EJERCICIOS DE APRENDIZAJE

Antes de comenzar con esta guía, les damos algunas recomendaciones:

Este módulo es uno de los más divertidos ya que vamos a comenzar a modelar los objetos del mundo real con el lenguaje de programación Java. Es importante tener en cuenta que entender la programación orientada a objetos lleva tiempo y sobre todo PRÁCTICA, así que, a no desesperarse, con cada ejercicio vamos a ir entendiendo un poco más cómo aplicar este paradigma.

VIDEOS: Te sugerimos ver los videos relacionados con este tema, antes de empezar los ejercicios, los podrás encontrar en tu aula virtual o en nuestro canal de YouTube.

1. Crear una clase llamada Libro que contenga los siguientes atributos: ISBN, Título, Autor, Número de páginas, y un constructor con todos los atributos pasados por parámetro y un constructor vacío. Crear un método para cargar un libro pidiendo los datos al usuario y luego informar mediante otro método el número de ISBN, el título, el autor del libro y el numero de páginas.

2. Declarar una clase llamada Circunferencia que tenga como atributo privado el radio de tipo real. A continuación, se deben crear los siguientes métodos:

   a) Método constructor que inicialice el radio pasado como parámetro.

   b) Métodos get y set para el atributo radio de la clase Circunferencia.

   c) Método para crearCircunferencia(): que le pide el radio y lo guarda en el atributo del objeto.

   d) Método area(): para calcular el área de la circunferencia $\mathbf{(Area = \pi * radio^2)}$.

   e) Método perimetro(): para calcular el perímetro $\mathbf{(Perimetro = 2 * \pi * radio)}$.

3. Crear una clase llamada Operacion que tenga como atributos privados numero1 y numero2. A continuación, se deben crear los siguientes métodos:

   a) Método constructor con todos los atributos pasados por parámetro.

   b) Metodo constructor sin los atributos pasados por parámetro.

   c) Métodos get y set.

   d) Método para crearOperacion(): que le pide al usuario los dos números y los guarda en los atributos del objeto.

   e) Método sumar(): calcular la suma de los números y devolver el resultado al main.

   f) Método restar(): calcular la resta de los números y devolver el resultado al main

g) Método multiplicar(): primero valida que no se haga una multiplicación por cero, si fuera a multiplicar por cero, el método devuelve 0 y se le informa al usuario el error. Si no, se hace la multiplicación y se devuelve el resultado al main

h) Método dividir(): primero valida que no se haga una división por cero, si fuera a pasar una división por cero, el método devuelve 0 y se le informa al usuario el error se le informa al usuario. Si no, se hace la división y se devuelve el resultado al main.

4. Crear una clase Rectángulo que modele rectángulos por medio de un atributo privado **base** y un atributo privado **altura**. La clase incluirá un método para crear el rectángulo con los datos del Rectángulo dados por el usuario. También incluirá un método para calcular la superficie del rectángulo y un método para calcular el perímetro del rectángulo. Por último, tendremos un método que dibujará el rectángulo mediante asteriscos usando la base y la altura. Se deberán además definir los métodos getters, setters y constructores correspondientes.

Superficie = base * altura / **Perímetro** = (base + altura) * 2.

5. Realizar una clase llamada Cuenta (bancaria) que debe tener como mínimo los atributos: **numeroCuenta** (entero), el **DNI del cliente** (entero largo) y el **saldo actual** (entero). Las operaciones asociadas a dicha clase son:

a) Constructor por defecto y constructor con DNI, saldo, número de cuenta e interés.

b) Agregar los métodos getters y setters correspondientes

c) Metodo para crear un objeto Cuenta, pidiéndole los datos al usuario.

d) Método ingresar(double ingreso): el método recibe una cantidad de dinero a ingresar y se la sumara a saldo actual.

e) Método retirar(double retiro): el método recibe una cantidad de dinero a retirar y se la restará al saldo actual. Si la cuenta no tiene la cantidad de dinero a retirar, se pondrá el saldo actual en 0.

f) Método extraccionRapida(): le permitirá sacar solo un 20% de su saldo. Validar que el usuario no saque más del 20%.

g) Método consultarSaldo(): permitirá consultar el saldo disponible en la cuenta.

h) Método consultarDatos(): permitirá mostrar todos los datos de la cuenta

6. Programa Nespresso. Desarrolle una clase Cafetera con los atributos **capacidadMaxima** (la cantidad máxima de café que puede contener la cafetera) y **cantidadActual** (la cantidad actual de café que hay en la cafetera). Implemente, al menos, los siguientes métodos:

