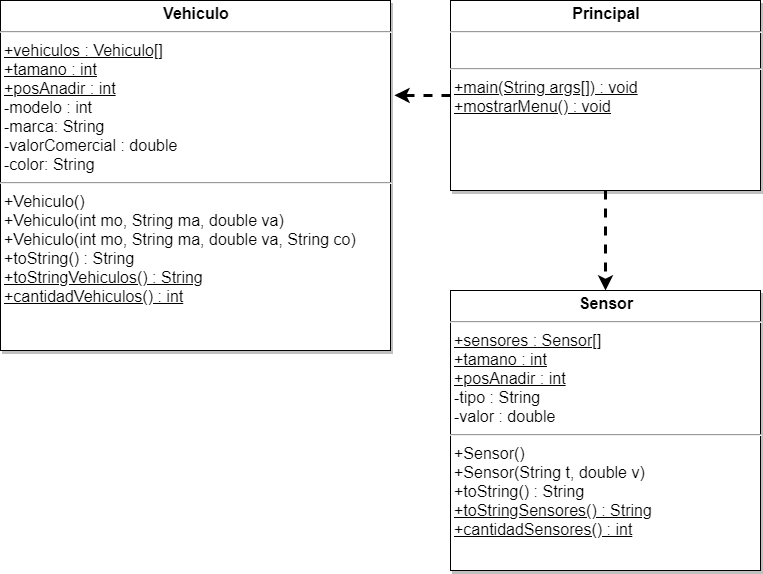
Proyecto # 1 – Fundamentos de programación – Docente: JUAN DAVID TAMAYO QUINTERO

|  |  |
| --- | --- |
| **Equipo de Trabajo** | **Nombre** |
|  |
|  |
|

Diagrama de clases base (incompleto)



Implementación en Java

Sobre la clase Vehículo parte 1

* Implemente los atributos de instancias tal cual se describe en el diagrama de clases.
* Implemente el atributo de clase “vehiculos” (**arreglo estático**), el cual almacenará hasta 10 vehiculos. Inicialice el arreglo con un tamaño de 10.
* Implemente el atributo de clase “tamano” e inicialícelo con un valor de 10. Este atributo servirá para indicar, hasta cuantos vehículos se pueden crear.
* Implemente el atributo de clase “posAnadir” e inicialícelo con un valor de 0. Este atributo servirá para indicar, en que posición del arreglo “vehiculos” se deberá añadir las nuevas instancias de vehiculos que se van creando.
* Implemente el constructor vacío que no haga nada.
* Implemente el constructor de 3 parámetros, que lo único que haga sea invocar al constructor de 4 parámetros y enviarle como color por defecto “verde”.
* Implemente el constructor de 4 parámetros que asigna los valores correspondientes a los atributos correspondientes.
* Implemente todos los métodos get y set correspondientes (no se colocaron en el diagrama de clases para que no quedara muy recargado).
* Implemente el método de instancia toString el cual devolverá un String con la información interna del vehículo (su modelo, marca, valorComercial y color).

Sobre la clase Vehículo parte 2

* Cada vez que cree una nueva instancia de vehículo, añádala al listado de vehículos de la clase Vehículo (utilice el atributo de clase posAnadir, para saber en qué posición añadirlo). Luego aumente en 1 el atributo postAnadir. Haga las modificaciones a los constructores o métodos anteriores, que considere pertinentes para que el sistema funcione tal cual lo requerido.
* Implemente el método de clase toStringVehiculos, el cual iterará sobre la lista de vehiculos de la clase Vehiculo y retornará un String concatenado con la información de todos los vehículos almacenados.
* Implemente el método de clase cantidadVehiculos, el cual retorna la cantidad actual de vehículos creados hasta el momento (pista: guíese por la información que brinda el atributo posAnadir).

Sobre la clase Sensor parte 1

* Implemente los atributos de instancias tal cual se describe en el diagrama de clases.
* Implemente el atributo de clase “sensores” (**arreglo estático**), el cual almacenará hasta 8 sensores. Inicialice el arreglo con un tamaño de 8.
* Implemente el atributo de clase “tamano” e inicialícelo con un valor de 8. Este atributo servirá para indicar, hasta cuantos sensores se pueden crear.
* Implemente el atributo de clase “posAnadir” e inicialícelo con un valor de 0. Este atributo servirá para indicar, en que posición del arreglo “sensores” se deberá añadir las nuevas instancias de sensores que se van creando.
* Implemente el constructor vacío que no haga nada.
* Implemente el constructor de 2 parámetros que asigna los valores correspondientes a los atributos correspondientes.
* Implemente todos los métodos get y set correspondientes (no se colocaron en el diagrama de clases para que no quedara muy recargado).
* Implemente el método de instancia toString el cual devolverá unString con la información interna del sensor (su tipo y valor).

