# Identificación

|  |  |
| --- | --- |
| Competencia a Desarrollar | Desarrollar el sistema de información que cumpla con los requerimientos de la solución informática. |
| **Actividad del Proyecto** | Identificar las características y procesos críticos garantizando estándares de calidad en el desarrollo de la solución informática. |
| **Resultados de aprendizaje** | Realizar la codificación de los módulos del sistema y el programa principal, a partir de la utilización del lenguaje de programación seleccionado, de acuerdo con las especificaciones del diseño. |
| **Ambientes de Aprendizaje** | Aula 3 Centro Minero. |
| **Orientador del Proceso** | Instructor Diego Alonso Ojeda Medina. |

# Taller

|  |  |
| --- | --- |
| Evidencia de Aprendizaje | Resultado de Aprendizaje |
| Identificación Básica de Sentencias. | Se realiza la programación de los ejercicios haciendo uso de todas las sentencias básicas del lenguaje Java y la teoría de la programación orientada a objetos: |
| Tipo de Entrega | Grupal según asignación de aprendices y ejercicios dados por el instructor |

## Vencimiento Actividad

|  |  |
| --- | --- |
| C:\Users\Carlos\Google Drive\SENA Presencial\Imagenes\fecha_hora.png | **Viernes, 05 de abril del 2019** |
| C:\Users\Carlos\Google Drive\SENA Presencial\Imagenes\fecha_hora.png | **13:00 PM** |

# Ejercicio No 1

Realizar una clase llamada **Persona** que siga las siguientes condiciones:

* Sus atributos son: **nombre, edad, documento, genero** (H para hombre, M para mujer), **peso y altura**. No queremos que se accedan directamente a ellos (encapsulación). Elegir cual modificador de acceso es el más adecuado, también su tipo. Si quieres añadir algún atributo puedes hacerlo.

Por defecto, todos los atributos menos el DNI serán valores por defecto según su tipo (0 números, cadena vacía para String, etc.). Sexo será hombre por defecto, usa una constante para ello.

* Se implantarán varios constructores:
  + Un constructor por defecto.
  + Un constructor con el nombre, edad y sexo, el resto por defecto.
  + Un constructor con todos los atributos como parámetro.
* Los métodos que se implementaran son:
  + **calcularIMC()**: calculara si la persona está en su peso ideal (peso en kg/(altura^2 en m)), devuelve un -1 si está por debajo de su peso ideal, un 0 si está en su peso ideal y un 1 si tiene sobrepeso .Te recomiendo que uses constantes para devolver estos valores.
  + **esMayorDeEdad()**: indica si es mayor de edad, devuelve un booleano.
  + **comprobarSexo(char sexo):** comprueba que el sexo introducido es correcto. Si no es correcto, será H. No será visible al exterior.
  + **toString():** devuelve toda la información del objeto.
  + **generaDNI():** genera un numero aleatorio de 8 cifras, genera a partir de este su número su letra correspondiente. Este método será invocado cuando se construya el objeto. Puedes dividir el método para que te sea más fácil. No será visible al exterior.
  + Métodos set de cada parámetro, excepto de DNI.

Ahora, crea una clase ejecutable que haga lo siguiente:

* Pide por teclado el nombre, la edad, sexo, peso y altura.
* Crea 3 objetos de la clase anterior, el primer objeto obtendrá las anteriores variables pedidas por teclado, el segundo objeto obtendrá todos los anteriores menos el peso y la altura y el último por defecto, para este último utiliza los métodos set para darle a los atributos un valor.
* Para cada objeto, deberá comprobar si está en su peso ideal, tiene sobrepeso o por debajo de su peso ideal con un mensaje.
* Indicar para cada objeto si es mayor de edad.
* Por último, mostrar la información de cada objeto.
* Puedes usar métodos en la clase ejecutable, para que os sea más fácil.

# Ejercicio No 2

Haz una clase llamada **Password** que siga las siguientes condiciones:

