Ejercicio 1: Conversión

La gravedad de la Luna es aproximadamente el 17% de la gravedad de la Tierra. Se pide:

1. Crear un programa que, dado un peso de la Tierra, calcule su equivalente en la Luna.
2. El programa debe mostrar por pantalla un mensaje similar a este: “Un peso de 30 kilos en la Tierra equivalen a X kilos en la Luna”.

Ejercicio 2: Operaciones matemáticas

Crear un programa que realice algunas operaciones matemáticas.

1. Pedir por consola al usuario que introduzca 3 números enteros y almacenar su valor en variables.
2. Operación 1. a*b/c Mostrar por consola el resultado
3. Operación 2. (a*c)%bMostrar por consola el resultado
4. Operación 3. 2*(a+c-b)/(b*c) Mostrar por consola el resultado
5. Operación 4. ((a*c)+(b%a))/a-c Mostrar por consola el resultado
6. Opcional: Repetir las operaciones con números decimales para ver las diferencias de resultado.

Ejercicio 3: Notas

Crear un programa que indique al usuario la nota media global de su curso. Requisitos:

1. El programa pedirá al usuario que introduzca 5 notas, para las asignaturas: Matemáticas, Física, Química, Lenguaje e Historia.
2. El programa realizará la media de esas cinco notas.
3. El programa indicará al usuario el rango al que equivale la nota final que ha sacado:
    a. Entre 0 – 3 Muy deficiente
    b. Entre 3 – 5 Insuficiente
    c. Entre 5 – 6 Suficiente
    d. Entre 6 – 7 Bien
    e. Entre 7 – 9 Notable
    f. Entre 9 – 10 Sobresaliente

Ejercicio 4: Contar espacios

Crear un programa que lea caracteres desde teclado indefinidamente. Requisitos:

1. El programa pedirá al usuario que inserte caracteres o frases completas por teclado de manera indefinida
2. El programa contará el número de espacios que se van introduciendo por consola
3. El programa finalizará cuando el usuario introduzca un punto “.”
4. El programa imprimirá por consola el número total de espacios que se han introducido

Ejercicio 5: Amstron

Desarrollar un programa que determine si un número es un número de Amstrong. Un número de Amstrong es aquel que es igual a la suma de sus dígitos elevados a la potencia de su número de cifras.

Recursos para el ejercicio:

- Math.floor: Devuelve el máximo entero menor o igual a un número pasado como parámetro.
- Math.log10: Devuelve el logaritmo en base 10 de un número pasado como parámetro.
- Math.pow: Devuelve el valor del primer argumento elevado a la potencia del segundo argumento.

Ejercicio 6: Conversión 2

Al igual que se hizo con la Ejercicio 1 de conversión de pesos, en este caso se pide crear un programa que imprima una tabla de conversión de pulgadas a metros. Requisitos:

1. Un metros son 39,37 pulgadas
2. Elaborar el programa desde la pulgada 1 hasta la 144
3. La tabla debe de dejar un espacio libre cada 12 pulgadas para ser más legible.

Ejercicio 7: CandyCalculator

En las fiestas de un barrio hay competiciones deportivas que premian el desempeño en ellas con cupones que luego puedes cambiar por golosinas. Una barra de caramelo se puede cambiar por 10 cupones, y un chicle por 3 cupones.

Escribe una clase CandyCalculator que tenga un método candyCalculator que permita:

1. Calcular, dado un número dado de cupones, cuantas barras de caramelo y chicles puedes obtener si gastas todos tus cupones en barras de caramelo primero, y utilizas los cupones restantes en chicles.
2. También te tiene que devolver el número de cupones restantes que no puedes gastar.
3. Devuelve los valores en un array en donde la posición [0] representa las barras de caramelo, la posición [1] los chicles y la posición [2] los cupones restantes.

Ejercicio 8: Elementos Duplicados

Crear un programa que, dado un array de números enteros, determine cuales son sus elementos que se encuentran duplicados. Por ejemplo:

int [] arrayDePrueba = {1, 2, 3, 3, 9, 8, 7, 4, 6, 7, 0, 4, 5};

Ejercicio 9: Contar vocales

Implementa la función int contarVocales(String) que dada una cadena, cuenta el número de vocales que existe en la misma. No importa que las vocales estén en mayúsculas o minúsculas y el resto de caracteres que no sean vocales son ignorados. También se ignoran las vocales acentuadas.

