COSC 1P03 Assignment 4

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

Due: Mar. 24, 2017 @ 4:00 pm (late date Mar. 27 @ 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.

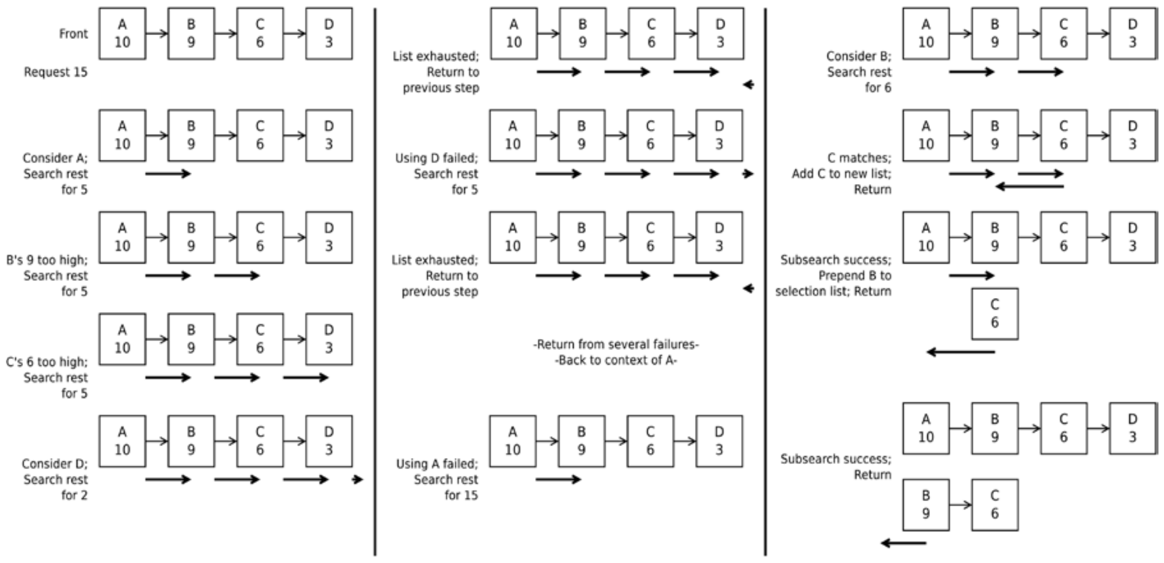
Problem

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`.



Front
Request 15
Consider A; Search rest for 5
B's 9 too high; Search rest for 5
C's 6 too high; Search rest for 5
Consider D; Search rest for 2
List exhausted; Return to previous step
Using D failed; Search rest for 5
List exhausted; Return to previous step
-Return from several failures-
-Back to context of A-
Using A failed; Search rest for 15
Consider B; Search rest for 6
C matches; Add C to new list; Return
Subsearch success; Prepend B to selection list; Return
Subsearch success; Return
A 10
B 9
C 6
D 3

[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 folder (say `Assign_4`) for the assignment is accessible on your computer.

2. Using DrJava, print (to `CutePDF Writer`) each of the `.java` files of your assignment using the name *`ClassName`*`.pdf` where *`ClassName`* is the class name (i.e. same name as the `.java` file) and save the `.pdf` files at the **top level** of the submission folder (i.e. directly within `Assign_4`).

3. Run the program using the data file `productcatalog.txt` following the script below. Before you press Quit on the display, select `File/Print Image of Window`... and print to `CutePDF Writer` as `output.pdf` at the **top level** of the project folder (i.e. directly within `Assign_4`).

4. The submission folder (`Assign_4`) should now include the `.java`, `.class` (created by DrJava) and `.pdf` files for the classes you wrote. It should also include the `.pdf` file for the display as above.
5. 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`).
6. Log on to Sakai and select the COSC 1P03 site.
7. On the `Assignments` page select `Assignment 4`. Attach your `.zip` file (e.g. `Assign_4.zip`) to the assignment submission (use the `Add 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. You should receive a confirmation email.
8. Assignments incorrectly submitted will lose marks. Assignments without the required files may not be marked.