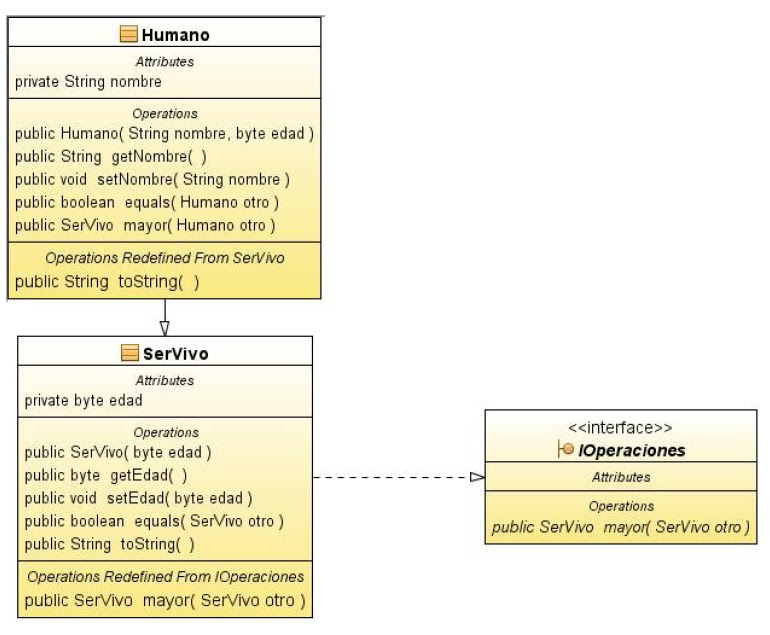
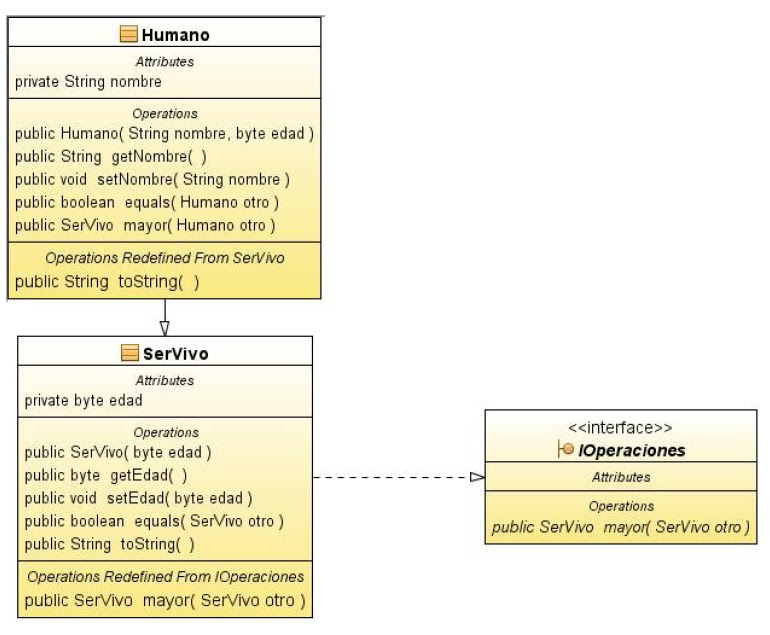
* Que tenga los atributos **longitud** y **contrasena**. Por defecto, la longitud será de 8.
* Los constructores serán los siguientes:
  + Un constructor por defecto.
  + Un constructor con la longitud que nosotros le pasemos. Generará una contraseña aleatoria con esa longitud.
* Los métodos que implementa serán:
  + **esFuerte()**: devuelve un booleano si es fuerte o no, para que sea fuerte debe tener más de 2 mayúsculas, más de 1 minúscula y más de 5 números, recuerde consultar la función toCharArray de la clase String.
  + **generarPassword()**: genera la contraseña del objeto con la longitud que tenga.
  + Método get para contraseña y longitud.
  + Método set sólo para longitud.

Ahora, crea una clase ejecutable:

* Crea un vector de Passwords con el tamaño que el usuario indique por teclado.
* Crea otro vector de booleanos, del mismo tamaño del vector anterior.
* A través de un ciclo FOR crea un objeto para cada posición del vector de passwords, indicando su longitud, y en el vector de booleanos deberás almacenar si el Password del vector de Passwords es o no fuerte.
* Al final, muestra la contraseña y si es o no fuerte. Usa este simple formato:
  + Contraseña 1: valor\_booleano\_si\_fuerte
  + Contraseña 1: valor\_booleano\_si\_fuerte

# Ejercicio No 3

Se tiene el siguiente diagrama de clases:



Debes tener en cuenta que el método **mayor (Humano otro)** debe retornar un objeto de la clase SerVivo.

* En el caso del método que está en la clase SerVivo vamos a retornar al SerVivo (de entre los dos que se comparan) de MAYOR EDAD.
* En el caso de la versión del método que se encuentra en la clase Humano DEBEREMOS DEVOLVER al humano (de entre los dos que se comparan) cuyo nombre sea mayor (de acuerdo con el orden alfabético) y que su edad sea mayor.

El método toString() DEBE retornar un String que contenga TODOS los datos del objeto.

