CURSO DE PROGRAMACIÓN FULL STACK

PROGRAMACIÓN ORIENTADA A OBJETOS

PARADIGMA ORIENTADO A OBJETOS

GUÍA DE PARADIGMA ORIENTADO A OBJETOS

CLASES Y OBJETOS

Una clase es un molde para crear múltiples objetos que encapsula datos y comportamiento. Una clase es una combinación específica de atributos y métodos y puede considerarse un tipo de dato de cualquier tipo no primitivo. Así, una clase es una especie de plantilla o prototipo de objetos: define los atributos que componen ese tipo de objetos y los métodos que pueden emplearse para trabajar con esos objetos. En su forma más simple, una clase se define por la palabra reservada class seguida del nombre de la clase. El nombre de la clase debe empezar por mayúscula. Si el nombre es compuesto, entonces cada palabra debe empezar por mayúscula. La definición de la clase se pone entre las llaves de apertura y cierre.

```
public class NombreClase {
      // miembros dato o atributos
      // constructores
      // funciones miembro
}
```

Una vez que se ha declarada una clase, se pueden crear objetos a partir de ella. A la creación de un objeto se le denomina instanciación. Es por esto que se dice que un objeto es una instancia de una clase y el término instancia y objeto se utilizan indistintamente. Para crear objetos, basta con declarar una variable de alguno de los tipos de las clases definidas.

```
NombreClase nombreObjeto;
```

Para crear el objeto y asignar un espacio de memoria es necesario realizar la instanciación con el operador new. El operador new reserva espacio en memoria para los miembros dato y devuelve una referencia que se guarda en la variable.

```
nombreObjeto = new nombreClase();
```

Tanto la declaración de un objeto como la asignación del espacio de memoria se pueden realizar en un solo paso:

```
NombreClase nombreObjeto = new NombreClase();
```

A partir de este momento los objetos ya pueden ser referenciados por su nombre.

ACCESO A LOS MIEMBROS

Desde un objeto se puede acceder a los miembros mediante la siguiente sintaxis:

```
nombreObjeto.miembro
```

ESTADO Y COMPORTAMIENTO

En términos más generales, un objeto es una abstracción conceptual del mundo real que se puede traducir a un lenguaje de programación orientado a objetos. Los objetos del mundo real comparten dos características: Todos poseen estado y comportamiento. Por ejemplo, el perro tiene estado (color, nombre, raza, edad) y el comportamiento (ladrar, caminar, comer, acostarse, mover la cola). Por lo tanto, un estado permite informar cuáles son las características del objeto y lo que éste representa, y el comportamiento, consiste en decir lo que sabe hacer.

El estado de un objeto es una lista de variables conocidas como sus atributos, cuyos valores representan el estado que caracteriza al objeto.

El comportamiento es una lista de métodos, procedimientos, funciones u operaciones que un objeto puede ejecutar a solicitud de otros objetos. Los objetos también se conocen como instancias.

ELEMENTOS DE UNA CLASE

Una clase describe un tipo de objetos con características comunes. Es necesario definir la información que almacena el objeto y su comportamiento.

ATRIBUTOS

La información de un objeto se almacena en atributos. Los atributos pueden ser de tipos primitivos de Java (descriptos en el Apéndice A) o del tipo de otros objetos. La declaración de un atributo de un objeto tiene la siguiente forma:

```
<modificador>* <tipo> <nombre> [ = <valor inicial> ];
```

- <nombre>: puede ser cualquier identificador válido y denomina el atributo que está siendo declarado.
- <modificador>: si bien hay varios valores posibles para el <modificador>, por el momento solo usaremos modificadores de visibilidad: public, protected, private.
- <tipo>: indica la clase a la que pertenece el atributo definido.
- <valor inicial>: esta sentencia es opcional y se usa para inicializar el atributo del objeto con un valor particular.