Ejercicio 10: Persona

Se requiere un programa que modele el concepto de una persona. Una persona posee nombre, apellido, número de documento de identidad y año de nacimiento. La clase debe tener un constructor que inicialice los valores de sus respectivos atributos.

La clase debe incluir los siguientes métodos:

1. Definir un método que imprima por pantalla los valores de los atributos del objeto.
2. En el método main se deben crear dos personas y mostrar los valores de sus atributos por pantalla.

Ejercicio 11: Calculadora Básica

Crea una clase en Java llamada Calculadora que implemente una calculadora básica con las siguientes funcionalidades:

Suma: Un método que toma dos números como parámetros y devuelve la suma de ambos.

Resta: Un método que toma dos números como parámetros y devuelve la resta del primero menos el segundo.

Multiplicación: Un método que toma dos números como parámetros y devuelve su producto.

División: Un método que toma dos números como parámetros y devuelve el cociente resultante de la división del primero entre el segundo. Si el segundo número es cero, debe mostrar un mensaje indicando que la división por cero no está permitida.

En el programa principal, crea una instancia de la clase Calculadora, realiza varias operaciones y muestra los resultados.

Ejercicio 12: Rectángulo

Crea una clase Rectángulo con las siguientes características

- Incluyen un constructor al que se le pasan los datos de base y altura. Si se intenta dar valor negativo a alguna de las dimensiones, lo corrige al valor positivo usando Math.abs(int a).
- Un constructor sin parámetros que inicializa un nuevo rectángulo con base 2 y altura 1.
- Un método esCuadrado que indica si el rectángulo actual es un cuadrado.
- Un método area que calcula el área del rectángulo actual.
- Un método perimetro que calcula el perímetro del rectángulo actual.
- Un método gira que gira 90 grados el rectángulo actual (intercambiado la base por la altura).

Ejercicio 13: Sistema de Gestión de Animales

Crea un sistema de gestión de animales utilizando herencia en Java. Define una clase base llamada Animal con los siguientes atributos: nombre y edad. Implementa un constructor y métodos de acceso para estos atributos.

Luego, crea dos clases derivadas: Perro y Gato. Ambas clases deben heredar de la clase base Animal. La clase Perro debe incluir un atributo adicional para representar la raza del perro, y la clase Gato debe incluir un atributo para representar el color del pelaje.

En el programa principal, crea instancias de ambas clases, establece valores para sus atributos y muestra la información básica de cada animal, incluyendo su nombre, edad y características específicas de la especie.

Ejercicio 14: Figuras geométricas

Crea una jerarquía de clases para representar figuras geométricas. La clase base debe ser Figura y debe contener métodos para calcular el área y el perímetro. Luego, implementa subclases para representar diferentes tipos de figuras como Círculo, Rectángulo y Triángulo. Cada subclase debe heredar de la clase Figura y proporcionar implementaciones específicas para los cálculos de área y perímetro. En el programa principal, crea instancias de estas clases y muestra los resultados de los cálculos.

Ejercicio 15: Sistema de Gestión de Empleados

Crea un sistema de gestión de empleados utilizando herencia en Java. Define una clase base llamada Empleado con los siguientes atributos: nombre, edad y salario. Implementa un constructor y métodos de acceso para estos atributos.

Luego, crea dos clases derivadas: EmpleadoTiempoCompleto y EmpleadoTiempoParcial. La clase EmpleadoTiempoCompleto debe incluir un atributo adicional para representar el cargo (por ejemplo, "Desarrollador" o "Gerente"). La clase EmpleadoTiempoParcial debe incluir un atributo para representar las horas trabajadas por semana.

Ambas clases derivadas deben heredar de la clase base Empleado. Implementa métodos específicos en cada clase derivada para calcular el salario total de un empleado. Para EmpleadoTiempoCompleto, el salario total se calcula sumando el salario base más un bono (por ejemplo, 10% del salario base). Para EmpleadoTiempoParcial, el salario total se calcula multiplicando las horas trabajadas por una tarifa por hora.

En el programa principal, crea instancias de ambas clases, establece valores para sus atributos y muestra la información detallada de cada empleado, incluyendo su salario total.