Para implementar el método equals (Humano otro) DEBES SABER QUE:

* El propósito de este método es comparar dos objetos e indicar si son iguales.
* Dos seres vivos se consideran iguales si tienen la misma edad.
* Dos humanos se consideran iguales si tienen la misma edad y el mismo nombre.

Una vez realizado estas clases deberemos:

* Declarar DOS objetos de la clase SerVivo
  + Objeto 1 tiene 3 años.
  + Objeto 2 tiene 5 años.
* Usa el método mayor (Humano otro) para imprimir el mayor SerVivo de los creados anteriormente.
* Crea dos instancias de la clase Humano asignándolas a los objetos declarados anteriormente.
* Los objetos son:
  + Objeto X: nombre Homero, 34 años
  + Objeto Y: nombre Bart, 9 años
* Usando el resultado devuelto por el método mayor () imprimiremos el mayor Humano de los creados anteriormente.
* Usando el resultado devuelto por el método mayor () imprimiremos el menor Humano de los creados anteriormente.

# Ejercicio No 4

Crearemos una clase llamada Electrodoméstico con las siguientes características:

* Sus atributos son **precio base, color, consumo energético** (letras entre A y F) y **peso**. Indica que se podrán heredar.
* Por defecto, el color será blanco, el consumo energético será F, el precioBase es de $800.000 y el peso de 5 kg. Usa constantes para ello.
* Los colores disponibles son: blanco, negro, rojo, azul y gris. No importa si el nombre está en mayúsculas o en minúsculas.
* Los constructores que se implementarán serán
  + Un constructor por defecto.
  + Un constructor con el precio y peso. El resto por defecto.
  + Un constructor con todos los atributos.
* Los métodos que se implementará serán:
* Métodos get de todos los atributos.
  + **comprobarConsumoEnergetico(char letra):** comprueba que la letra es correcta, sino es correcta usará la letra por defecto. Se invocara al crear el objeto y no será visible al exterior de la clase.
  + **comprobarColor(String color):** comprueba que el color es correcto, sino lo es usa el color por defecto. Se invocará al crear el objeto y no será visible al exterior de la clase.
  + **precioFinal():** según el consumo energético, aumentará su precio, y según su tamaño, también. Esta es la lista de precios:

| **Consumo Energético** | **Precio** |
| --- | --- |
| A | $100.000 |
| B | $80.000 |
| C | $60.000 |
| D | $50.000 |
| E | $30.000 |
| F | $10.000 |

| **Tamaño** | **Precio** |
| --- | --- |
| Entre 0 y 19 kg | $10.000 |
| Entre 20 y 49 kg | $50.000 |
| Entre 50 y 79 kg | $80.000 |
| Mayor que 80 kg | $100.000 |

Crearemos una clase la cual heredará de la clase electrodoméstico-llamada **Lavadora**, con las siguientes características:

* Su atributo es **carga**, además de los atributos heredados.
* Por defecto, la carga es de 5 kg. Usa una constante para ello.
* Los constructores que se implementarán serán:
  + Un constructor por defecto.
  + Un constructor con el precio y peso. El resto por defecto.
  + Un constructor con la carga y el resto de los atributos heredados. Recuerda que debes llamar al constructor de la clase padre.
* Los métodos que se implementara serán:
  + Método get de carga.
  + **precioFinal**(): si tiene una carga mayor de 30 kg, aumentara el precio en $50.000, sino es así no se incrementará el precio. Llama al método padre y añade el código necesario. Recuerda que las condiciones que hemos visto en la clase Electrodoméstico también deben afectar al precio.

Crearemos una clase la cual heredará de electrodoméstico llamada **Televisión** con las siguientes características:

* Sus atributos son **resolución** (en pulgadas) y **sintonizador TDT** (booleano), además de los atributos heredados.
* Por defecto, la resolución será de 20 pulgadas y el sintonizador será false.
* Los constructores que se implementarán serán:
  + Un constructor por defecto.
  + Un constructor con el precio y peso. El resto por defecto.
  + Un constructor con la resolución, sintonizador TDT y el resto de los atributos heredados. Recuerda que debes llamar al constructor de la clase padre.
* Los métodos que se implementara serán:
  + Método get de resolución y sintonizador TDT.
  + **precioFinal**(): si tiene una resolución mayor de 40 pulgadas, se incrementará el precio un 30% y si tiene un sintonizador TDT incorporado, aumentara $50.000. Recuerda que las condiciones que hemos visto en la clase Electrodoméstico también deben afectar al precio.

