COSC 1P03 Assignment 4

"What is a knap, anyway? And why does it need a sack?"

***Due: Friday Mar. 20, 2020 @ 4:00 pm** (late date Mar. 23 @ 4:00 pm)*

In preparation for this assignment, create a folder called `Assign_4` for the DrJava project for the assignment. The objective of this assignment is to develop a recursive solution to a problem.

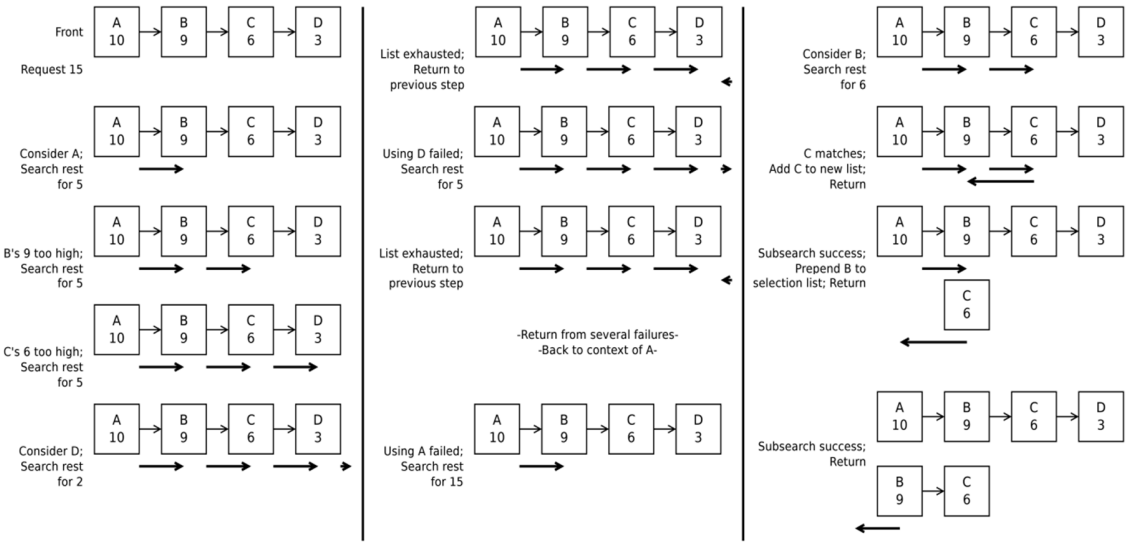
Problem

Has someone ever given you a gift card for a store you don't normally shop at? You might not want to waste money by leaving a balance in the account, but you also probably don't want to use your own money on merchandise you wouldn't normally purchase for yourself. So, what's the solution?

Try to find things to buy that add up to exactly the same balance as the gift card (e.g. for a $20 card, maybe you'd purchase one item for $16, and another for $4).

This is essentially a simplified case of the knapsack problem[1] (a classic example of combinatorial optimization). There are several ways to solve it, but we'll be looking for a recursive solution.

The algorithm is simple: if you have a collection of several potential items to purchase and `x` dollars available, consider selecting an item costing `y`, and see if you could find a selection within the remainder of the collection that costs `x-y` dollars. Recurse as necessary. If there is no combination within the rest of the collection that adds up to `x-y`, then give up on using the item costing `y`, and try again on the remaining collection for `x`.



[1] See: https://en.wikipedia.org/wiki/Knapsack_problem

Requirements

For this assignment, you'll be writing a program to load in an inventory of `Products` from an `ASCIIDataFile` (two such sample files are included).