CONSTRUCTORES

Además de definir los atributos de un objeto, es necesario definir los métodos que determinan su comportamiento. Toda clase debe definir un método especial denominado constructor para instanciar los objetos de la clase. Este método tiene el mismo nombre de la clase. La declaración básica toma la siguiente forma:

```
[<modificador>] <nombre de clase> ( <argumento>* ) {
   <sentencia>*
}
```

- <nombre de clase>: El nombre del constructor debe ser siempre el mismo que el de la clase.
- <modificador>: Actualmente, los únicos modificadores válidos para los constructores son public, protected y private.
- <argumentos>: es una lista de parámetros que tiene la misma función que en los métodos.

El método constructor se ejecuta cada vez que se instancia un objeto de la clase. Este método se utiliza para inicializar los atributos del objeto que se instancia.

Para diferenciar entre los atributos del objeto y los identificadores de los parámetros del método constructor, se utiliza la palabra this. De esta forma, los parámetros del método pueden tener el mismo nombre que los atributos de la clase.

La instanciación de un objeto consiste en asignar un espacio de memoria al que se hace referencia con el nombre del objeto. Los identificadores de los objetos permiten acceder a los valores almacenados en cada objeto.

El Constructor por Defecto

Cada clase tiene al menos un constructor. Si no se escribe un constructor, el lenguaje de programación Java le provee uno por defecto. Este constructor no posee argumentos y tiene un cuerpo vacío. Si se define un constructor que no sea vacío, el constructor vacío se pierde, salvo que explícitamente lo definamos.

MÉTODOS

Los métodos son funciones que determinan el comportamiento de los objetos. Un objeto se comporta de una u otra forma dependiendo de los métodos de la clase a la que pertenece. Todos los objetos de una misma clase tienen los mismos métodos y el mismo comportamiento. Para definir los métodos, el lenguaje de programación Java toma la siguiente forma básica:

```
<modificador>* <tipo de retorno> <nombre> ( <argumento>>* ) {
    <sentencias>*
    return valorRetorno;
}
```

- <nombre>: puede ser cualquier identificador válido, con algunas restricciones basadas en los nombres que ya están en uso.
- <modificador>: el segmento es opcional y puede contener varios modificadores diferentes incluyendo a public, protected y private. Aunque no está limitado a estos.
- <tipo de retorno>: el tipo de retorno indica el tipo de valor devuelto por el método. Si el método no devuelve un valor, debe ser declarado void. La tecnología Java es rigurosa acerca de los valores de retorno. Si el tipo de retorno en la declaración del método es un int, por ejemplo, el método debe devolver un valor int desde todos los posibles caminos de retorno (y puede ser invocado solamente en contextos que esperan un int para ser devuelto). Se usa la sentencia return dentro de un método para devolver un valor.
- <argumento>: permite que los valores de los argumentos sean pasados hacia el método. Los elementos de la lista están separados por comas y cada elemento consiste en un tipo y un identificador.

Existen tres tipos de métodos: métodos de consulta, métodos de consulta y operaciones Los métodos de consulta sirven para extraer información de los objetos, los métodos modificadores sirven para modificar el valor de los atributos del objeto y las operaciones definen el comportamiento de un objeto.

Para acceder a los atributos de un objeto se definen los métodos get y set. Los métodos get se utilizan para consultar el estado de un objeto y los métodos set para modificar su estado. Un método get se declara public y a continuación se indica el tipo de dato que devuelve. Es un método de consulta. La lista de parámetros de un método get queda vacía. En el cuerpo del método se utiliza return para devolver el valor correspondiente al atributo que se quiere devolver, y al cual se hace referencia como this.nombreAtributo.

Por otra parte, un método set se declara public y devuelve void. La lista de parámetros de un método set incluye el tipo y el valor a modificar. Es un método modificador. El cuerpo de un método set asigna al atributo del objeto el parámetro de la declaración.

Por último, un método de tipo operación es aquel que realiza un cálculo o modifica el estado de un objeto. Este tipo de métodos pueden incluir una lista de parámetros y puede devolver un valor o no. Si el método no devuelve un valor, se declara void.