- Constructor predeterminado o vacío
- Constructor con la capacidad máxima y la cantidad actual

- Métodos getters y setters.
- Método llenarCafetera(): hace que la cantidad actual sea igual a la capacidad máxima.
- Método servirTaza(int): se pide el tamaño de una taza vacía, el método recibe el tamaño de la taza y simula la acción de servir la taza con la capacidad indicada. Si la cantidad actual de café "no alcanza" para llenar la taza, se sirve lo que quede. El método le informará al usuario si se llenó o no la taza, y de no haberse llenado en cuanto quedó la taza.
- Método vaciarCafetera(): pone la cantidad de café actual en cero.
- Método agregarCafe(int): se le pide al usuario una cantidad de café, el método lo recibe y se añade a la cafetera la cantidad de café indicada.

7. Realizar una clase llamada Persona que tenga los siguientes atributos: nombre, edad, sexo ('H' hombre, 'M' mujer, 'O' otro), peso y altura. Si el alumno desea añadir algún otro atributo, puede hacerlo. Los métodos que se implementarán son:

- Un constructor por defecto.
- Un constructor con todos los atributos como parámetro.
- Métodos getters y setters de cada atributo.
- Metodo crearPersona(): el método crear persona, le pide los valores de los atributos al usuario y después se le asignan a sus respectivos atributos para llenar el objeto Persona. Además, comprueba que el sexo introducido sea correcto, es decir, H, M o O. Si no es correcto se deberá mostrar un mensaje
- Método calcularIMC(): calculara si la persona está en su peso ideal (peso en kg/(altura^2 en mt2)). Si esta fórmula da por resultado un valor menor que 20, significa que la persona está por debajo de su peso ideal y la función devuelve un -1. Si la fórmula da por resultado un número entre 20 y 25 (incluidos), significa que la persona está en su peso ideal y la función devuelve un 0. Finalmente, si el resultado de la fórmula es un valor mayor que 25 significa que la persona tiene sobrepeso, y la función devuelve un 1.
- Método esMayorDeEdad(): indica si la persona es mayor de edad. La función devuelve un booleano.

A continuación, en la clase main hacer lo siguiente:

Crear 4 objetos de tipo Persona con distintos valores, a continuación, llamaremos todos los métodos para cada objeto, deberá comprobar si la persona está en su peso ideal, tiene sobrepeso o está por debajo de su peso ideal e indicar para cada objeto si la persona es mayor de edad.

Por último, guardaremos los resultados de los métodos calcularIMC y esMayorDeEdad en distintas variables, para después en el main, calcular un porcentaje de esas 4 personas

cuantas están por debajo de su peso, cuantas en su peso ideal y cuantos, por encima, y también calcularemos un porcentaje de cuantos son mayores de edad y cuantos menores

CLASES DE UTILIDAD EN JAVA

Los métodos disponibles para las clases de utilidad Integer, Arrays y Date están en esta guía. Recordar que la clase String y Math están explicadas en la guía anterior de Intro Java.

VIDEOS: Te sugerimos ver los videos relacionados con este tema, antes de empezar los ejercicios, los podrás encontrar en tu aula virtual o en nuestro canal de YouTube.

8. Realizar una clase llamada Cadena que tenga como atributos una frase (de tipo de String) y su longitud. En el main se creará el objeto y se le pedirá al usuario que ingrese una frase que puede ser una palabra o varias palabras separadas por un espacio en blanco y a través de los métodos set, se guardará la frase y la longitud de manera automática según la longitud de la frase ingresada. Deberá además implementar los siguientes métodos:

   a) Método mostrarVocales(), deberá contabilizar la cantidad de vocales que tiene la frase ingresada.

   b) Método invertirFrase(), deberá invertir la frase ingresada y mostrarla por pantalla. Por ejemplo: Entrada: "casa blanca", Salida: "acnalb asac".

   c) Método vecesRepetido(String letra), recibirá un carácter ingresado por el usuario y contabilizar cuántas veces se repite el carácter en la frase, por ejemplo:

   d) Entrada: frase = "casa blanca". Salida: El carácter 'a' se repite 4 veces.

   e) Método compararLongitud(String frase), deberá comparar la longitud de la frase que compone la clase con otra nueva frase ingresada por el usuario.

   f) Método unirFrases(String frase), deberá unir la frase contenida en la clase Cadena con una nueva frase ingresada por el usuario y mostrar la frase resultante.

   g) Método reemplazar(String letra), deberá reemplazar todas las letras "a" que se encuentren en la frase, por algún otro carácter seleccionado por el usuario y mostrar la frase resultante.

   h) Método contiene(String letra), deberá comprobar si la frase contiene una letra que ingresa el usuario y devuelve verdadero si la contiene y falso si no.

