Lab lecture exercises – Week 8

First we have to set up Eclipse so that it recognizes JavaFX. In order to do so, do the following steps in Eclipse. [Corresponding steps on the command line are as explained in the lecture.]

- First create a new project, e.g., **`lab08`** (do not create a module).
- Right click on the new project and follow the descriptions: BuildPath → Configure build path → Libraries → Add external JARs.
- Navigate to **`/bham/modules/roots/msc-sw/2019-20/lib/javafx-sdk-11.0.2/lib/`** and add all the **`.jar`** files.
- Click on **`Apply`**.

In order to run the class, change the **`Run configurations`** (right to the arrow to run the code):

- Arguments → VMArguments add:
  `--module-path /bham/modules/roots/msc-sw/2019-20/lib/javafx-sdk-11.0.2/lib/`
  `--add-modules=javafx.controls`

On the Canvas page you find files in **`lab08-fragment.zip`**. The files are also available locally from **`/home/staff/mmk/dropbox/wk08/`**. You find:

1. A class **`Measure.java`** with two field variables **`private String description`** and **`private double value`**, where a value is always non-negative (for instance, this could be films with a rating, bank accounts with a non-negative balance, processes with times needed to complete them, customers and their age, and so on).
2. A fragment of a class **`DisplayBarChart.java`** for the visual presentation of measures by creating vertical bars (see, e.g., `http://www.mathsisfun.com/data/bar-graphs.html`) from a given **`private ArrayList<Measure> measures`**.

Load the two files into the **`src`** folder of the project.
Complete the class **`DisplayBarChart.java`**. To this end:

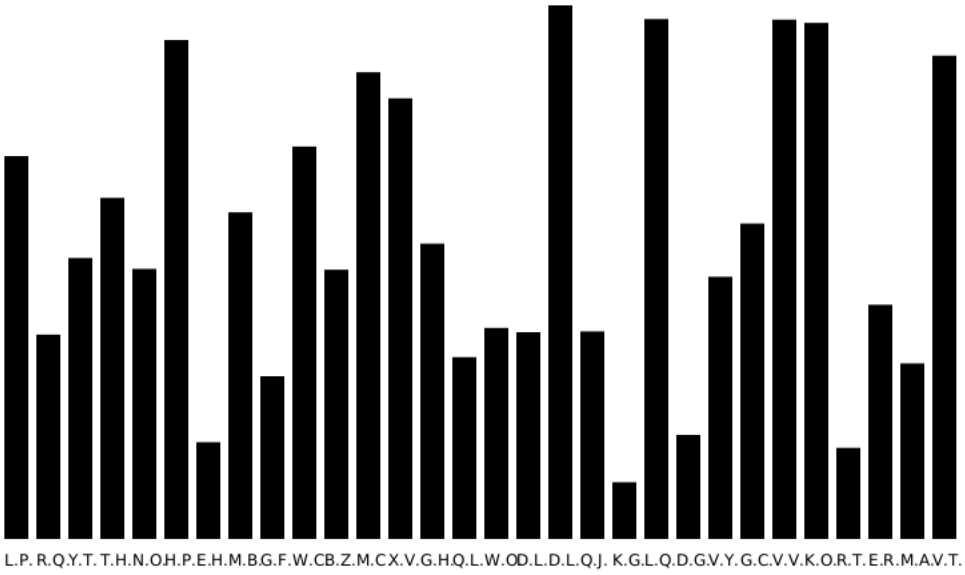
1. Compute the maximum of **`measures`**. If it is non-zero, normalize the values so that the maximal bar is represented by a given number of pixels such as **`int yNumberOfPixels = 400`**. The scene should be slightly higher. The width should be adaptable to the number of values, but not be much bigger than `800` pixels.

2. Write a method **`public static ArrayList<Measure> randomMeasures(int n, double low, double high)`** that generates an ArrayList of type **`Measure`** with length **`n`** and random values between **`low`** and **`high`**.

3. For a short ArrayList of size less than or equal to 10 present the bar chart by bars of fixed width **`30`** pixel and the empty space between two bars of **`10`** pixels.

4. If the ArrayList is bigger (i.e., sizes greater than 10 but less than or equal to 200) reduce the width of the bars and the gaps down to **`3`** and **`1`** for an ArrayList of size 200; and to something in between **`3`** and **`30`** for the width of the bars (in between **`1`** and **`10`** for the gaps) for not quite so long ArrayLists. In any case the total width should not be much bigger than **`800`** pixels.

5. For ArrayLists with sizes up to 30, display a **`description`** below the bars.



6. If the ArrayList contains more than 200 elements, but fewer than or equal to 600 elements) use a filled **`Polygon`** to display the values.

7. If the ArrayList is even bigger (more than 600 elements) print just out a warning that the ArrayList is too long and will not be displayed.