Invocación de métodos

Un método se puede invocar dentro o fuera de la clase donde se ha declarado. Si el método se invoca dentro de la clase, basta con indicar su nombre. Si el método se invoca fuera de la clase entonces se debe indicar el nombre del objeto y el nombre del método. Cuando se invoca a un método ocurre lo siguiente:

- En la línea de código del programa donde se invoca al método se calculan los valores de los argumentos.
- Los parámetros se inicializan con los valores de los argumentos.
- Se ejecuta el bloque código del método hasta que se alcanza return o se llega al final del bloque.
- Si el método devuelve un valor, se sustituye la invocación por el valor devuelto.
- La ejecución del programa continúa en la siguiente instrucción donde se invocó el método.

Parámetros y argumentos

Los parámetros de un método definen la cantidad y el tipo de dato de los valores que recibe un método para su ejecución. Los argumentos son los valores que se pasan a un método durante su invocación. El método recibe los argumentos correspondientes a los parámetros con los que ha sido declarado. Un método puede tener tantos parámetros como sea necesario. La lista de parámetros de la cabecera de un método se define con la siguiente sintaxis:

```
tipo nombre [,tipo nombre ]
```

Durante la invocación de un método es necesario que el número y el tipo de argumentos coincidan con el número y el tipo de parámetros declarados en la cabecera del método. Durante el proceso de compilación se comprueba que durante la invocación de un método se pasan tantos argumentos como parámetros tiene declarados y que además coinciden los tipos. Esta es una característica de los lenguajes que se denominan "strongly typed" o "fuertemente tipados".

Paso de parámetros

Cuando se invoca un método se hace una copia de los valores de los argumentos en los parámetros. Esto quiere decir que, si el método modifica el valor de un parámetro, nunca se modifica el valor original del argumento.

Cuando se pasa una referencia a un objeto se crea un nuevo alias sobre el objeto, de manera que esta nueva referencia utiliza el mismo espacio de memoria del objeto original y esto permite acceder al objeto original.

El valor de retorno

Un método puede devolver un valor. Los métodos que no devuelven un valor se declaran void, mientras que los métodos que devuelven un valor indican el tipo que devuelven: int, double, char, String o un tipo de objeto.

Sobrecarga de métodos

La sobrecarga de métodos es útil para que el mismo método opere con parámetros de distinto tipo o que un mismo método reciba una lista de parámetros diferente. Esto quiere decir que puede haber dos métodos con el mismo nombre que realicen dos funciones distintas. La diferencia entre los métodos sobrecargados está en su declaración, y más específicamente, en la cantidad y tipos de datos que reciben.

ABSTRACCIÓN Y ENCAPSULAMIENTO

La abstracción es la habilidad de ignorar los detalles de las partes para enfocar la atención en un nivel más alto de un problema. El encapsulamiento sucede cuando algo es envuelto en una capa protectora. Cuando el encapsulamiento se aplica a los objetos, significa que los datos del objeto están protegidos, "ocultos" dentro del objeto. Con los datos ocultos, ¿cómo puede el resto del programa acceder a ellos? (El acceso a los datos de un objeto se refiere a leerlos o modificarlos.) El resto del programa no puede acceder de manera directa a los datos de un objeto; lo tiene que hacer con ayuda de los métodos del objeto. Al hecho de proteger los datos o atributos con los métodos se denomina encapsulamiento.

ABSTRACCIÓN

La abstracción es la propiedad que considera los aspectos más significativos o notables de un problema y expresa una solución en esos términos. La abstracción posee diversos grados o niveles de abstracción, los cuales ayudan a estructurar la complejidad intrínseca que poseen los sistemas del mundo real. La abstracción encarada desde el punto de vista de la programación orientada a objetos es el mecanismo por el cual se proveen los límites conceptuales de los objetos y se expresan sus características esenciales, dejando de lado sus características no esenciales. Si un objeto tiene más características de las necesarias los mismos resultan difíciles de usar, modificar, construir y comprender. En el análisis hay que concentrarse en ¿Qué hace? y no en ¿Cómo lo hace?

ENCAPSULAMIENTO

La encapsulación o encapsulamiento significa reunir en una cierta estructura a todos los elementos que a un cierto nivel de abstracción se pueden considerar pertenecientes a una misma entidad, y es el proceso de agrupamiento de datos y operaciones relacionadas bajo una misma unidad de programación, lo que permite aumentar la cohesión de los componentes del sistema.