The program will then use a `BasicForm` to:

- Let the user enter a target amount to spend
- This field should always have a default value of the sum total of all `Products`
- Let the user browse the inventory
  - i.e. display the inventory – and their costs – in a `TextArea`
- When the user chooses to buy, either display a selection of products adding up to the target, or indicate that no such selection exists
  - Purchased items **are not** removed from the inventory, but are limited to 1 per customer, so you may only choose an item once per query
- Halt when the user chooses to `Quit`

Hints

- Use appropriate procedural abstraction
- Each `Product` is a separate record (object)
- You'll probably want to use a linked list to hold the `Product` inventory, and another to hold any returned `Product` selections
- As a reminder, your solution must determine the selection **recursively**
  - You should have a recursive function, returning a list of the selection
  - Since the same `Product` can be a member of two lists, remember that that means you need to create new `Node`s for the returned list
  - Consider: how could one indicate that a selection wasn't possible?
- To avoid the hassle of rounding errors with floating-point numbers, all costs are integers
- Refer to the end of this assignment for a sample execution script

Submission:

Details regarding preparation and submission of assignments in COSC 1P03 are found on the COSC 1P03 Sakai Site as `Assignment Guidelines` under `Course Documents`. This document includes a discussion of assignment preparation, programming standards, evaluation criteria and academic conduct (including styles for citation) in addition to the detailed assignment submission process copied below. To prepare and submit the assignment electronically, follow the procedure below:

1. Ensure your submission folder (say `Assign_4`) for the assignment is accessible on your computer and contains the Dr. Java project and all associated java and class files for the assignment.
2. Create a `.zip` file of your submission by right-clicking on the top level folder (i.e. `Assign_4`) and selecting `Send to/Compressed (zipped) folder`. A zipped version of the folder will be created. Use the default name (`Assign_4.zip`). It is **important** that you only submit a `.zip` file, not `.rar` or `.tar` or any other type of compression. If you use a type of compression other than `.zip` your assignment may not be marked.
3. Log on to Sakai and select the COSC 1P03 site.

4. On the `Assignments` page select `Assignment 4`. Attach your `.zip` file (e.g. `Assign_4.zip`) to the assignment submission (use the `Add/Remove Attachments` button and select `Browse`). Navigate to where you stored your assignment and select the `.zip` file (e.g. `Assign_4.zip`). The file will be added to your submission. Be sure to check the `Honor Pledge` checkbox. Press `Submit` to submit the assignment.
5. Assignments incorrectly submitted will lose marks. Assignments without the required files may not be marked.
6. Sakai is set to allow 2 re-submissions. It is the student's responsibility to ensure the assignment is correct before submitting it. Only the last submission will be marked. Previous submissions will be disregarded.

