fidélitas Universidad	Ingeniería en Sistemas de Computación	Curso: Introducción a la Programación.
		Profesor: M.Sc. Ing. José Carlos Álvarez Umaña.
	Caso práctico 2	Código curso: SC-202

Porcentaje: 20%	Puntos	Porcentaje	Nota
Puntaje total: 100			

Instrucciones generales

El presente documento define las pautas para la elaboración de la Caso práctico 2 del curso de **INTRODUCCIÓN A LA PROGRAMACIÓN.**

Esta evaluación está incluida dentro de las evaluaciones de la **Directriz sobre Honestidad Académica**, presentada y aceptada en el **Programa del Curso**, el incumplimiento con la directriz mencionada generará la aplicación correspondiente del artículo 31 del reglamento estudiantil vigente.

A la hora de entregar el archivo se debe subir el proyecto completo a la sección correspondiente en el campus virtual, el nombre del archivo debe tener el siguiente formato.

#TipoEvaluacion_NombreApellido

1. **Objetivo:**

   Aplicar los conocimientos relacionados con orientación a objetos, arreglos y arreglos de objetos.

2. **Especificación**

   El caso práctico 2 es individual.

   No está permitido ningún *framework* o código previo. Todo código debe ser generado por el estudiante desde cero. Utilice NetBeans o IntelliJ Idea para generar la solución.

3. **Caso**

   La Universidad Felices S.A. ha solicitado sus servicios para construir un sistema de control de estudiantes. Se debe usar el paradigma orientado a objetos en el lenguaje Java para construir las clases. Dichas clases deben poseer los siguientes atributos y métodos. También en un archivo .txt se debe adjuntar el diagrama de clases, si quiere hacer uso de alguna herramienta grafica de modelado es libre de hacerlo. Todo el entregable debe ir en formato zip.

Se debe crear un programa en Java con su Main correspondiente, en el cual se evidencie el uso de todas las clases descritas a continuación.
Construya al menos una instancia de cada clase y use todos los métodos implementados.

1. ***Diagrama de clase Estudiante (9 puntos):***
    - Construya el diagrama de clase para Estudiante, siguiendo los requerimientos en el punto 2.
2. ***Clase Estudiante (10 puntos):***

- *Atributos:*
    - Id (int)
    - Nombre (String)
    - Apellido1 (String)
    - Apellido2 (String)
    - Carné (String)
    - Calificaciones (double[])
- *Métodos:*
    - Constructor: dicho método constructor debe recibir por parámetro Id, Nombre, Apellido1, Apellido2, Carné, Calificaciones.
    - Construya los getters y setters para los siguientes atributos:
        - Id
        - Nombre
        - Apellido1
        - Apellido2
        - Carné
        - Calificaciones.
    - ***GetPromedio:*** dicho método retorna el promedio de las calificaciones del estudiante.
    - ***GetNotaMayor:*** dicho método retorna la mayor nota por parte del estudiante.
    - ***GetNotaMenor***: dicho método retorna la menor nota por parte del estudiante.

3. ***Clase Main:***

- ***Método Main (9 puntos):***
    - Construya el método main.
    - Inicialice un arreglo de tipo Estudiante [] de tamaño N.
- ***Método ImprimirEstudiantes (9 puntos):***
    - Construya un método que imprima todos los contenidos de un arreglo estudiantes.

    - Dicho método debe ser static.
    - Public static void ImprimirEstudiantes(Estudiante[] estudiantes)
- ***Método InicializarEstudiantes (9 puntos):***
    - Construya un método que inicialice un arreglo de tamaño N, con N estudiantes.
    - No necesita solicitarle información al usuario, puede crear las instancias de la clase Estudiante directamente en su código.
    - Dicho método debe ser static.
    - Public static void InicializarEstudiantes(Estudiante[] estudiantes)
- ***Método BuscarEstudiantePorId (9 puntos):***
    - Construya un método que busque un estudiante por Id, en caso de encontrarlo imprima su contenido y termine la búsqueda. En caso de no encontrarlo solo muestre el mensaje "Id de estudiante no encontrado".
    - Dicho método debe ser static.
    - Public static void BuscarEstdiantePorId(Estudiante[] estudiantes, int id)
- ***Método BuscarEstudiantePorNombre (9 puntos):***
    - Construya un método que busque un estudiante por Nombre, en caso de encontrarlo imprima su contenido y termine la búsqueda. En caso de no encontrarlo solo muestre el mensaje "Nombre de estudiante no encontrado"
    - Dicho metodo debe ser static.
    - Public static void BuscarEstudiantePorNombre(Estudiante[] estudiantes, int nombre)
- ***Método EncontrarEstudianteMejorPromedio (9 puntos):***
    - Construya un método que busque el estudiante con mayor promedio e imprima su contenido.
    - Dicho método debe ser static.
    - Public static void EncontrarEstudianteMejorPromedio(Estudiante[] estudiantes)
- ***Método EncontrarEstudianteMayorNota (9 puntos):***
    - Construya un método que busque el estudiante con mayor nota e imprima su contenido.
    - Dicho método debe ser static.
    - Public static void EncontrarEstudianteMayorNota(Estudiante[] estudiantes)
- ***Método EncontrarEstudianteMenorNota (9 puntos):***
    - Construya un método que busque el estudiante con menor nota e imprima su contenido.

- Dicho método debe ser static.
- Public static void EncontrarEstudianteMenorNota(Estudiante[] estudiantes)

- ***<u>Método ModificarNombreEstudiantePorId (9 puntos):</u>***
  - Construya un método que busque el estudiante por Id y modifique su nombre. En caso de no encontrarlo simplemente muestre un mensaje "Id de estudiante no encontrado"
  - Dicho método debe ser static.
  - Public static void ModificarNombreEstudiantePorId(Estudiante[] estudiantes, int id, String nombre)