El encapsulamiento oculta lo que hace un objeto de lo que hacen otros objetos y del mundo exterior por lo que se denomina también ocultación de datos. Un objeto tiene que presentar "una cara" al mundo exterior de modo que se puedan iniciar sus operaciones.

Los métodos operan sobre el estado interno de un objeto y sirven como el mecanismo primario de comunicación entre objetos. Ocultar el estado interno y hacer que toda interacción sea a través de los métodos del objeto es un mecanismo conocido como encapsulación de datos.

MODIFICADORES DE ACCESO

Todas las clases poseen diferentes niveles de acceso en función del modificador de acceso (visibilidad). A continuación, se detallan los niveles de acceso con sus símbolos correspondientes:

- Public: Este modificador permite a acceder a los elementos desde cualquier clase, independientemente de que esta pertenezca o no al paquete en que se encuentra el elemento.

- Private: Es el modificador más restrictivo y especifica que los elementos que lo utilizan sólo pueden ser accedidos desde la clase en la que se encuentran. Este modificador sólo puede utilizarse sobre los miembros de una clase y sobre interfaces y clases internas, no sobre clases o interfaces de primer nivel, dado que esto no tendría sentido. Es importante destacar también que el modificador private convierte los elementos en privados para otras clases, no para otras instancias de la clase. Es decir, un objeto de una determinada clase puede acceder a los miembros privados de otro objeto de la misma clase.

- **Protected:** Este modificador indica que los elementos sólo pueden ser accedidos desde su mismo paquete y desde cualquier clase que extienda la clase en que se encuentra, independientemente de si esta se encuentra en el mismo paquete o no. Este modificador, como private, no tiene sentido a nivel de clases o interfaces no internas.

Si no especificamos ningún modificador de acceso se utiliza el nivel de acceso por defecto (Default), que consiste en que el elemento puede ser accedido sólo desde las clases que pertenezcan al mismo paquete. Los distintos modificadores de acceso quedan resumidos en la siguiente tabla

Visibilidad	Public	Private	Protected	Default
Desde la misma Clase	SI	SI	SI	SI
Desde cualquier Clase del mismo Paquete	SI	SI	SI	SI
Desde una Subclase del mismo Paquete	SI	SI	SI	SI
Desde una Subclase fuera del mismo Paquete	SI	NO	SI, a través de la herencia	NO
Desde cualquier Clase fuera del Paquete	SI	NO	NO	NO

ATRIBUTOS Y METODOS ESTÁTICOS

Un atributo o un método de una clase se puede modificar con la palabra reservada static para indicar que este atributo o método no pertenece a las instancias de la clase si no a la propia clase.

Se dice que son atributos de clase si se usa la palabra clave static: en ese caso la variable es única para todas las instancias (objetos) de la clase (ocupa un único lugar en memoria), es decir que, si se poseen múltiples instancias de una clase, cada una de ellas no tendrán una copia propia de este atributo, si no que todas estas instancias compartirán una misma copia del atributo. A veces a las variables de clase se les llama variables estáticas. Si no se usa static, el sistema crea un lugar nuevo para esa variable con cada instancia (la variable es diferente para cada objeto).

En el caso de una constante no tiene sentido crear un nuevo lugar de memoria por cada objeto de una clase que se cree. Por ello es adecuado el uso de la palabra clave static. Cuando usamos *"static final"* se dice que creamos una constante de clase, un atributo común a todos los objetos de esa clase.

ATRIBUTOS FINALES

En este contexto indica que una variable es de tipo constante: no admitirá cambios después de su declaración y asignación de valor. La palabra reservada *final* determina que un atributo no puede ser sobrescrito o redefinido, es decir, no funcionará como una variable "tradicional", sino como una constante. Toda constante declarada con *final* ha de ser inicializada en el mismo momento de declararla. El modificador *final* también se usa como palabra clave en otro contexto: una clase *final* es aquella que no puede tener clases que la hereden. Lo veremos más adelante cuando hablemos sobre herencia.

Cuando se declaran constantes es muy frecuente que los programadores usen letras mayúsculas (como práctica habitual que permite una mayor claridad en el código), aunque no es obligatorio.

