



Informe del Taller # 02

Introducción a la Programación Orientada a Objetos

Christian Camilo Taborda Campiño 1632081-3743

Cristian Camilo Vallecilla Cuellar 1628790-3743

Juan Sebastián Paz Velásquez 1626846-3743

Punto 1: (Letrero)

Para el desarrollo de este programa se diseñó una clase -Letrero-, la cual entre sus métodos tenía tres herramientas que ayudaban a convertir la entrada en la salida deseada. Uno de los métodos pasaba las letras de minúsculas a mayúsculas, otra agregaba la cantidad de espacios necesarios, y la última dividía las líneas de la entrada con la ayuda de la función -substring- y la estructura de iteración -for-. Al realizar una instancia de -Letrero- se usaba el algoritmo con los tres métodos y después se imprimía la salida:

Implementación:

```
Ingrese la frase:
Sere un maestro pokemon

*****
*       S*
*      SE*
*     SER*
*    SERE*
*   SERE *
*  SERE U*
* SERE UN*
*SERE UN *
*ERE UN M*
*RE UN MA*
*E UN MAE*
* UN MAES*
*UN MAEST*
*N MAESTR*
* MAESTRO*
*MAESTRO *
*AESTRO P*
*ESTRO PO*
*STRO POK*
*TRO POKE*
*RO POKEM*
*O POKEMO*
* POKEMON*
*POKEMON *
*OKEMON  *
*KEMON   *
*EMON    *
*MON     *
*ON      *
*N       *
*        *
*****
```

Punto 2: (Triqui)

Para este programa se desarrollaron dos clases: -Jugador- y -Tablero-, a través de las cuales se implementó la relación de uso en uno de los métodos de -Jugador-, el cual modificaba uno de los elementos de la matriz que hacía parte de los atributos de -Tablero- por medio de una posición (X,Y). También se implementaron métodos en el -main- y la clase -Tablero- para validar si una entrada era correcta o no, y para decidir cuándo terminar la ejecución; todo lo anterior a través de la estructura condicional -if-.

Implementación:

Impresión del tablero inicial:

```
|-----TRIQUI-----|
*****
*   ESTADO DEL JUEGO   *
*****
|   |   |   |   |   |
*****
|   |   |   |   |   |
*****
|   |   |   |   |   |
*****
```

Impresión de una entrada del Jugador 1:

```
Jugador 1 (X)
Ingrese la fila:
1
Ingrese la columna:
1

|-----TRIQUI-----|

*****
*   ESTADO DEL JUEGO   *
*****
|  X  | |  |  |  |  |
*****
|     | |  |  |  |  |
*****
|     | |  |  |  |  |
*****
|     | |  |  |  |  |
*****
```

Impresión de una entrada del Jugador 2:

```
Jugador 2 (O)
Ingrese la fila:
1
Ingrese la columna:
2

|-----TRIQUI-----|

*****
*   ESTADO DEL JUEGO   *
*****
|  X  | |  O  | |  |  |
*****
|     | |  |  |  |  |
*****
|     | |  |  |  |  |
*****
|     | |  |  |  |  |
*****
```

Ingreso de una posición ocupada:

```
|-----TRIQUI-----|
*****
*   ESTADO DEL JUEGO   *
*****
| X | | 0 | |   |
*****
| X | | X | |   |
*****
| 0 | |   | |   |
*****

Jugador 2 (0)

Ingrese la fila:
2

Ingrese la columna:
2

Posición ocupada o no válida.
```

Ingreso de una posición incorrecta:

```
Ingrese la fila:
0

Ingrese la columna:
7

Posición ocupada o no válida.
```

Finalización del juego:

```
|-----TRIQUI-----|

*****
*   ESTADO DEL JUEGO   *
*****
| X | | O | |   |
*****
| X | | X | | O |
*****
| O | |   | | X |
*****

El juego ha terminado: Gana el jugador 1
```

Punto 3: (Conversación)

Para este punto se diseñaron dos clases, -Persona- y -Conversacion-. Se leía el archivo de texto ingresado línea a línea hasta encontrar la palabra “chao”, y cada línea se almacenaba en un vector. Una vez listo, el vector se pasaba como atributo para crear un objeto de la clase -Conversacion-. La clase -Persona- contiene los métodos –hablar- y –contestar-, los cuales retornan -strings-, la diferencia está en que –contestar- varía dependiendo de otro –string- de entrada, si esté hace parte del grupo de comandos, retornará el string junto con una parte extra en el mensaje dependiendo del comando que sea. El programa se basa en la estructura iterativa –while- para obtener líneas de la clase –Conversacion- y hacer que los objetos de –Persona- se contesten los mensajes uno a otro hasta encontrar la palabra “chao” en alguno de los mensajes emitidos.