Ejercicio 16: Sistema de Biblioteca

Diseña un sistema de gestión para una biblioteca utilizando principios de Programación Orientada a Objetos (POO) en Java. Debes implementar al menos las siguientes clases:

Libro: Representa un libro con atributos como título, autor, año de publicación, y disponibilidad (si está prestado o no).

Lector: Representa a una persona que puede tomar libros prestados de la biblioteca. Debe tener atributos como nombre, número de identificación, y una lista de libros prestados.

Biblioteca: Representa la biblioteca en sí. Debe contener una lista de libros disponibles, una lista de libros prestados y una lista de lectores registrados. Implementa métodos para prestar un libro a un lector, devolver un libro, mostrar información sobre los libros disponibles y los lectores registrados.

Programa Principal: En el programa principal, crea instancias de libros, lectores y la biblioteca. Realiza algunas operaciones como registrar un lector, agregar libros a la biblioteca, prestar libros a los lectores y mostrar información relevante.

Este ejercicio te permitirá aplicar conceptos de POO como encapsulamiento, herencia y polimorfismo. Además, deberías considerar la relación entre las clases (por ejemplo, un lector puede tener varios libros prestados, un libro puede estar en la biblioteca o prestado a un lector).

Ejercicio 17: Sistema de Autenticación

Crea una interfaz llamada Autenticable que tenga un método autenticar que devuelve un valor booleano. Luego, implementa esta interfaz en clases como Usuario y Administrador. Cada clase debe proporcionar su propia implementación del método autenticar. En la clase principal, crea instancias de estas clases y demuestra la autenticación.

Ejercicio 18: Sistema de Gestión de Empleados

Crea una interfaz llamada Empleado que tenga métodos para calcular el salario y mostrar la información del empleado. Luego, implementa esta interfaz en clases como EmpleadoAsalariado, EmpleadoPorHoras y EmpleadoComision. Cada clase debe tener propiedades específicas, como salario mensual, tarifa por hora, ventas realizadas, etc.

En la clase principal, crea un array o una lista de objetos Empleado que contenga instancias de las clases anteriores. Calcula y muestra el salario de cada empleado y su información.

Ejercicio 19: Números capicúa

Escribir un programa que pida por teclado un número entero N de más de una cifra y verifique si es capicúa.

Ejercicio 20: FizzBuzz

Escribe un programa en Java que imprima los números del 1 al 20. Sin embargo, para los múltiplos de 3, imprime "Fizz" en lugar del número, y para los múltiplos de 5, imprime "Buzz". Para los números que son múltiplos de ambos 3 y 5, imprime "FizzBuzz".

Ejemplo de salida esperada:

1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16 17 Fizz 19 Buzz

Ejercicio 21: Análisis de Datos

Supongamos que estás desarrollando un programa para analizar datos de temperatura. Crea un programa en Java que realice las siguientes tareas:

- Crea un array de temperaturas temperaturas que almacene las temperaturas de una semana (por ejemplo, los valores de lunes a domingo).

- Implementa un método que calcule y devuelva la temperatura media de la semana.
- Implementa un método que encuentre y devuelva la temperatura máxima de la semana, así como el día en que ocurrió.
- Implementa un método que encuentre y devuelva la temperatura mediana de la semana, así como el día en que ocurrió.
- En el método main, crea un array con temperaturas de ejemplo y utiliza los métodos implementados para mostrar la temperatura media, máxima y mínima de la semana.

Ejercicio 22: Gestión de Librería

Supongamos que estás desarrollando un sistema para gestionar una librería. Crea las clases necesarias para representar libros y clientes, y luego implementa un programa principal que demuestre la funcionalidad básica del sistema.

Crea una clase Libro con los siguientes atributos:

- titulo (String): el título del libro.
- autor (String): el autor del libro.
- anioPublicacion (int): el año de publicación del libro.
- precio (double): el precio del libro.

Crea una clase Cliente con los siguientes atributos:

- nombre (String): el nombre del cliente.
- direccion (String): la dirección del cliente.
- telefono (String): el número de teléfono del cliente.

Implementa métodos en ambas clases para establecer y obtener los valores de los atributos.

Crea una clase GestionLibreria que contenga el método main. En este método, instancia al menos dos libros y dos clientes. Luego, realiza las siguientes acciones:

- Imprime la información de cada libro y cliente.
- Cambia el precio de uno de los libros.
- Imprime la nueva información del libro modificado.
- Cambia la dirección de uno de los clientes.
- Imprime la nueva información del cliente modificado.