PREGUNTAS DE APRENDIZAJE

1) Responda Verdadero (V) o Falso (F)

Java es un lenguaje de tipado estático, por lo tanto, todas las variables deben ser declaradas antes de ser utilizadas.	( )	( )
El acceso a una variable local no inicializada no da error	( )	( )
Cuando se coloca la palabra final precediendo la declaración de una variable la misma se transforma en una constante	( )	( )
Una variable local no puede ser declarada en cualquier lugar del cuerpo de una clase o método	( )	( )
Tanto las operaciones de lectura como las de escritura son de la clase java.net	( )	( )
El método constructor de una clase puede tener cualquier nombre	( )	( )
Cuando un método no devuelve ningún valor se utiliza la palabra reservada void para indicar que no devuelve nada	( )	( )
La palabra reservada final determina que un atributo no puede ser redefinido	( )	( )
La devolución de un valor a través de un método se hace con la sentencia restore	( )	( )

2) Un lenguaje fuertemente tipado:

a) Es un lenguaje en el que las variables pueden cambiar de tipos de datos si se especifica previamente.
b) No es necesario declarar todas las variables siempre y cuando pasen como parámetros a los métodos.
c) Deben declararse todas las variables que se utilicen.
d) Ninguna de las anteriores.

3) El llamado de una librería en Java se hace usando el:

a) Import
b) Scanner
c) String
d) Ninguna de las anteriores

4) ¿Cuál es la descripción que crees que define mejor el concepto clase en la programación orientada a objetos?

a) Es un concepto similar al de array
b) Es un tipo particular de variable
c) Es un modelo o plantilla a partir de la cual creamos objetos
d) Es una categoría de datos ordenada secuencialmente

5) ¿Qué elementos crees que definen a un objeto?

a) Su cardinalidad y su tipo
b) Sus atributos y sus métodos
c) La forma en que establece comunicación e intercambia mensajes
d) Su interfaz y los eventos asociados

6) ¿Qué significa instanciar una clase?

a) Duplicar una clase
b) Eliminar una clase
c) Crear un objeto a partir de la clase
d) Conectar dos clases entre sí

7) Queremos crear una clase Java con variables miembro que puedan ser accedidas, ¿qué opción elegirías como la mejor?

a) Variables miembro públicas
b) Variables miembro static
c) Variables miembro privadas con getters y setters
d) Ninguna de las anteriores

8) ¿Qué permite agrupar la palabra package?

a) Clases e interfaces
b) Agrupar códigos
c) Agrupar signos
d) Agrupar secuencias

9) Los nombres de los paquetes son:

a) Complejos
b) Para agrupar clases y evitar conflictos con los nombres entre las clases importadas como también para ayudar en el control de la accesibilidad de clases y miembros.
c) Para almacenar directorios
d) Ninguna de las anteriores

10) ¿Cuáles son las dos formas para utilizar import?

a) Para una clase y para todo un package
b) Para dos clases de package
c) Para Java.io
d) Para Java.net

11) Se crean anteponiendo la palabra static a su declaración:

a) Variables miembro de objeto
b) Variables miembro de clase
c) Variables finales
d) Ninguna de las anteriores

12) No puede cambiar su valor durante la ejecución del programa:

a) Variables miembro de objeto
b) Variables miembro de clase
c) Variables finales
d) Todas las anteriores

13) ¿Qué es Netbeans?

a) Una librería de Java
b) Una versión de Java especial para servidores
c) Un IDE para desarrollar aplicaciones
d) Ninguna de las anteriores

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.

A partir de esta guía todos los ejemplos de los videos serán desarrollados siguiendo el ejemplo de un Tinder de Mascotas. A continuación se presenta el contexto del problema para entender los objetos del mundo real que vamos a modelar.

VER VIDEOS:

A. Introducción
B. Contexto del Tinder de Mascotas

Realizar los siguientes ejercicios:

VER VIDEOS:

A. Introducción a Objetos
B. Método Constructor
C. Encapsulamiento Parte 1
D. Encapsulamiento Parte 2

1. Crear una clase llamada Libro que contenga los siguientes atributos: ISBN, Título, Autor, Número de páginas, y un constructor que inicialice esos atributos con los valores pasados como parámetros. Definir una instancia para cargar un libro y luego informar mediante otro método el número de ISBN, el título y el autor del libro.