Implementación:

Este es el archivo (.txt) de entrada:

```
Hola
mucho gusto.
¿Como vas?
y con mucha energía.
Eso está muy bien.
¿Tu edad?
pero ya me tengo que ir.
ahh ok.
Chao
```

Esta es la salida:

```
Camilo: Hola  
Juan: Hola, me llamo Juan, mucho gusto.  
Camilo: ¿Como vas?  
Juan: feliz, y con mucha energía.  
Camilo: Eso está muy bien.  
Juan: ¿Tu edad?  
Camilo: 30, pero ya me tengo que ir.  
Juan: ahh ok.  
Camilo: Chao  
Juan: Chao
```

Punto 4: (Uvnómina)

Para este punto se crearon varias clases. –Cargo- y –Salario-, las cuales con relación de composición complementaban a –Empleado-. Luego con relación de herencia se diseñaron las clases –Empleado_Ejecutivo- y –Empleado_Operativo-, por último se creó la clase –Consolidado-. El programa funciona a través de un vector dinámico para almacenar los objetos de empleados que se van ingresando. Luego sólo es usar los métodos -get- correspondientes a cada clase para mostrar todos los datos que el programa va almacenando. Para decidir qué actividad realizar el programa utiliza la estructura condicional –switch-.

Implementación:

Impresión del menú principal del programa:

```
////////////////////////////////////  
*****  
*   BIENVENIDO A UVNÓMINA   *  
*****  
*   Seleccione una opción:  *  
*****  
* 1. Ingresar empleado.    *  
* 2. Consultar empleado.   *  
* 3. Consultar Informes.   *  
* 4. Salir.                *  
*****
```

Impresión de opciones al elegir la opción 1 del menú principal:

```
Ingrese su opción:  
1  
  
Ingrese el tipo de empleado:  
1. Ejecutivo.  
2. Operativo.
```

Recolección de datos para el ingreso de un empleado ejecutivo:

```
Ingrese el nombre del empleado:
Christian

Ingrese la edad del empleado:
17

Ingrese la sección del cargo:
Oficina

Ingrese la descripción del cargo:
Programador

Ingrese el código del cargo:
AT178

Ingrese el monto básico del salario:
500

Ingrese las comisiones del salario:
150

Ingrese las primas extra-legales del salario:
730

Ingrese el número de empleados a cargo:
5

Ingrese el valor de los viáticos:
300

Ingreso exitoso!
```

Recolección de datos para el ingreso de un empleado operativo:

```
Ingrese el nombre del empleado:
Juan

Ingrese la edad del empleado:
26

Ingrese la sección del cargo:
Mantenimiento

Ingrese la descripción del cargo:
Reparador

Ingrese el código del cargo:
WE138G

Ingrese el monto básico del salario:
260

Ingrese las comisiones del salario:
120

Ingrese las primas extra-legales del salario:
200

Ingrese el nombre del jefe a cargo:
Christian

Ingreso exitoso!
```

Consulta de un empleado (opción 2 del menú principal):

```
Ingresa la identificación del empleado:
0

////////////////////////////////////

*****
*          BIENVENIDO A UVNÓMINA          *
*****
*          DATOS DEL EMPLEADO            *
*****
- Identificación: 0.
- Nombre: Christian.
- Empleado: Ejecutivo.
- Edad: 17.
- Num. Empleados a cargo: 5.
*****
*          CARGO                          *
*****
- Sección: Oficina.
- Descripción: Programador.
- Código: AT178.
*****
*          SALARIO                        *
*****
- Básico: 500.
- Comisión: 150.
- Prima Extra-Legal: 730.
- Viáticos: 300.
- Total: 1680.
*****
```

Impresión del mini-menú al seleccionar la opción 3 del menú principal:

```
Ingresa su opción:
3

*****
*          INFORMES                      *
*****
* 1. Informe por total.                  *
* 2. Informe por promedio.              *
*****
```

Consulta de informes por total:

```
////////////////////////////////////  
*****  
*   BIENVENIDO A UVNÓMINA   *  
*****  
*   INFORME POR TOTAL      *  
*****  
* Total Ejecutivos: 1.  
* Total Salarios: 1680.  
* Total Operativos: 1.  
* Total Salarios: 619.  
*****
```

Consulta de informes por promedio:

```
////////////////////////////////////  
*****  
*   BIENVENIDO A UVNÓMINA   *  
*****  
*   INFORME POR PROMEDIO    *  
*****  
* Total Ejecutivos: 1.  
* Promedio Salarios: 1680.  
* Total Operativos: 1.  
* Promedio Salarios: 619.  
*****
```

Salir del programa (opción 4 del menú principal):

```
Ingrese su opción:  
4  
christian@christian-VirtualBox:~/Escritorio/Taller 02 IP00/Punto 4$
```