Método Static y Clase Math

9. Realizar una clase llamada Matemática que tenga como atributos dos números reales con los cuales se realizarán diferentes operaciones matemáticas. La clase deber tener un constructor vacío, parametrizado y get y set. En el main se creará el objeto y se usará el Math.random para generar los dos números y se guardaran en el objeto con su respectivos set. Deberá además implementar los siguientes métodos:

a) Método devolverMayor() para retornar cuál de los dos atributos tiene el mayor valor

b) Método calcularPotencia() para calcular la potencia del mayor valor de la clase elevado al menor número. Previamente se deben redondear ambos valores.

c) Método calculaRaiz(), para calcular la raíz cuadrada del menor de los dos valores. Antes de calcular la raíz cuadrada se debe obtener el valor absoluto del número.

Clase Arrays

10. Realizar un programa en Java donde se creen dos arreglos: el primero será un arreglo A de 50 números reales, y el segundo B, un arreglo de 20 números, también reales. El programa deberá inicializar el arreglo A con números aleatorios y mostrarlo por pantalla. Luego, el arreglo A se debe ordenar de menor a mayor y copiar los primeros 10 números ordenados al arreglo B de 20 elementos, y rellenar los 10 últimos elementos con el valor 0.5. Mostrar los dos arreglos resultantes: el ordenado de 50 elementos y el combinado de 20.

Clase Date

11. Pongamos de lado un momento el concepto de POO, ahora vamos a trabajar solo con la clase Date. En este ejercicio deberemos instanciar en el main, una fecha usando la clase Date, para esto vamos a tener que crear 3 variables, dia, mes y anio que se le pedirán al usuario para poner el constructor del objeto Date. Una vez creada la fecha de tipo Date, deberemos mostrarla y mostrar cuantos años hay entre esa fecha y la fecha actual, que se puede conseguir instanciando un objeto Date con constructor vacío.

Ejemplo fecha: Date fecha = new Date(anio, mes, dia);

Ejemplo fecha actual: Date fechaActual = new Date();

12. Implemente la clase Persona. Una persona tiene un nombre y una fecha de nacimiento (Tipo Date), constructor vacío, constructor parametrizado, get y set. Y los siguientes métodos:

- Agregar un método de creación del objeto que respete la siguiente firma: crearPersona(). Este método rellena el objeto mediante un Scanner y le pregunta al usuario el nombre y la fecha de nacimiento de la persona a crear, recordemos que la fecha de nacimiento debe guardarse en un Date y los guarda en el objeto.
- Escribir un método calcularEdad() a partir de la fecha de nacimiento ingresada. Tener en cuenta que para conocer la edad de la persona también se debe conocer la fecha actual.
- Agregar a la clase el método menorQue(int edad) que recibe como parámetro otra edad y retorna true en caso de que el receptor tenga menor edad que la persona que se recibe como parámetro, o false en caso contrario.
- Metodo mostrarPersona(): este método muestra la persona creada en el método anterior.

Uso de vectores como atributos de clase

13. Un profesor particular está dando cursos para grupos de cinco alumnos y necesita un programa para organizar la información de cada curso. Para ello, crearemos una clase entidad llamada Curso, cuyos atributos serán: nombreCurso, cantidadHorasPorDia, cantidadDiasPorSemana, turno (mañana o tarde), precioPorHora y alumnos. Donde alumnos es un arreglo de tipo String de dimensión 5 (cinco), donde se alojarán los nombres de cada alumno. A continuación, se implementarán los siguientes métodos:

    - Un constructor por defecto.
    - Un constructor con todos los atributos como parámetro.
    - Métodos getters y setters de cada atributo.
    - método cargarAlumnos(): este método le permitirá al usuario ingresar los alumnos que asisten a las clases. Nosotros nos encargaremos de almacenar esta información en un arreglo e iterar con un bucle, solicitando en cada repetición que se ingrese el nombre de cada alumno.
    - Método crearCurso(): el método crear curso, le pide los valores de los atributos al usuario y después se le asignan a sus respectivos atributos para llenar el objeto Curso. En este método invocaremos al método cargarAlumnos() para asignarle valor al atributo alumnos
    - Método calcularGananciaSemanal(): este método se encarga de calcular la ganancia en una semana por curso. Para ello, se deben multiplicar la cantidad de horas por día, el precio por hora, la cantidad de alumnos y la cantidad de días a la semana que se repite el encuentro.

