COSC201 Assignment 2: Managing the potionarium

Due: 11:59 p.m. Wednesday, May 10, 2023

Introduction

This assignment is in three parts:

- coding the business logic for a universal potion cabinet;
- coding an order system for building potions from it using both a standard algorithm, and one of your choice; and
- writing a short report, delivered as a pitch to the company you are working for as to why your work (and choice of algorithm) would justify retaining you to do further work.

The coding portion is worth 10 points total, and the report is worth 5 points. More details of the coding requirements are given below and in the javadoc for the provided classes. Correctness is the paramount consideration, but, particularly for the first part of the coding, efficiency will also be considered in grading.

The Potionarium

A *potionarium* or potion cabinet is a (presumably magical) connection to a near-infinite supply of potion ingredients organised in drawers. Each drawer is labelled by a non-negative `long` and may be empty or contain a set of potion ingredients which, for the purposes of this assignment are simply `String` values (e.g., `"eye of newt"`). Two ingredients are considered the same only if the `equals` method on `String` objects returns true (e.g., `"eye of newt"` and `"Eye of Newt"` are different ingredients).

The first part of the coding assignment is to write a `Potionarium.java` class that supports methods for accomplishing the following tasks:

- Determining the set of all available ingredients in the cabinet.
- Determining the list of drawers containing a given ingredient.
- Filling a drawer (if currently empty) with a set of ingredients.
- Adding an ingredient to a drawer.
- Removing one or more ingredients from a drawer (if present).

The detailed contracts of these methods are set out in the skeleton class provided.

First and foremost your methods must be correct both individually and when called in sequence. You do not need to allow for concurrent access to the cabinet – that is, you may assume that methods will be called one at a time. Secondly, it should not be possible for any outside program making use of the `public` methods of the class to make any changes to the contents of the potion cabinet, except those specified by the methods. Finally, your methods should be as efficient as possible in the worst-case sense. Ideally, they should all operate in constant time per item processed.

The Potionmasters

The main role of a potionmaster is of course to brew potions. Brewing potions starts with collecting the necessary ingredients from the potion cabinet. Each potionmaster is expected to develop their own style as they gain experience but, when learning potion making, trainees are introduced to *Standard Sam* a mythical potionmaster and their collecting technique.

Sam starts with a potion recipe: a list (without duplicates) of ingredients that are needed to brew a potion. First Sam verifies that it's possible to get all the required ingredients from the cabinet. Then, while they have not collected all the necessary ingredients, they find the first item on the list that's still needed. They remove this item from the lowest-numbered drawer containing it. While doing so, they also remove any other items still needed that are present in that drawer.

Finally, they construct an order report. The format of this report is the same for all potionmasters. If it was not possible to complete the recipe the order report is simply `"Missing ingredients: "` followed by a comma-separated list of the ingredients from the recipe that are not in the cabinet. If it was possible to complete the recipe then the report consists of a sequence of lines. Each line begins with the number of one of the drawers used, followed by a colon, a space, and then a comma-separated list of the ingredients taken from that drawer. The drawers should be listed in increasing order.

Your first programming task in this part is to implement Sam's procedure as a class `StandardSam.java` extending the abstract `Potionmaster` class.

Your second task in this part is to come up with a method for the aspiring potionmaster Novel Nic that you feel is superior to Sam's in some way, and implement it similarly as a class `NovelNic.java`.

Simple examples of expected output from Sam's method are provided with this assignment.

Code submission (10 points)

The code templates you'll be working from are available already. Code submission instructions will be given closer to the deadline.

Note that material from lectures 13 and 14 is (at least implicitly) used in this assignment, but the important part is understanding how Java's standard Collections framework represents sets and maps.

Written submission (5 points)

Your written report for this assignment is a presentation to the board of the Inderdimensional Potionmakers Authority. Its purpose is to convince them to keep you on retainer for further work on the back end of their management system. Since the board members are busy entities, your report (as a PDF file) should not exceed two sides of A4 (including any figures, charts or tables). Since they are generally elderly, the font size should be at least 11 point. Since they are thoughtful and analytical types, a presentation that draws on evidence from your work will be more effective (and get a higher grade) than empty rhetoric or unsubstantiated claims of excellence.