Tarea 2 - JConsolePlot

Profesor: Alexandre Bergel
Auxiliar: Sebastián Ferrada

En esta tarea usted deberá implementar distintos tipos de gráficos para ser mostrados por consola, con argumentos indicados por el usuario. Su aplicación, además de contar con todas las funcionalidades indicadas, debe cumplir con las buenas prácticas de programación y ser desarrollada en Java usando los patrones de diseño vistos en clases.

1. Requisitos

Se requieren dos tipos de gráficos que se impriman por consola. Cada gráfico tiene sus propiedades y parámetros, los cuales serán definidos a continuación:

- **BarPlot:** Corresponde a un gráfico de barras. Recibe datos de la forma (X, Y) tal que X es una categoría e Y una frecuencia. Luego, dibuja un gráfico cuyo eje horizontal presenta las categorías X(en orden de aparición de los datos) y sobre ellas una barra, cuya altura indica el valor de la frecuencia Y. El eje vertical está etiquetado según estas frecuencias. Puede encontrar un ejemplo en la Figura 1.
- **ScatterPlot:** Un scatter plot es un gráfico que muestra la dispersión de los puntos de interés en el espacio cartesiano. Recibe tuplas de datos (X, Y) tales que X e Y son coordenadas, entonces el scatter plot mostrará una marca en cada punto (X, Y). Ambos ejes están etiquetados convenienemente según los datos ingresados. Puede ver un ejemplo en la Figura 2.

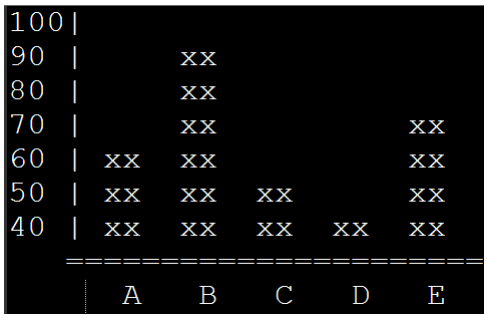

Figura 1: Gráfico de barras

El tipo de gráfico y los datos para ellos se deben especificar a través de los argumentos del programa. El primer argumento debe indicar el tipo de gráfico que se desea construir `BarPlot`, `ScatterPlot` o `LinePlot`. A continuación se especifican los datos a utilizar. Esto se puede hacer de dos formas:

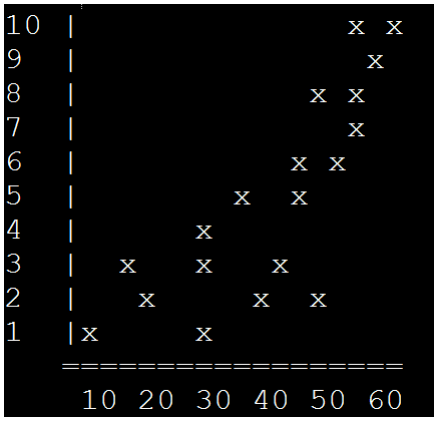

Figura 2: Scatter plot

- Directamente como argumentos: usar el comando -P y listar a continuación las tuplas a graficar separadas por espacios, por ejemplo: `ScatterPlot -P 3,4 2,1 1,0`.
- Indicando un archivo con los datos: usar el comando -F y a continuación la ruta del archivo con los datos. Cada línea del archivo debe contener una tupla de datos, cada tupla tiene dos valores separados por comas. Ejemplo: `BarPlot -F /data/cc3002/plot.txt`.

Finalmente, es necesario indicarle al programa la forma de ajustar los ejes. Para esto hay dos opciones, indicar que se debe ajustar a los datos mínimo y máximo por cada eje, o bien indicar manualmente el valor máximo del eje. La primera opción se ejecuta por defecto. La segunda, mediante los comandos `-y` y `-x` para indicar el valor máximo del eje respectivo. Por ejemplo, `BarPlot -F /data/d.txt -x 9` indica que se usará un BarPlot con los datos de `/data/d.txt` y que el eje x está acotado por 9, el eje y está acotado por el máximo presente en los datos.

2. Requerimientos Adicionales

Además de una implementación basada en las buenas prácticas y patrones de diseño vistos en clases, usted debe considerar:

- **Cobertura.** Cree los tests unitarios, en JUnit 4, que sean necesarios para alcanzar un 90 % de coverage por paquete. Estos tests deben estar en la carpeta `test` de su proyecto.
- **Javadoc y convenciones.** Cada clase pública y cada método público deben estar debidamente documentados. Tanto su código como la documentación deben seguir también las convenciones indicadas en `http://google.github.io/styleguide/javaguide.html`.
- **Resumen.** Debe entregar un archivo `pdf` que contenga su nombre y un diagrama UML que muestre las clases y métodos que usted definió en su proyecto.

3. Evaluación

- **Código fuente (3.5 puntos)** Se analizará que su código provea la funcionalidad descrita y esté implementada utilizando un buen diseño.
- **Tests (1 punto)** Sus casos de prueba deben crear diversos escenarios y contar con un coverage de 90 % por paquete
- **Convenciones (1 punto)** Cada clase y método público debe ser documentado. Su código debe seguir las convenciones indicadas en la sección anterior. Se descontará por cada falta.
- **Resumen (0.5 puntos)** El resumen mencionado en la sección anterior. De no enviarlo, se considerará que no utilizó un buen diseño y será penalizado.

Para la evaluación del coverage y de la documentación utilizaremos un sistema automático llamado El Chango[1], donde usted deberá registrarse como usuario. Este sistema estará disponible a la brevedad para que puedan probar sus tareas (más detalles serán dados en el Foro del curso). En él, usted sube su carpeta `src` en un archivo `zip` y se generará automáticamente un reporte indicando su nota máxima posible y una serie de indicaciones sobre cómo mejorar las convenciones, testing y documentación de su programa.

4. Entrega

Usted debe subir a U-Cursos o bien el proyecto de eclipse realizado o bien las carpetas `src` y `test` de su proyecto. El plazo de entrega es hasta el 26 de mayo a las 23:59 hrs. No se aceptarán peticiones de extensión del plazo.

[1] `http://cc3002.dcc.uchile.cl/`