COSC 1P03 Assignment 3
"It's in the Bag"

***Due: Mar. 17, 2017 @ 4:00 pm** (late date Mar. 20 @ 4:00 pm)*

In preparation for this assignment, create a folder called `Assign_3` for the DrJava projects for the assignment. The objective of this assignment is to develop an implementation of an ADT.

Bag

A bag or multiset is a collection of things in which duplicates may occur. Think of a bag of marbles into which is placed 3 red marbles, two blue marbles and a yellow marble. There are 6 items in the bag, three of which are indistinguishable (all red), another two that are indistinguishable (both blue) and one other. Additional marbles can be added and marbles can be removed. If you look into the bag, you can remove a marble of a particular colour (for example any of the 3 indistinguishable red ones). Alternatively, you can simply reach into the bag and remove any marble, without knowing the colour until you remove it. Again, if you look into the bag, you can determine whether or not there is a marble of a particular colour or how many marbles of a particular colour there are.

Thus `Bag` is an abstract data type (ADT) representing collections of things and which has a set of operations as described above.

Part A – `Bag` ADT

The `Bags` folder contains a DrJava project for the `Bags` library package. It contains the interface for the `Bag` ADT as `Bag.java`.

As part of a package called `Bags`, write the necessary Java classes to provide the `Bag` ADT. The interface for the `Bag` ADT is provided as `Bag.java`. You must write the exception classes: `NoSpaceException` and `NoItemException` and one implementation class: `ConBag`.

For this assignment, the items in the bag are `String`s. That is, in the example above a red marble might be represented by the `String` `"red"` and placing 3 red marbles into a `Bag` called `aBag` could be done by:

```
aBag.add("red");
aBag.add("red");
aBag.add("red");
```

The implementation class `ConBag` is a contiguous (array-based) implementation of the `Bag` interface. The items in the bag are represented by a "variable-sized" array of `String`s. Items added to the bag are added at the end of the array. When an item is removed, the items following it in the array are moved one position to the left. The selection of an arbitrary item from the bag is done by the random generation of an index position within the array. When comparing `String`s for equality, be sure to use the `equals` method.

The `ConBag` class should provide at least three constructors. The default constructor (no parameters) creates a new empty bag with capacity of 100 items. A constructor with an `int` parameter creates a new empty bag with capacity given by the parameter. A constructor that takes an array of `String`s as a parameter should **copy** all of the elements from the array into the new empty bag whose capacity is 100 items.

Creating a library `.jar` file

In order to use (`import`) a library package in other projects, a `.jar` (Java Archive) file must be created. Be sure that the project has first been compiled. In the project menu, select `Create Jar File From Project…`. In the resulting dialog, check `Jar classes` and then click on the … button beside `Jar File`. Navigate to the folder within which you want to create the archive file (in this case, probably the folder for the `Bags` project itself), fill in the file name (should be the package name: `Bags.jar`) and press `Save`. Now press `OK` to dismiss the dialog.

Part B – Testing the `Bag` ADT

As a package `TestBags` within your `Assign_3` folder, write a test harness to fully test the `ConBag` implementation of `Bag`. Be sure to test all limiting cases (including a full bag) and test and catch all exceptions.

Making a library accessible to a project

To make use of a library package that is not part of the standard libraries, an additional classpath must be added to the project to enable the compiler to find the library. Before attempting to compile the project, select `Project Properties` from the `Project` menu. In the resulting dialog, press `Add` under the `Extra Classpath` box. Navigate to the folder containing the `.jar` file for the desired library (the `Bags.jar` file in this case), select the `.jar` file and press `Select`. The classpath for this file should now be in the `Extra Classpath` box. Press `OK` to dismiss the dialog.