Ahora crea una clase ejecutable que realice lo siguiente:

* Crea un array de Electrodomésticos de 10 posiciones.
* Asigna a cada posición un objeto de las clases anteriores con los valores que desees.
* Ahora, recorre este array y ejecuta el método precioFinal().
* Deberás mostrar el precio de cada clase, es decir, el precio de todos los televisores, por un lado, el de las lavadoras por otro y la suma de todos los Electrodomésticos.

Por ejemplo, si tenemos un Televisor con un precio final de $800.000, una lavadora de $1.000.000 y un televisor de $950.000, el resultado final será de $2.750.000 (800.000+1.000.000+950.000) para todos los electrodomésticos, $800.000 para lavadoras y $1.950.000 para televisores.

# Ejercicio No 5

Crearemos una clase llamada **Serie** con las siguientes características:

* Sus atributos son **título**, **numero de temporadas**, **entregado**, **género y creador**.
* Por defecto, el número de temporadas es de 3 temporadas y entregado **false**. El resto de los atributos serán valores por defecto según el tipo del atributo.
* Los constructores que se implementarán serán:
  + Un constructor por defecto.
  + Un constructor con el título y creador. El resto por defecto.
  + Un constructor con todos los atributos, excepto de entregado.
* Los métodos que se implementara serán:
  + Métodos get de todos los atributos, excepto de entregado.
  + Métodos set de todos los atributos, excepto de entregado.
  + Sobrescribe los métodos toString.

Crearemos una clase **Videojuego** con las siguientes características:

* Sus atributos son **título, horas estimadas, entregado, género y compañía.**
* Por defecto, las horas estimadas serán de 10 horas y entregado false. El resto de los atributos serán valores por defecto según el tipo del atributo.
* Los constructores que se implementarán serán:
  + Un constructor por defecto.
  + Un constructor con el título y horas estimadas. El resto por defecto.
  + Un constructor con todos los atributos, excepto de entregado.
* Los métodos que se implementara serán:
  + Métodos get de todos los atributos, excepto de entregado.
  + Métodos set de todos los atributos, excepto de entregado.
  + Sobrescribe los métodos toString.

Como vemos, en principio, las clases anteriores no son padre-hija, pero si tienen en común, por eso vamos a hacer una interfaz llamada **Entregable** con los siguientes métodos:

* **entregar():** cambia el atributo prestado a true.
* **devolver():** cambia el atributo prestado a false.
* **isEntregado():** devuelve el estado del atributo prestado.
* Método **compareTo (Object a),** compara las horas estimadas en los videojuegos y en las series el número de temporadas. Como parámetro que tenga un objeto, no es necesario que implementes la interfaz Comparable. Recuerda el uso de los castings de objetos.

Implementa los anteriores métodos en las clases Videojuego y Serie. Ahora crea una aplicación ejecutable y realiza lo siguiente:

* Crea dos arrays, uno de **Series** y otro de **Videojuegos**, de 5 posiciones cada uno.
* Crea un objeto en cada posición del array, con los valores que desees, puedes usar distintos constructores.
* Entrega algunos **Videojuegos** y **Series** con el método **entregar()**.
* Cuenta cuantos **Series** y **Videojuegos** hay entregados. Al contarlos, devuélvelos.
* Por último, indica el **Videojuego** tiene más horas estimadas y la serie con más temporadas. Muéstralos en pantalla con toda su información (usa el método toString()).

# Criterios de Evaluación.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Actividad a evaluar | Excelente | Buena | Regular | Rechazado |
| Estructura del programa  (20%) | El código del programa muestra legibilidad excelente en su estructura, declaración de variables correctas,  Constantes, operaciones y operan dos | El código del programa muestra una buena legibilidad aceptable en su estructura, por la ausencia o mala ubicación de algunos de sus componentes en la misma estructura | El código del programa muestra legibilidad aceptable en su estructura, por la ausencia o mala ubicación de varios de sus componentes en la misma estructura | El código del programa es completamente ilegibilidad en su estructura |
| Proceso de compilación  (20%) | El programa realiza el proceso de compilación sin errores | El programa realiza el proceso de compilación sin errores | El programa realiza el proceso de compilación sin errores | El programa no realiza el proceso de compilación, muestra errores de ejecución |
| Proceso de salida  (60%) | El programa muestra los resultados esperados, en función de los datos de entrada | El programa muestra los resultados, pero no son los esperados en función de los datos de entrada,  Los resultados que muestra son generados por una serie de cálculos mal realizados | El programa muestra los resultados, que no corresponden a las condiciones del programa | El programa no genera una salida |