COSC 1P03 Assignment 3
“A Pareto Set”

***Due: Mar. 6, 2020 @ 4:00pm** (late date Mar. 9 @ 4:00 pm)*

The objective of this assignment is to implement a class for an abstract data type (ADT) to an interface. In preparation for this assignment, create a folder called `Assign_3` and two subfolders: `Pareto` and `TestParetoSet` for the DrJava projects for the assignment.

Pareto Set

For this assignment, we'll be looking to create an Abstract Data Type (ADT) to assist with collecting and filtering observations from experiments. Specifically, we'll be defining a *collection* of observations where each observation has a label (to identify the observation) and a number of readings (observed values). Our ADT will be called a *Pareto Set* defined by the interface `ParetoSet` and the observations represented by the class `Observation`.

Background

To understand what a Pareto set is, we'll need to build up its underlying premise.

Dominance

Suppose we have two `Observations`: `[A,1,3]` and `[B,2,4]` where the first value (A, B) is the label and the rest are the readings (i.e. values observed). Further suppose we're doing some experiment wherein “lower is better”. In that case, which observation is “better”? Clearly the first. But what if the observations were `[C,1,4]` and `[D,2,3]`? Perhaps they're equal, since they have the same sum? However, for some problems, we can't assume that each reading carries the same weight so adding them makes no sense.

For these cases, we define a relation—*dominance*—to indicate that one observation is clearly better than the other. We define *dominance* as:

A dominates B iff:

- each reading within A is less than or equal to the reading at the same position within B
  and
- A has at least one reading that is strictly less than the reading at the same position within B

If A and B are identical (i.e. all readings equal), or if each has at least one reading that beats the other, then neither dominates the other.

With the examples above, neither `[C,1,4]` nor `[D,2,3]` dominates each other. However, `[A,1,3]` dominates all three of the other observations.

Pareto Set

When performing experiments, many observations (sets of readings) might be collected. If the readings are the result of attempts towards optimization, for example, then only the “most optimized” results will warrant further consideration. To that end, when presented with a very large collection of observations, we may wish to filter them, to only retain the "best" observations. If, as above, we can't combine the values of the readings for comparison, we'll instead rely on dominance.

We define the *Pareto Set* as being the set of observations such that no observation is dominated by any other observation within the set.

With the four observations above, the Pareto Set contains the single observation `[A,1,3]`. The other three observations are all dominated by `A`, and thus not included. Normally, Pareto Sets consist of more than a single observation, so let's look at a more interesting example.

Multi-objective Optimization

A common reason to use Pareto sets in Computer Science is when we want to perform some form of *combinatorial optimization* and want to minimize multiple different kinds of costs. One such example is the *vehicle routing* problem: if a delivery company wishes to drop off packages all across a city, then the company will want to choose routes that minimize both the number of required vehicles and fuel costs (the readings for an observation). However, those two costs tend to compete with each other (you either have one vehicle zigzag, costing extra fuel, or you need to send out more than one vehicle). Consider the following observations where the first reading is number of vehicles and the second is total distance travelled (giving the fuel cost).

```
[A,5,200] [B,4,160] [C,4,210] [D,5,170] [E,7,170] [F,8,102] [G,7,120]
[H,7,245] [I,6,140] [J,6,240] [K,9,102] [L,9,260] [M,2,260] [N,3,225]
```

Which observations are part of the Pareto Set?

- `M`, because even though it has the largest distance, it uses the fewest number of vehicles
- `F`, because it ties with `K` for the shortest distance, but has a lower number of vehicles
- `N`, `B`, `I`, and `G` also are in the Pareto Set

Why aren't the others?

- `K` is tied with `F` for the shortest distance, but `F` uses fewer vehicles (`F` dominates `K`)
- `D` is beat in both readings by `B` (`B` dominates `D`)
- `C` ties with `B` for number of vehicles, but is beaten for distance (`B` dominates `C`)
- likewise `A`, `E`, `H`, `I`, `J`, `L` are all dominated by `B`

Pareto Set as an Abstract Data Type (ADT)

What we want is a collection ADT that *will only retain those observations that belong in the Pareto set*. That is, as each observation is added, it will not be retained in the set if it is dominated by any reading already within the set. And, if the new observation dominates any observations already present in the set, those dominated readings are removed.

The assignment folder includes:

- `Observation.java`—a class defining the observation as having a label and a set of readings.
- `ParetoSet.java`—an interface defining the `ParetoSet` ADT

- `DimensionalityException.java`— an exception that indicates that the observation being added has a different number of readings than the `ParetoSet` is expecting or if an attempt is made to retrieve the observation with lowest value for a reading when the reading index is out of bounds.
- `UnderflowException.java`—an exception that is thrown if there is an attempt to retrieve an observation from an empty `ParetoSet`.
- `dataForSubmission.txt`—data to be used for submission of the assignment. Contains: number of observations (`int`), number of readings per observation (`int`), for each observation: label (`char`), readings (`double`). It is the data described above.

Implementation

There are two phases to this assignment:

1. develop a *linked* implementation of `ParetoSet` called `LnkParetoSet`. Your implementation *must* follow the specification exactly, including exceptions. The `ParetoSet` interface may not be modified.
    - Include a constructor:
      ```
      public LnkParetoSet ( int nReadings )
      ```
      that creates an empty `LnkParetoSet` where `Observations` have `nReadings` readings.
    - Implement the method `toString` that returns a string consisting of the labels of the observations in the `LnkParetoSet` separated by commas and enclosed in square brackets such as: `[M,I,G,B,F,N]`
2. write a *test harness* as part of a package `TestParetoSet`. It should “put the collection through its paces” including:
    - load an `ASCIIDataFile` (`testData.txt`) of `Observation` data to populate the `ParetoSet` and then display the resulting set. Verify all results are as expected.
    - trigger each *exception* that `ParetoSet` can throw.
        - of course, your test harness must also *catch* those exceptions, since your tests can't crash
    - in addition to the provided data examples, it's highly advisable to create your own, to test everything (as the markers might be using different data files)

Tips:

- The algorithm for adding an observation is simpler than it might seem at first. Working it out on paper first is helpful.
    - You could consider 2 phases
        1. See if the observation should be added at all (i.e. that it isn’t dominated by any other observation)
        2. See if the observation should cause other observations to be removed (the observation dominates the other observation)
- When implementing `add` you might find it easier to create a new list of observations that are part of the Pareto Set rather than modifying the current one.
- Though you shouldn't add any more *public* methods beyond what's promised by the interface, clean, well-written code will definitely have a couple *private* ones such as a method to determine if one observation dominates another.

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 (`Assign_3`) contains two folders `Pareto` and `TestParetoSet` for the 2 DrJava projects 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_3`) and selecting `Send to/Compressed (zipped) folder`. A zipped version of the folder will be created. Use the default name (`Assign_3.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 3`. Attach your `.zip` file (e.g. `Assign_3.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_3.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.

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.