Part C – Using the `Bag` ADT

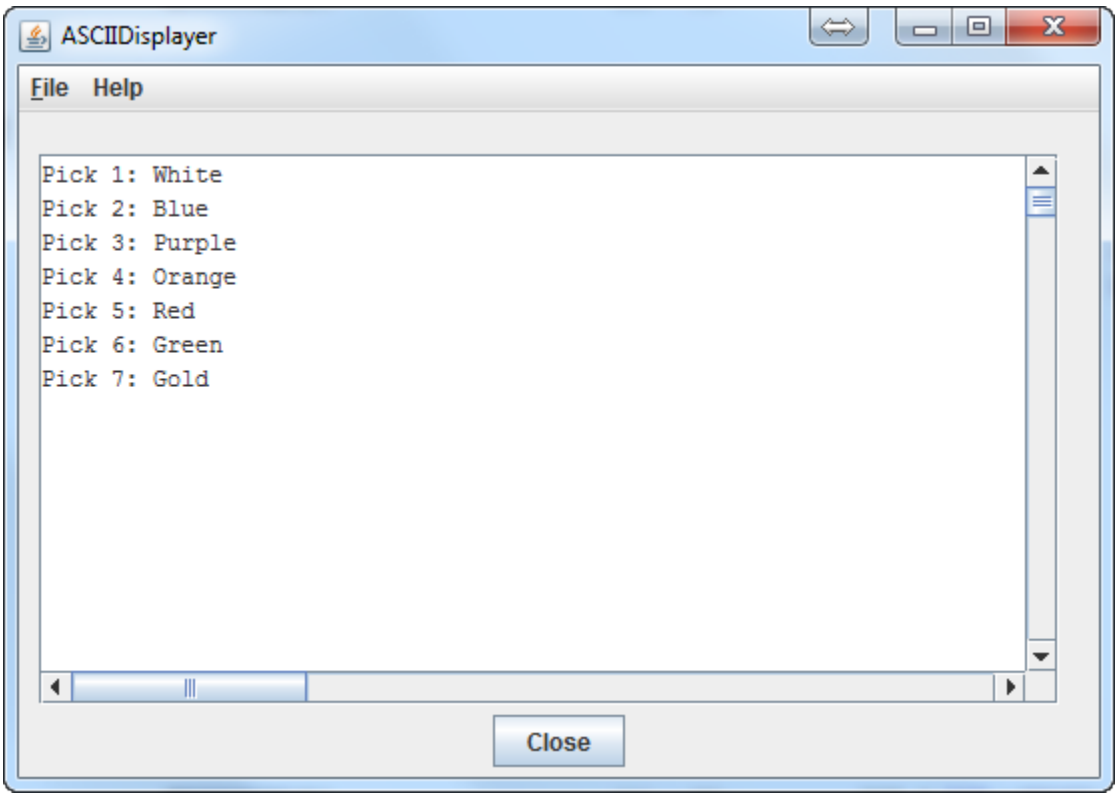
As a package called `Draft` within your `Assign_3` folder, write a Java program to simulate the entry draft for a hockey Little League.

The local little league is holding its annual entry draft to select new players for the coming season. The league consists of a number of teams, each team named by a different colour. The standings at the end of the previous season determine how many tokens each team gets going into the entry draft. Teams with a lower standing are given more tokens than teams with a high standing. For example, the standings at the end of the year and the tokens are given below:

Team	Place	Tokens
Red	1	1
Gold	1	1
Purple	3	2
Blue	3	2
Orange	5	3
Green	6	4
White	7	5

At the start of the draft all teams put their tokens into a box. The league commissioner reaches into the box, randomly selecting a token. The team whose token was selected then chooses a player and the remainder of the tokens for that team are removed. The commissioner then selects another token and so on until all teams have selected a player. This method makes it fair for lower ranked teams which have a better chance that they select first.

Use a `Bag` to represent the box where the team names (colours) will be the items in the bag. Use the `ConBag` implementation. The `ASCIIDataFile` `standings.txt` contains one line per team giving the team name (`String`) followed by the number of tokens (`int`). Use this to load the `Bag`. The output from your program should be to an `ASCIIDisplayer` listing the order of the draft selections by team name such as below:



Don't forget to add the `Bags.jar` file to the `Extra Classpath` in the `Project Properties` as above.

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_3`) for the assignment is accessible on your computer. It should have three subfolders for the three projects: `Bags`, `TestBags` and `Draft`.
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_3`).
3. Run the `Draft` program. When the program terminates, 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_3`).
4. The submission folder (`Assign_3`) should now include the `.java`, `.class` (created by DrJava), `.jar` 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_3`) and selecting `Send to/Compressed (zipped) folder`. A zipped version of the folder will be created. Use the default name (`Assign_3.zip`).
6. Log on to Sakai and select the COSC 1P03 site.
7. On the `Assignments` page select `Assignment 3`. Attach your `.zip` file (e.g. `Assign_3.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_3.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.