14. Una tienda que vende teléfonos móviles quiere tener registro de cada producto que posee en un sistema computacional. Para ello, crearemos un programa donde se pueda almacenar cada producto con su información. Crear una entidad Movil con los atributos marca, precio, modelo, memoriaRam, almacenamiento y codigo. El atributo código será un arreglo numérico de dimensión 7 (siete) donde cada subíndice alojará un número correspondiente al código. A continuación, se implementarán los siguientes métodos:

    - Un constructor por defecto.
    - Un constructor con todos los atributos como parámetro.
    - Métodos getters y setters de cada atributo.
    - Método cargarCelular(): se solicita al usuario que ingrese los datos necesarios para instanciar un objeto Celular y poder cargarlo en nuestro programa.
    - Método ingresarCodigo(): este método permitirá ingresar el código completo de siete números de un celular. Para ello, puede utilizarse un bucle repetitivo

EJERCICIOS DE APRENDIZAJE EXTRA

Estos van a ser ejercicios para reforzar los conocimientos previamente vistos. Estos pueden realizarse cuando hayas terminado la guía y tengas una buena base sobre lo que venimos trabajando. Además, si ya terminaste la guía y te queda tiempo libre en las mesas, podes continuar con estos ejercicios extra, recordando siempre que no es necesario que los termines para continuar con el tema siguiente. Por último, recordá que la prioridad es ayudar a los compañeros de la mesa y que cuando tengas que ayudar, lo más valioso es que puedas explicar el ejercicio con la intención de que tu compañero lo comprenda, y no sólo mostrarlo. ¡Muchas gracias!

1. Desarrollar una clase Cancion con los siguientes atributos: titulo y autor. Se deberá definir además dos constructores: uno vacío que inicializa el titulo y el autor a cadenas vacías y otro que reciba como parámetros el titulo y el autor de la canción. Se deberán además definir los métodos getters y setters correspondientes.

2. Definir una clase llamada Puntos que contendrá las coordenadas de dos puntos, sus atributos serán, x1, y1, x2, y2, siendo cada x e y un punto. Generar un objeto puntos usando un método crearPuntos() que le pide al usuario los dos números y los ingresa en los atributos del objeto. Después, a través de otro método calcular y devolver la distancia que existe entre los dos puntos que existen en la clase Puntos. Para conocer como calcular la distancia entre dos puntos consulte el siguiente link:

   http://www.matematicatuya.com/GRAFICAecuaciones/S1a.html

3. Vamos a realizar una clase llamada Raices, donde representaremos los valores de una ecuación de 2o grado. Tendremos los 3 coeficientes como atributos, llamémosles a, b y c. Hay que insertar estos 3 valores para construir el objeto a través de un método constructor. Luego, las operaciones que se podrán realizar son las siguientes:

   - Método getDiscriminante(): devuelve el valor del discriminante (double). El discriminante tiene la siguiente formula: (b^2)-4*a*c
   - Método tieneRaices(): devuelve un booleano indicando si tiene dos soluciones, para que esto ocurra, el discriminante debe ser mayor o igual que 0.
   - Método tieneRaiz(): devuelve un booleano indicando si tiene una única solución, para que esto ocurra, el discriminante debe ser igual que 0.
   - Método obtenerRaices(): llama a tieneRaíces() y si devolvió true, imprime las 2 posibles soluciones.
   - Método obtenerRaiz(): llama a tieneRaiz() y si devolvió true imprime una única raíz. Es en el caso en que se tenga una única solución posible.
   - Método calcular(): esté método llamará tieneRaices() y a tieneRaíz(), y mostrará por pantalla las posibles soluciones que tiene nuestra ecuación con los métodos obtenerRaices() o obtenerRaiz(), según lo que devuelvan nuestros métodos y en caso de no existir solución, se mostrará un mensaje.

Nota: Formula ecuación 2o grado: (-b±√((b^2)-(4*a*c)))/(2*a) Solo varia el signo delante de -b

4. Dígito Verificador. Crear una clase NIF que se usará para mantener DNIs con su correspondiente letra (NIF). Los atributos serán el número de DNI (entero largo) y la letra (String o char) que le corresponde. Dispondrá de los siguientes métodos:

   - Métodos getters y setters para el número de DNI y la letra.
   - Método crearNif(): le pide al usuario el DNI y con ese DNI calcula la letra que le corresponderá. Una vez calculado, le asigna la letra que le corresponde según el resultado del calculo.
   - Método mostrar(): que nos permita mostrar el NIF (ocho dígitos, un guion y la letra en mayúscula; por ejemplo: 00395469-F).

