# **Fundamentos de Programación**

## ***Práctica Final – 2019-1***

Este documento describe la práctica final de la asignatura Fundamentos de Programación, del semestre 2019-1.

## ***Objetivo General***

El objetivo de este trabajo es poner en práctica los conocimientos adquiridos en el curso, realizando procesos de análisis, diseño, pruebas, documentación e implementación de una aplicación.

## ***Objetivos Específicos***

En la práctica se deben poner en evidencia las siguientes habilidades:

1. Capacidad de esquematizar en un *diagrama de clases* las entidades que conforman la solución del problema.
2. Capacidad de *implementar* un programa en Java que funcione correctamente y que sea acorde al diagrama de clases elaborado.
3. Capacidad para diseñar y realizar *pruebas* de la aplicación
4. Capacidad para generar la *documentación* del código utilizando JavaDOC

***Tema para el trabajo***

Para la realización de la práctica, cada estudiante debe hacer una implementación de lo que llamaremos un *marco de datos* (en inglés *Data Frame*). En un marco de datos se cuenta con un conjunto de datos que se capturan respecto a algún fenómeno que se quiere estudiar. Vamos a trabajar con la temperatura y precipitación diarias en Medellín desde el 1 de enero de 2000, reportadas por el NOAA (*National Oceanic and Atmospheric Administration -* <https://www.ncdc.noaa.gov/>). En otro adjunto se explica cómo bajar datos similares para muchas ciudades del mundo y en otros rangos de fechas.

Sobre el marco de datos se pueden realizar cuatro tipos de operaciones:

1. ***Operaciones estadísticas básicas***, tales como promedio, mínimo, máximo, moda, desviación estándar, conteo, etc. Estas operaciones se realizan sobre los valores de la variable que se mide.
2. ***Operaciones de filtro***, tales como: mayor que un valor, menor que un valor, igual a un valor, número del sensor, etc. Estas operaciones producen un nuevo marco de datos con los valores del marco de datos original que cumplan con la condición determinada.
3. ***Operaciones de visualización***, tales como: diagrama de líneas, diagrama de puntos, diagrama de barras, diagrama de torta, etc. (esta parte es **opcional**)
4. ***Operaciones de entrada/salida***: leer un marco de datos desde un archivo o escribir el marco de datos actual en un archivo con un nombre determinado.

El código de la práctica debe incluir por lo menos una clase para guardar cada dato registrado (se debe llamar Dato) y una colección para guardar una colección con los datos que conforman el marco de datos (se debe llamar MarcoDeDatos). La clase MarcoDeDatos debe tener por lo menos métodos para leer los datos de un marco de datos de un archivo texto y uno para grabarlos. Usted, como programador, decide en cuál clase incluir los métodos para las operaciones de filtro, las estadísticas y las de visualización.

La interfaz con el usuario debe ser tipo texto. Esto es, no se debe utilizar elementos de interfaz gráfica como menús, cajas de diálogo y similares. Por este motivo, los comandos que ingresa el usuario se leen desde la consola. Usted, como programador, diseña los comandos para la interacción con el usuario.

Para la visualización de los datos *sí* se puede utilizar un JFrame y un JPanel, como se hizo en el curso en la semana 8. Por sencillez, y si usted como desarrollador lo quiere hacer así, puede hacer la visualización desde un programa aparte, naturalmente también escrito en Java.

La interfaz debe permitir cargar estaciones a las que se les puede aplicar los diferentes métodos u operaciones descritas en este documento.

Generar opción para imprimir listado de estaciones mostrando información del nombre de la estación, promedio total de precipitaciones, promedio total de temperatura, temperatura máxima y temperatura mínima. La impresión debe verse en forma de tabla

La funcionalidad mínima que debe tener su aplicación se describe en la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| ***Tipo de operación*** | ***Funcionalidad mínima*** | ***Ejemplos*** |
| Entrada/salida | Leer de un archivo  Escribir en un archivo |  |
| Filtro | Por lo menos dos de las operaciones de la siguiente columna | Menor que  Mayor que  Menor o igual que  Mayor o igual que  Igual a  Diferente de |
| Visualización | Por lo menos dos de las operaciones de la siguiente columna (ver ejemplos más abajo) | Diagrama de líneas  Diagrama de puntos  Diagrama de barras  Diagrama de tortas  (Para los dos últimos el usuario define unos rangos para los datos y en el diagrama se indica la proporción de los datos que está en cada rango) |
| Estadísticas | Por lo menos dos de las operaciones de la siguiente columna | Contar  Mínimo  Máximo  Promedio  Moda  Desv. Estándar |

***Entregas***

Hay dos entregas relacionadas con la práctica. En la primera entrega se avanza hacia la solución final del problema y en la segunda se completa la solución.

### **Entrega 1 (se debe subir por Interactiva Virtual a más tardar el 8 de mayo)**

En la entrega 1 el estudiante demuestra que comprende el problema y avanza hacia una primera solución.

