Instructions:
==========

Please find attached a zip file containing some classes and sample JSON files that you will need for the following exercises.

Exercise 1:

A wine is made up from grapes of different sources after being blended at the winery. These grapes have different properties such as year, variety (e.g. Chardonnay, Pinot Noir) and region (e.g. Yarra Valley, Mornington Peninsula, Macedon Ranges, etc). When bottling a wine, the winery needs to know the percentage breakdown of these properties so they know what they can legally claim on the wine's label.

Some basic classes have been included in a zip file. Your mission is to complete the four methods in WineTest.java to print a breakdown of the year, variety, region and year + variety information for the wine. It needs to be printed from highest percentage to lowest and we only want to see one line printed for each unique year in printYearBreakdown(...), one line for each unique variety in printVarietyBreakdown(...), one line for each unique region in printRegionBreakdown(...) and one line for each unique combination of year + variety in printYearAndVarietyBreakdown(...).

We need a Java implementation that will generate a nice printable composition.

For example, in printYearBreakdown(...) we want to see something like:

85% - 2011
15% - 2010

... and so on for the other methods. Each method shows the percentage of the unique property along with the unique properties sorted from highest percentage to lowest.

And a similar breakdown by variety in printVarietyBreakdown(...), region in printRegionBreakdown(...) and combinations in printYearAndVarietyBreakdown(...)

Hint: feel free to add any extra classes / methods as necessary, but don't re-invent the wheel if there are algorithms in the standard Java SDK that will already do some of the work for you.

Exercise 2:

Provide a simple user interface in JSF that would allow a user to view the basic details of a wine, as well as allowing the user to see the Year, Variety, Region and Year + Variety breakdown. For this purpose, assume that a Wine contains the following bits of information that are important to a user (in order of importance):

Assume the most important details that a user will want to see when looking at a wine are:

Lot code (text, usually 10-12 characters) [reference code used throughout a wine's production]
Description (text, usually about 40-60 characters) [human readable description of the wine]
Volume (floating point number, but rounded to zero places) [how much wine in litres there is of this wine]
Tank (text, usually about 5-6 characters) [the code of the physical tank that the wine is currently in]
Product state (text, usually 15-20 characters) [the state that the wine is in... changes throughout production]
Owner (text, usually about 15-20 characters) [who the owner of the wine is]

Although not as important as the information above, the wireframe should take into account that the user may also want to see the year, variety, region or combined year + variety breakdown composition as detailed in the first exercise above. Give the user a control to switch between viewing the different composition types.

The composition details can get quite long in some cases. When having a glance at the data a user will always want to see the compositions in order of the percentage they make up (highest first). Assume that if the composition breakdown is long, that a user might only want to see the first 5 elements, but would expect to have a Show More option to see more details.

Assume that when a user is viewing the details of a wine, they may want to edit the description or product state. There would typically be other options on this view to record an operation against the wine (such as blending into another tank). No need to show an implementation of one of these operations, but take into account that this would be the type of action that a user may want quick access to from this screen.

The project includes some JSON samples. Look at the data in the JSON files to give you an idea of any potential edge cases that may have an impact on the mock-up / design. These JSON files follow the class structure of the Wine.java sample class. Assume that when I am looking at your submission that I will be using these to test the composition breakdown methods in the first question, and I will be looking to see that the mock-up considers any edge cases based on the JSON data. Ideally your JSF backing bean should be able to load a Wine object from any of the JSON sample files so I can test it against the 3 different files.

Please use maven to package up your sample application and dependencies so it can be installed and run with a single maven command such as::

```
mvn spring-boot:run
```

Please submit back as a GitHub/GitLab repository link including a README file containing exact command required to run the sample.