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 calcular el área de la circunferencia ($Area = \pi * radio^2$).
d) Método para calcular el perímetro ($Perimetro = 2 * \pi * radio$).

3. Crear una clase llamada Fraccion que contenga métodos para sumar, restar, multiplicar y dividir fracciones. Los argumentos de cada método son el numerador y denominador según corresponda. La clase también debe contener un método constructor con parámetros.

4. Definir una clase llamada Punto con un constructor que inicialice las coordenadas x e y. Generar dos instancias, es decir, crear dos objetos llamados punto1 y punto2 y comprobar la distancia que existe entre ambos. Para conocer como calcular la distancia entre dos puntos consulte el siguiente link: http://www.matematicatuya.com/GRAFICAecuaciones/S1a.html.

5. Definir la clase Tiempo, la cual tendrá la hora, minutos y segundos. Definir dos constructores: un constructor vacío y otro con la hora, minutos y segundos ingresado por el usuario. Deberá definir además, los métodos getters y setters correspondientes, y el método imprimirHoraCompleta().

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

7. Crear una clase Rectángulo que modele rectángulos por medio de un atributo privado alto y un atributo privado largo, dados por el usuario. La clase también incluirá un método para calcular la superficie del rectángulo, un método para calcular el perímetro del rectángulo y otro que desplace el rectángulo en el plano. Y 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.

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

VER VIDEO: Clases Entidad y Control

8. Realizar una clase llamada Cuenta (bancaria) que debe tener como mínimo los atributos: numeroCuenta (entero), el DNI del cliente (entero largo), el saldo actual y el interés anual que se aplica a la cuenta (porcentaje). Las operaciones asociadas a dicha clase son:

- Constructor por defecto y constructor con DNI, saldo, número de cuenta e interés.
- Agregar los métodos getters y setters correspondientes con los que llenaremos el objeto en el Main.
- Método actualizarSaldo(): actualizará el saldo de la cuenta aplicándole el interés diario (interés anual dividido entre 365 aplicado al saldo actual)
- Método ingresar(double): permitirá ingresar una cantidad en la cuenta bancaria.
- Método retirar(double): permitirá sacar una cantidad de la cuenta (si hay saldo).
- Método consultarSaldo(): permitirá consultar el saldo disponible en la cuenta.
- Método consultarDatos(): permitirá mostrar todos los datos de la cuenta.

9. 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

10. Crear una clase Fecha con atributos para el día, mes y año. Se debe incluir al menos los siguientes métodos:

- Constructor predeterminado con el 1-1-1900 como fecha por defecto
- Constructor parametrizado con día, mes y año ingresados por el usuario.
- Método leer() para pedir al usuario el día (1 a 31), el mes (1 a 12) y el año (1900 a 2050).
- Método bisiesto() para indicar si el año es bisiesto o no.
- Método diasMes(int) para devolver el número de días del mes indicado (para el año de la fecha).
- Método validar() para comprobar si la fecha es correcta (entre el 1-1-1900 y el 31-12-2050). Si el día no es correcto, se pondrá en 1; si el mes no es correcto se pondrá en 1; y si el año no es correcto lo pondrá en 1900. Esto último se realizará mediante los métodos setters de cada atributo. Los setters también se llamarán en el constructor parametrizado y en el método leer().
- Método diasTranscurridos() para devolver la cantidad de días transcurridos desde el 1-1-1900 hasta la fecha ingresada por el usuario.
- Método diasEntre(Fecha) para devolver el número de días entre la fecha ingresada por el usuario y la proporcionada como parámetro.
- Método siguiente() para devolver el día siguiente del día de la fecha ingresada.
- Método anterior() para devolver el día anterior del día de la fecha ingresada.