DrJava

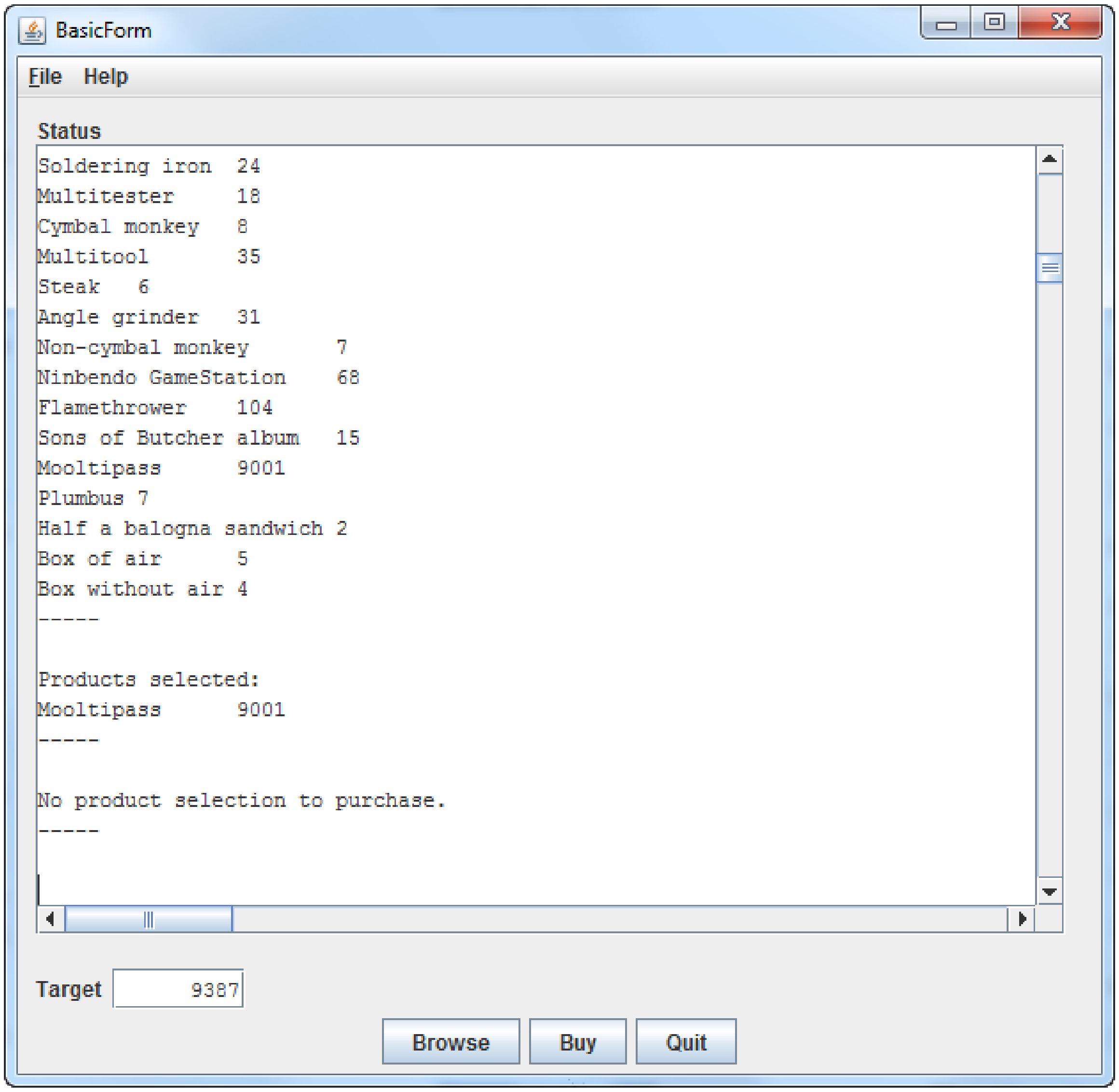
The `.zip` folder you submit should contain the project folder including all files relevant to the project—the `.drjava`, `.java`, `.jar` and `.class` files for the assignment and `.pdf` files for program listings and output.

Other Platforms

If you are using an IDE other than DrJava to prepare your assignment, you must include the `.java` source files and the `.pdf` files described above as well as a file (likely `.class` or `.jar`) that will execute on the lab machines.

Submission Script

- When you first load the product catalog, 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` 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



```
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