   La letra correspondiente al dígito verificador se calculará a traves de un método que funciona de la siguiente manera: Para calcular la letra se toma el resto de dividir el número de DNI por 23 (el resultado debe ser un número entre 0 y 22). El método debe buscar en un array (vector) de caracteres la posición que corresponda al resto de la división para obtener la letra correspondiente. La tabla de caracteres es la siguiente:

POSICIÓN	LETRA
0	T
1	R
2	W
3	A
4	G
5	M
6	Y
7	F
8	P
9	D
10	X
11	B
12	N
13	J
14	Z
15	S
16	Q
17	V

18	H
19	L
20	C
21	K
22	E

5. Crea una clase en Java donde declares una variable de tipo array de Strings que contenga los doce meses del año, en minúsculas. A continuación, declara una variable mesSecreto de tipo String, y hazla igual a un elemento del array (por ejemplo, mesSecreto = mes[9]. El programa debe pedir al usuario que adivine el mes secreto. Si el usuario acierta mostrar un mensaje, y si no lo hace, pedir que vuelva a intentar adivinar el mes secreto. Un ejemplo de ejecución del programa podría ser este:

   Adivine el mes secreto. Introduzca el nombre del mes en minúsculas: febrero

   No ha acertado. Intente adivinarlo introduciendo otro mes: agosto

   ¡Ha acertado!

6. Juego Ahorcado: Crear una clase Ahorcado (como el juego), la cual deberá contener como atributos, un vector con la palabra a buscar, la cantidad de letras encontradas y la cantidad jugadas máximas que puede realizar el usuario. Definir los siguientes métodos con los parámetros que sean necesarios:

   - Constructores, tanto el vacío como el parametrizado.
   - Metodo crearJuego(): le pide la palabra al usuario y cantidad de jugadas máxima. Con la palabra del usuario, pone la longitud de la palabra, como la longitud del vector. Después ingresa la palabra en el vector, letra por letra, quedando cada letra de la palabra en un índice del vector. Y también, guarda en cantidad de jugadas máximas, el valor que ingresó el usuario y encontradas en 0.
   - Método longitud(): muestra la longitud de la palabra que se debe encontrar. Nota: buscar como se usa el vector.length.
   - Método buscar(letra): este método recibe una letra dada por el usuario y busca sila letra ingresada es parte de la palabra o no. También informará si la letra estaba o no.
   - Método encontradas(letra): que reciba una letra ingresada por el usuario y muestre cuantas letras han sido encontradas y cuantas le faltan. Este método además deberá devolver true si la letra estaba y false si la letra no estaba, ya que, cada vez que se busque una letra que no esté, se le restará uno a sus oportunidades.
   - Método intentos(): para mostrar cuantas oportunidades le queda al jugador.
   - Método juego(): el método juego se encargará de llamar todos los métodos previamente mencionados e informará cuando el usuario descubra toda la palabra o se quede sin intentos. Este método se llamará en el main.

   Un ejemplo de salida puede ser así:

Ingrese una letra:

a

Longitud de la palabra: 6

Mensaje: La letra pertenece a la palabra

Número de letras (encontradas, faltantes): (3,4)

Número de oportunidades restantes: 4

Ingrese una letra:

z

Longitud de la palabra: 6

Mensaje: La letra no pertenece a la palabra

Número de letras (encontradas, faltantes): (3,4)

Número de oportunidades restantes: 3

Ingrese una letra:

b

Longitud de la palabra: 6

Mensaje: La letra no pertenece a la palabra

Número de letras (encontradas, faltantes): (4,3)

Número de oportunidades restantes: 2

Ingrese una letra:

u

Longitud de la palabra: 6

Mensaje: La letra no pertenece a la palabra

Número de letras (encontradas, faltantes): (4,3)

Número de oportunidades restantes: 1

Ingrese una letra:

q

Longitud de la palabra: 6

Mensaje: La letra no pertenece a la palabra

Mensaje: Lo sentimos, no hay más oportunidades

Pueden encontrar un ejemplo para descargar de vector en el GIT de tu Curso.

BIBLIOGRAFÍA

Información sacada de libros:

- Fundamentos de Programación de Luis Joyanes Aguilar

- Fundamentos de Programación Java de Jorge Martínez Ladrón de Guevara en conjunto con la Facultad de Informática (Universidad Complutense de Madrid).

- Introducción a la programación Java de John S. Dean y Raymond H. Dean

Información sacada de las páginas:

- https://profile.es/blog/que-son-los-paradigmas-de-programacion/

- https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/