11. 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: establece la capacidad máxima en 1000 (c.c.) y la actual en cero (cafetera vacía).
- Constructor con la capacidad máxima de la cafetera; inicializa la cantidad actual de café igual a la capacidad máxima.
- Constructor con la capacidad máxima y la cantidad actual. Si la cantidad actual es mayor que la capacidad máxima de la cafetera, la ajustará al máximo.
- Métodos getters y setters.
- Método llenarCafetera():hace que la cantidad actual sea igual a la capacidad.
- Método servirTaza(int): simula la acción de servir una taza con la capacidad indicada. Si la cantidad actual de café "no alcanza" para llenar la taza, se sirve lo que quede.
- Método vaciarCafetera(): pone la cantidad de café actual en cero.
- Método agregarCafe(int): añade a la cafetera la cantidad de café indicada

12. Dígito Verificador. Crear una clase Letra que se usará para mantener DNIs con su correspondiente letra. Los atributos serán el número de DNI (entero largo) y la letra que le corresponde. La clase dispondrá de los siguientes métodos:

- Constructor predeterminado que inicialice el nº de DNI a 0 y la letra a espacio en blanco (será un NIF no válido).
- Constructor que reciba el DNI y establezca la letra que le corresponde.
- Métodos getters y setters para el número de DNI (que ajuste también automáticamente la letra).
- Método leer(): para pedir el número de DNI (ajustando automáticamente la letra)
- Método que nos permita mostrar el NIF (ocho dígitos, un guión y la letra en mayúscula; por ejemplo: 00395469-F).

La letra correspondiente al dígito verificador se calculará a través 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 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

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

- Un método para mostrar la longitud de la palabra que se debe encontrar.
- Un método para buscar si una letra ingresada por el usuario es parte de la palabra o no.
- Un método para informar error o acierto.
- Un método para mostrar cuantas oportunidades le queda al jugador.
- Un método que al pedir ingresar una letra muestre que letras han sido encontradas y en qué posición. Además, se debe mostrar también cuántas oportunidades quedan.
- Un método para buscar si se encontró la palabra completa.
- En el método main() se deberá pedir al usuario una letra hasta que el usuario haya gastado todas sus oportunidades o bien hasta que encuentre la palabra.

Un ejemplo de salida puede ser así:

Longitud de la palabra: 6

Ingrese una letra:

a

Mensaje: La letra pertenece a la palabra
Número de letras (encontradas,faltantes): (3,4)
Número de oportunidades restantes: 3
Estado Actual: a a a

Ingrese una letra:

z

Mensaje: La letra no pertenece a la palabra
Número de letras (encontradas,faltantes): (3,4)
Número de oportunidades restantes: 2
Estado Actual: a a a

Ingrese una letra:

b

Mensaje: La letra no pertenece a la palabra
Número de letras (encontradas,faltantes): (3,4)
Número de oportunidades restantes: 2
Estado Actual: a a b a

Ingrese una letra:

u

Mensaje: La letra no pertenece a la palabra
Número de letras (encontradas,faltantes): (3,4)
Número de oportunidades restantes: 1
Estado Actual: a a a

Ingrese una letra:

q

Mensaje: La letra no pertenece a la palabra
Mensaje: Lo sentimos, no hay más oportunidades

14. 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
.

Se implementarán tres constructores:

- Un constructor por defecto.
- Un constructor con el nombre, edad y sexo recibidos como parámetro; y el resto de los atributos se inicializarán con valores por defecto.
- Un constructor con todos los atributos como parámetro.

Los métodos que se implementarán son:

- Método calcularIMC(): calculara si la persona está en su peso ideal (peso en kg/(altura^2 en m)). Si esta fórmula da por resultado un valor menor que 20, la función devuelve un -1. Si la fórmula da por resultado un número entre 20 y 25 (incluidos), significa que el peso está por debajo de 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. Se recomienda hacer uso de constantes para devolver estos valores.

- Método esMayorDeEdad(): indica si la persona es mayor de edad. La función devuelve un booleano.
- Método comprobarSexo(char sexo): comprueba que el sexo introducido sea correcto, es decir, H M ó O. Si no es correcto se deberá mostrar un mensaje.
- Métodos getters y setters de cada atributo.