Ejercicio 23: Sistema de Biblioteca con Prestamos y Devoluciones

Vamos a crear un sistema para gestionar una biblioteca que incluye préstamos y devoluciones de libros. Utilizaremos una interfaz más compleja que involucre múltiples clases.

Crea una interfaz llamada ItemBiblioteca con los siguientes métodos:

- prestar(): método que simula el préstamo del ítem.
- devolver(): método que simula la devolución del ítem.
- estaPrestado(): método que devuelve un booleano indicando si el ítem está prestado o no.

Implementa la interfaz ItemBiblioteca en dos clases concretas:

- Libro: representa un libro en la biblioteca. Incluye atributos como titulo, autor, y numPaginas.
- DVD: representa un DVD en la biblioteca. Incluye atributos como titulo, director, y duracion.

Crea una clase Biblioteca que contiene una lista de ItemBiblioteca. Implementa un método llamado mostrarEstadoBiblioteca() que imprime el estado actual de todos los ítems en la biblioteca, indicando si están prestados o no.

En la clase GestionBiblioteca (que contiene el método main), crea instancias de varios libros y DVDs, agrégales a la biblioteca y realiza algunas operaciones de préstamo y devolución para simular el uso de la biblioteca.

Ejercicio 24: Cálculos numéricos

Se requiere definir una clase denominada CalculosNumericos que realice las siguientes operaciones:

1. Calcular el logaritmo neperiano recibiendo un valor double como parámetro. Este método debe ser estático. Si el valor no es positivo se genera una excepción aritmética.
2. Calcular la raíz cuadrada recibiendo un valor double como parámetro. Este método debe ser estático. Si el valor no es positivo se genera una excepción aritmética.
3. Se debe crear un método main que utilice dichos métodos ingresando un valor por teclado.
4. Se deben de capturar las excepciones.

Ejercicio 25: Sistema de Manejo de Cuentas Bancarias

Crea un programa que simule un sistema de cuentas bancarias. Cada cuenta bancaria tiene un saldo inicial y permite realizar operaciones como depósitos y retiros. Sin embargo, debes manejar adecuadamente las excepciones que puedan surgir durante estas operaciones.

Crea una clase CuentaBancaria con los siguientes atributos:

- saldo (double) para almacenar el saldo de la cuenta.
- Implementa un método en la clase CuentaBancaria llamado realizarDeposito que acepte un parámetro monto (double) y aumente el saldo de la cuenta en esa cantidad. Si el monto es negativo, lanza una excepción IllegalArgumentException con el mensaje "No se permiten depósitos negativos".
- Implementa otro método llamado realizarRetiro que acepte un parámetro monto (double) y disminuya el saldo de la cuenta en esa cantidad. Si el monto es negativo o mayor que el saldo actual, lanza una excepción IllegalArgumentException con el mensaje correspondiente.
- En el programa principal, crea una instancia de CuentaBancaria, solicita al usuario que realice un depósito y un retiro, y muestra el saldo resultante después de cada operación. Maneja las excepciones de manera adecuada para garantizar que el programa no se bloquee debido a entradas incorrectas del usuario.

Ejercicio 26: Ejercicio de ArrayList

Crea un programa en Java que gestione una lista de nombres mediante un ArrayList. El programa debe ofrecer las siguientes opciones:

1. Agregar un nombre a la lista.
2. Eliminar un nombre de la lista.
3. Imprimir la lista de nombres.
4. Buscar un nombre en la lista.
5. Salir del programa.

Implementa un menú que permita al usuario seleccionar una de las opciones mencionadas. Asegúrate de manejar adecuadamente la entrada del usuario y de proporcionar mensajes claros en cada operación. El programa debe continuar ejecutándose hasta que el usuario elija salir.

Ejercicio 27: Ordenar Lista de Objetos

Crea una clase Persona con campos como nombre, edad, etc. Crea un programa que tenga una lista de objetos de tipo Persona y ordene la lista según algún criterio (por ejemplo, por edad).