Sobre la clase Sensor parte 2

* Cada vez que cree una nueva instancia de sensor, añádala al listado de sensores de la clase Sensor (utilice el atributo de clase posAnadir, para saber en qué posición añadirlo). Luego aumente en 1 el atributo postAnadir. Haga las modificaciones a los constructores o métodos anteriores, que considere pertinentes para que el sistema funcione tal cual lo requerido.
* Implemente el método de clase toStringSensores, el cual iterará sobre la lista de sensores de la clase Sensor y retornará un String concatenado con la información de todos los sensores almacenados.
* Implemente el método de clase cantidadSensores, el cual retorna la cantidad actual de sensores creados hasta el momento (pista: guíese por la información que brinda el atributo posAnadir).

Sobre la clase Principal

* Cree la clase Principal.
* Cree un método de clase main desde donde iniciará toda la ejecución del proyecto.
* Cree un método de clase mostrarMenu, el cual ejecutará un while infinito con las siguientes características:
  + Al inicio del while pídale al usuario que ingrese un número (muestre primero un mensaje donde quede claro si ingresa 0 que pasa, si ingresa 1 que pasa, etc).
  + Si se ingresa el número 0, se detiene la ejecución del while y finaliza la ejecución del programa.
  + Si se ingresa el número 1, se le piden al usuario los 4 parámetros de vehículo y se crea una instancia de vehículo. Ojo verifique antes que si puede añadir más vehículos a la lista de vehículos. En caso de que no pueda, muestre un mensaje “Error base de datos llena”.
  + Si se ingresa el número 2, se esperaría mostrar por pantalla la información de todos los vehículos almacenados hasta el momento.
  + Si se ingresa el número 3, se esperaría mostrar por pantalla la información de la cantidad de vehículos creados hasta el momento.
  + Si se ingresa el número 4, muestre por pantalla la información de todos los vehículos que tengan color “verde” (cree los métodos que considere pertinentes y/o haga las modificaciones pertinentes).
  + Si se ingresa el número 5, muestre por pantalla la información de todos los vehículos que tengan modelo entre 2000 y 2021 (cree los métodos que considere pertinentes y/o haga las modificaciones pertinentes).
  + Si se ingresa el número 6, se le piden al usuario los 2 parámetros de sensor y se crea una instancia de sensor. Ojo verifique antes que si puede añadir más sensores a la lista de sensores. En caso de que no pueda, muestre un mensaje “Error base de datos llena”.
  + Si se ingresa el número 7, se esperaría mostrar por pantalla la información de todos los sensores almacenados hasta el momento.
  + Si se ingresa el número 8, se esperaría mostrar por pantalla la información de la cantidad de sensores creados hasta el momento.
  + Si se ingresa el número 9, muestre por pantalla la información de todos los sensores que sean de tipo “temperatura” (cree los métodos que considere pertinentes y/o haga las modificaciones pertinentes).
  + Si se ingresa el número 666, muestre por pantalla los sensores tipo temperatura ordenados por valor. Para esto deberá crear un nuevo método, que retorne un nuevo arreglo con los sensores tipo temperatura ordenados por valor (el menor valor al principio). Y luego dentro de la ejecución del caso 666, se invoque ese método, se recoja el nuevo arreglo recibido, y se muestre la información de ese arreglo (cree los métodos que considere pertinentes y/o haga las modificaciones pertinentes // verifique bien a que clases le asigna esas responsabilidades).
* Invoque el método mostrarMenu desde el método main, y verifique que su proyecto funciona (cree sensores, cree vehículos, liste sensores, liste vehículos, etc). Verifique que todo funciona.

Reglas importantes sobre el programa

* Jamás haga un print, dentro de las clases Vehiculo o Sensor.
* Jamás haga uso del Scanner, dentro de las clases Vehiculo o Sensor.

Instrucciones de entrega:

* Desarrolle este proyecto en equipos de 2
* Cuando termine el proyecto (o a medida que lo vaya desarrollando), súbalo a GitHub. Para esta entrega lo único que usted le deberá enviar al docente es el enlace del proyecto GitHub (en Teams podrá encontrar un tutorial de conexión de BlueJ y GitHub).
* Enviar video explicando el programa realizado, subir solo el enlace encontrado en Teams.
* Defina con sus compañeros de equipo un estilo de codificación, Todos deberán usar un mismo estilo para definir atributos (por ejemplo, todos los atributos se definen con camelCase), para definir clases, para definir ciclos, variables, etc. Por ejemplo, podrían utilizar esta guía como base: <https://google.github.io/styleguide/javaguide.html>
* Se DEBE ENVIAR ANTES DEL **DOMINGO 30 de octubre**. (PLAZO DOS SEMANAS) del 17 de octubre al 30 de octubre hasta las 11:59