DrJava

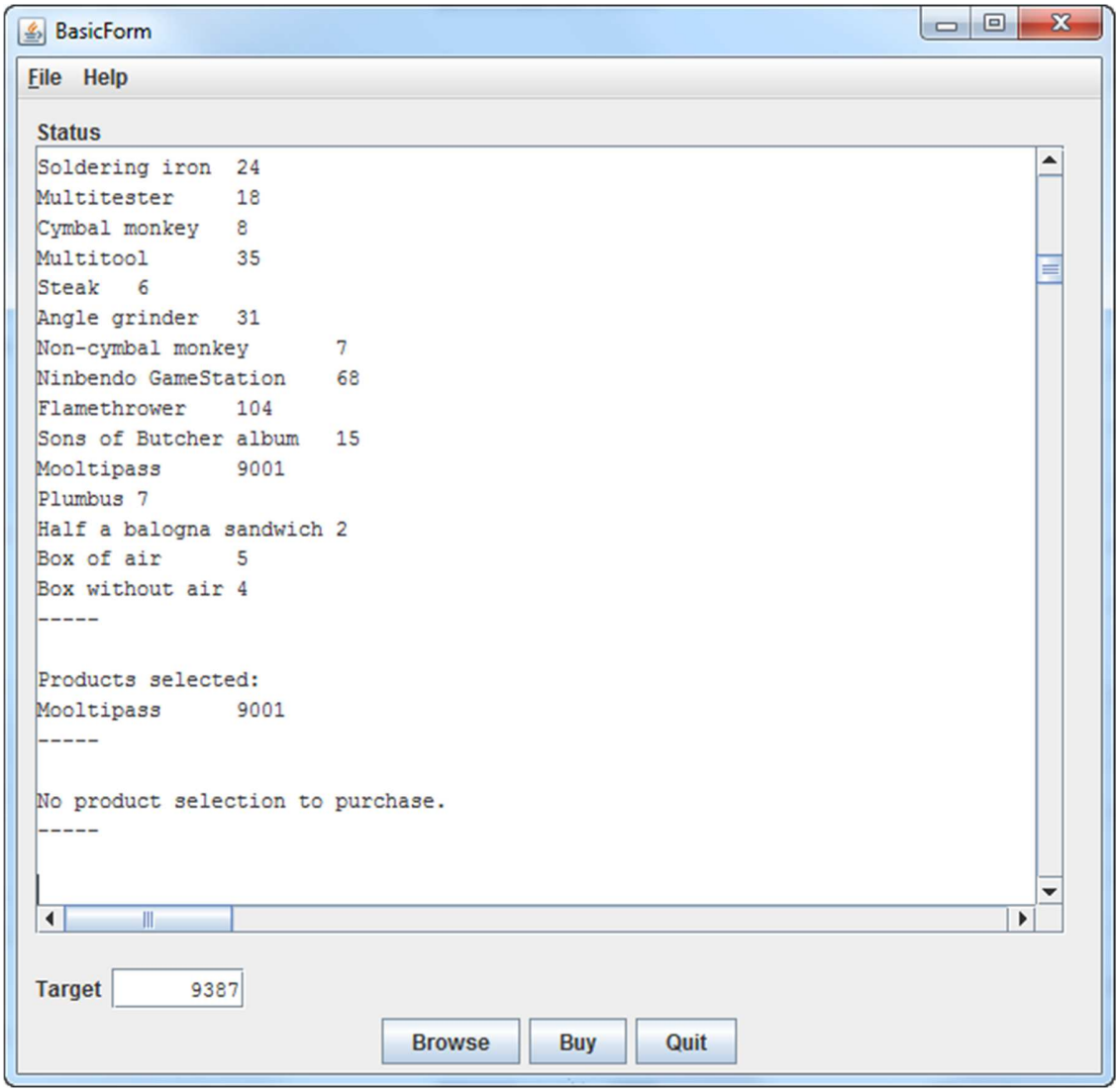
The `.zip` folder you submit should contain the project folder including all files relevant to the project—the `.drjava`, `.java`, and `.class` files for the assignment. If your project requires any special instructions to run, these instructions must be included in a read me file.

Other Platforms

Students must create their project using an IDE that is available on the Brock Computer Science lab computers. Currently, these are NetBeans, IntelliJ and Dr. Java. Markers must be able to open, compile and run your project on the lab computers. Assignments completed using some other IDE may not be marked. It is the student's responsibility to ensure their project will load, compile and run on a lab computer.

Submission Script

- When you first load the product catalog (`productcatalog.txt`), you should see a total value of `9387`
- Clicking `Browse` should show the full list of products
- Say your gift card is $40. Enter `40` into the `Target` text field and click `Buy`
  - It should have selected a `Hula hoop` and a `Megaphone`
  - The `Target` field should be reset back to `9387`
- Try `37` and click `Buy`
  - There were multiple potential combinations for `37`, but the algorithm as we've defined it should have selected `Hula hoop`, `Rope`, `Dollar`, `Steak`, and `Box of air`
- Try `4` and click `Buy`
  - This should have selected the last product: `Box without air`
- Try `0` and click `Buy`
  - You can't get anything for free, so this fails
- Try `9001` and click `Buy`
  - This will select a `Mooltipass`, but only after quite a bit of backtracking
- Try `387` and click `Buy`
  - This time, it can't meet the target


BasicForm
File Help
Status
Soldering iron 24
Multitester 18
Cymbal monkey 8
Multitool 35
Steak 6
Angle grinder 31
Non-cymbal monkey 7
Ninbendo GameStation 68
Flamethrower 104
Sons of Butcher album 15
Mooltipass 9001
Plumbus 7
Half a balogna sandwich 2
Box of air 5
Box without air 4

Products selected:
Mooltipass 9001

No product selection to purchase.

Target 9387
Browse
Buy
Quit