Entregables:

* Especificación de los *requisitos* del programa: Elaborar, en más detalle, qué funcionalidad le brinda la aplicación al usuario
* *Diagrama de Clases* creado en StarUML, ArgoUML o un software similar.
* *Diagrama de objetos* ilustrando los datos que se van a utilizar y las relaciones entre ellos.
* *Mockups* de la interfaz con el usuario. Se debe dar ejemplos de como será la interacción con el usuario. Hay que recordar que la interfaz es tipo texto (no GUI).
* *Algoritmos* más importantes de la aplicación, especificados en pseudo-código e implementados en Java (por lo menos un método para la funcionalidad de filtro, uno para la funcionalidad de visualización y uno para la funcionalidad estadística). Por ejemplo:
  + Método para hacer un filtro *menor que*.
  + Método para hacer un diagrama de puntos con los datos del marco de datos
  + Método para contar cuántos elementos tiene un marco de datos
* Diseño de (los) *archivo*(s) de *texto* desde los cuales se van a leer y escribir los datos de la aplicación.

### **Entrega 2 (se debe subir a Interactiva Virtual a más tardar 31 de mayo)**

En la entrega 2 el estudiante completa la implementación y las pruebas de la aplicación.

Entregables:

* *Diagrama de Clases* detallado creado en StarUML, ArgoUML o un software similar, es posible hacer modificaciones al diagrama de la entrega anterior.
* *Diagrama de Objetos.* Es posible hacer modificaciones al diagrama de la entrega anterior.
* Agregar, al programa de la entrega 1 lo siguiente:
  + Código para la interacción con el usuario
  + Código para leer y escribir un marco de datos, con la respectiva interfaz con el usuario
  + Código para hacer por lo menos dos operaciones por cada tipo de funcionalidad.
* Código de la aplicación completa y funcionando (el código se entrega en un proyecto bien sea de NetBeans, o BlueJ).
* Documentación del código utilizando JavaDoc. Es importante documentar cada método y cada atributo público de cada clase.
* Documentación de los procesos de pruebas realizadas.

**Evaluación Final**

Para realizar la sustentación final, los estudiantes de cada grupo se citarán en un salón y cada estudiante deberá llevar a cabo una serie de modificaciones que indicará el profesor. Naturalmente cada estudiante debe conocer muy bien su código para poder realizar los cambios solicitados. La fecha de esta evaluación final será definida por cada profesor y se informará a los alumnos.

**Ejemplos de los Diagramas**

Los siguientes son ejemplos de cada tipo de diagrama, creados con Excel. Son simplemente ejemplos para explicar el tipo de diagrama. Naturalmente el diagrama que se va a implementar en la práctica tendrá una apariencia muchos más sencilla.

|  |  |
| --- | --- |
|  |  |
| Diagrama de líneas | Diagrama de puntos |
|  |  |
| Diagrama de barras | Diagrama de torta |

## ***NOTA 1:***

La práctica se desarrollará de manera individual.

## ***NOTA 2:***

Cada entrega se realiza por Interactiva Virtual en la fecha determinada. En casos excepcionales se aceptará una entrega tardía, pero implica una rebaja en la nota. Se debe acordar previamente con el profesor.

## ***Importante:***

Cada estudiante debe dar cuenta completa de *todos* y *cada uno* de los productos que entrega. No se debe utilizar código creado por otras personas, ni código disponible en Internet, con excepción de las librerías de Java y el código que entrega el profesor de la asignatura. Durante la sustentación el profesor hablará con cada estudiante y solicitará que se realicen cambios al código. Si el estudiante no puede realizar estas modificaciones, se entenderá que *no hizo la aplicación* y se considerará como fraude de acuerdo con el reglamento estudiantil. Muy importante mirar los artículos pertinentes del Reglamento Académico en las siguientes páginas:

<http://www.eafit.edu.co/institucional/reglamentos/Documents/pregrado/regimen-disciplinario/cap1.pdf>

<http://www.eafit.edu.co/institucional/reglamentos/Documents/pregrado/regimen-disciplinario/cap4.pdf>

***Código de Ética***:

* Usted puede conversar con sus compañeros acerca de los enfoques que cada uno está utilizando para la práctica, pero NO se debe mirar el código de sus compañeros y mucho menos usarlo como parte de su práctica.
* Se puede utilizar, como punto de partida, el código que se entrega en el curso.
* No debe aceptar que otra persona (compañero, tío, amigo, novia, primo hermano del mocho) “le ayude” escribiendo parte del código de su práctica. Al hacerlo estaría perdiendo la mejor oportunidad para aprender que usted tiene en el curso.

*Esperamos que disfruten mucho haciendo su práctica. Las cosas que uno disfruta haciendo resultan mejor.*

**La recomendación más importante:**

*Es mejor entregar una práctica con una funcionalidad sencilla, pero hecha completamente por Usted, que una práctica muy completa hecha con ayuda indebida por parte de otra persona. Realizar la práctica, de manera individual, es la mejor experiencia de aprendizaje de la asignatura.*