A continuación, crear una clase ejecutable que haga lo siguiente:

Pedir por teclado el nombre, la edad, sexo, peso y altura. Luego se crearán 3 objetos de la clase Persona de la siguiente manera: el primer objeto obtendrá los valores pedidos por teclado, el segundo objeto obtendrá del usuario todos los atributos menos el peso y la altura, y el tercer objeto será inicializado con valores por defecto. Para este último utiliza los métodos set para darle a los atributos un valor.

Para cada objeto, deberá comprobar si la persona está en su peso ideal, tiene sobrepeso o está por debajo de su peso ideal. Mostrar un mensaje.

Indicar para cada objeto si la persona es mayor de edad.

Por último, se debe mostrar la información completa de cada objeto, es decir, los valores de todos sus atributos.

Clases de Utilidad en Java

Los métodos disponibles para las clases de utilidad String, Integer, Math, Array, Date, Calendar y GregorianCalendar se pueden consultar el “Apéndice B”.

VER VIDEO: Clases String e Integer

Ver más acerca de la clase String

Ver más acerca de la clase Integer

15. Realizar una clase llamada Cadena que tenga como atributos una frase (de tipo de String) y su longitud. La clase deber tener un constructor mediante el cual se inicialice la frase, con una o más palabras ingresadas por el usuario (separadas entre sí por un espacio en blanco) y se inicialice de manera automática su longitud. Deberá además implementar los siguientes métodos:

- Método mostrarVocales(), deberá contabilizar la cantidad de vocales que tiene la frase ingresada.
- Método invertirFrase(), deberá invertir la frase ingresada y mostrar la frase invertida por pantalla. Por ejemplo: Entrada: "casa blanca", Salida: "acnalb asac".
- Método vecesRepetido(), deberá recibir por parámetro un carácter ingresado por el usuario y contabilizar cuántas veces se repite el carácter en la frase, por ejemplo:

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

- Método compararLongitud(), deberá comparar la longitud de la frase que compone la clase con otra nueva frase ingresada por el usuario.
- Método unirFrases(), deberá unir la frase contenida en la clase Cadena con una nueva frase ingresada por el usuario y mostrar la frase resultante.
- Método reemplazar(), deberá reemplazar todas las letras "a" que se encuentren en la frase, por algún otro carácter seleccionado por el usuario.

VER VIDEO: Método Static y Clase Math

16. 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 mediante el cual se inicialicen ambos valores en cero. Deberá además implementar los siguientes métodos:

    - Método devolverMayor() para retornar cuál de los dos atributos tiene el mayor valor
    - Método calcularPotencia() para calcular la potencia del mayor valor de la clase elevado al menor número. Previamente se deben redondear ambos valores.
    - 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.
    - Método sumaAngulos(), para calcular la suma de dos ángulos:

      sen(a+b) = sen(a)*cos(b)+cos(a)*sen(b)

      donde a y b son los dos valores que componen la clase.

VER VIDEO: Clase Arrays

Ver más acerca de la clase Arrays

17. 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

    No ha acertado. Intente adivinarlo introduciendo otro mes: octubre

    ¡Ha acertado!

18. Realizar un programa en Java donde se creen dos arreglos: el primero será un arreglo de 50 números reales, y el segundo, un arreglo de 20 números, también reales. El programa deberá inicializar el arreglo con números aleatorios y mostrarlo por pantalla. Luego, el arreglo se debe ordenar de menor a mayor y copiar los primeros 10 números ordenados al segundo arreglo 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.

VER VIDEOS:

Clase Date

Clases Calendar y GregorianCalendar

Ver más acerca de la clase Date

Ver más acerca de las clases Calendar y GregorianCalendar

19. Implemente la clase Persona. Una persona tiene un nombre y una fecha de nacimiento, por lo que debe ser posible pedirle su nombre, fecha de nacimiento y edad.

- Métodos get y set para llenar el objeto en el Main.
- 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(Persona persona) que recibe como parámetro a otra persona 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.
- Agregar un método de creación del objeto que respete la siguiente firma: Persona(String nombre, Date fechaNacimiento). Este método recibe como parámetros el nombre y la fecha de nacimiento de la persona a crear y retorna un objeto inicializado con los valores recibidos como parámetro.