation
     H) Home Screen
     S) Switch to Safari
     B) Back
Please select an option: B

Stack Debug:
[Home->MapsHome->F:Microsoft Store
Current Screen: Showing results for Microsoft Store
Maps Options:
     F) Find a place
     P) Plan a route
     N) Start Navigation
     H) Home Screen
     S) Switch to Safari
     B) Back
Please select an option: S

Stack Debug:
[Home->SafariHome->G:Best web browser
Current Screen: Google: Best web browser
Safari Options:
     G) Google Something
     F) Go to a favorite (bookmark)
     L) Follow a link
     H) Home Screen
     S) Switch to Maps
     B) Back
Please select an option: L
Please enter a link: firefox.com

Stack Debug:
[Home->SafariHome->G:Best web browser->L:firefox.com
Current Screen: firefox.com
Safari Options:
     G) Google Something
     F) Go to a favorite (bookmark)
     L) Follow a link
     H) Home Screen
     S) Switch to Maps
     B) Back
Please select an option: B

Stack Debug:
[Home->SafariHome->G:Best web browser
Current Screen: Google: Best web browser
Safari Options:
     G) Google Something
     F) Go to a favorite (bookmark)
     L) Follow a link
     H) Home Screen
     S) Switch to Maps
     B) Back
Please select an option: B

Stack Debug:
[Home->SafariHome
Current Screen: Safari Home
Safari Options:
     G) Google Something
     F) Go to a favorite (bookmark)
     L) Follow a link
     H) Home Screen
     S) Switch to Maps
     B) Back
```

```
Please select an option: F
Please enter bookmark name: 214 Sections Page
Stack Debug:
[Home->SafariHome->F:214 Sections Page
Current Screen: Safari Home
Safari Options:
     G) Google Something
     F) Go to a favorite (bookmark)
     L) Follow a link
     H) Home Screen
     S) Switch to Maps
     B) Back
Please select an option: B

Stack Debug:
[Home->SafariHome
Current Screen: Safari Home
Safari Options:
     G) Google Something
     F) Go to a favorite (bookmark)
     L) Follow a link
     H) Home Screen
     S) Switch to Maps
     B) Back
Please select an option: B //H would do the same thing

Home Options:
    S) Safari
    M) Maps
    Q) Quit
Please select an option: M

Stack Debug:
[Home->MapsHome->F:Microsoft Store
Current Screen: Showing results for Microsoft Store
Maps Options:
     F) Find a place
     P) Plan a route
     N) Start Navigation
     H) Home Screen
     S) Switch to Safari
     B) Back
Please select an option: H

Home Options:
    S) Safari
    M) Maps
    Q) Quit
Please select an option: Q

Sorry to see you go, tell the iPod I said hi!
```

Extra Credit

You will get up to 5 points extra credit for creating a JavaFX GUI, and up to 12 points for creating an Android App, 15 points for making a web application using JSP or JSF, 15 points for a really good Android app (with good animations and an on-screen iPhone, HW3 only special! Get it while it's hotter than your GPU after 30 seconds of Super Mario in 4k). Please discuss with your Grading TA if you choose to make an app, website or use Scene Builder, so you can coordinate submitting supplementary files with them. There is also extra credit for sorting (up to 5 points).