Ejercicio 28: Gestión de Personas

1. Crea una clase llamada Persona con los siguientes atributos privados:
   - nombre (String)
   - apellido (String)
   - genero (String)
   - edad (int)
2. La clase debe tener un constructor que acepte todos los atributos y un constructor predeterminado sin parámetros.
3. Proporciona métodos de acceso (getters y setters) para cada atributo.
4. En la clase Main, solicita al usuario ingresar los datos de varias personas (por ejemplo, 5 personas).
5. Crea una lista de objetos Persona y almacena en ella las personas ingresadas.
6. Implementa los siguientes métodos estáticos en la clase Main para realizar operaciones sobre la lista de personas:
   - retornarNombreGenero: Imprime el nombre, apellido y género de cada persona.
   - retornarPromedioEdades: Calcula y muestra el promedio de las edades de todas las personas.
   - retornarPersonasMasculinas: Muestra la cantidad de personas de género masculino.
   - retornarPersonasFemeninas: Muestra la cantidad de personas de género femenino.

Ejercicio 29: Ejercicio con Set

En una aplicación de gestión de estudiantes, se desea mantener un registro de los cursos en los que están inscritos los estudiantes. Diseña una clase Estudiante con atributos como nombre, número de estudiante, y un Set que almacene los nombres de los cursos en los que está inscrito el estudiante. Implementa métodos para agregar un curso, eliminar un curso y mostrar la lista de cursos de un estudiante específico. Asegúrate de manejar posibles duplicados de cursos para un estudiante.

Ejercicio 30: Ejercicio con Map

En una aplicación de inventario de una tienda, se desea realizar un seguimiento de la cantidad de productos disponibles en el inventario. Diseña una clase Inventario que utilice un Map para almacenar información sobre los productos. Cada producto está representado por un código único. Implementa métodos para agregar productos al inventario, actualizar la cantidad de existencias de un producto, y obtener la cantidad de existencias de un producto dado su código. Además, asegúrate de manejar casos en los que se intenta agregar un producto que ya existe en el inventario.

Ejercicio 31: Filtrar y Transformar

Supongamos que tienes una lista de objetos Persona con atributos como nombre, edad y ciudad. Escribe un programa que use Streams para realizar las siguientes operaciones:

1. Filtra las personas que tienen una edad mayor a 25.
2. Transforma el resultado para obtener una lista de cadenas que contienen el nombre y la ciudad de cada persona.

Ejercicio 32: Operaciones con Números

Dado un array de números, realiza las siguientes operaciones utilizando Streams:

1. Filtra los números pares.
2. Eleva al cuadrado cada número filtrado.
3. Suma los cuadrados obtenidos.

Ejercicio 33: Sistema de Gestión de Clientes

1. Crea una base de datos llamada GestionClientes.
2. Define una tabla llamada clientes con las siguientes columnas:
    - id (clave primaria, autoincremental)
    - nombre (cadena de texto)
    - apellido (cadena de texto)
    - edad (entero)

3. Implementa un programa en Java que interactúe con la base de datos GestionClientes. Este programa deberá realizar las siguientes operaciones:

4. Crear Cliente:
   - Solicita al usuario que ingrese los datos de un nuevo cliente (nombre, apellido, edad).
   - Inserta esos datos en la tabla clientes.
   - Muestra un mensaje indicando que el cliente ha sido creado con éxito.
5. Leer Clientes:
   - Recupera todos los clientes de la tabla clientes.
   - Muestra la información de cada cliente en la consola.
6. Actualizar Cliente:
   - Solicita al usuario que ingrese el ID del cliente que desea actualizar.
   - Pregunta al usuario qué información desea actualizar (nombre, apellido, edad).
   - Realiza la actualización en la base de datos.
   - Muestra un mensaje indicando que la actualización ha sido realizada con éxito.
7. Eliminar Cliente:
   - Solicita al usuario que ingrese el ID del cliente que desea eliminar.
   - Elimina el cliente correspondiente de la base de datos.
   - Muestra un mensaje indicando que el cliente ha sido eliminado con éxito.

Ejercicio 34: Ejercicio Avanzado de JDBC en Java

Crea una aplicación Java que gestione la información de una biblioteca. La base de datos de la biblioteca debe contener dos tablas: una para los libros y otra para los autores. Cada libro debe tener un identificador único, título, año de publicación, y una referencia al autor. Cada autor debe tener un identificador único, nombre y nacionalidad.

La aplicación debe realizar las siguientes operaciones:

1. Inserción de Datos:

- Permitir la inserción de nuevos autores y libros en la base de datos.

2. Consulta de Datos:

- Mostrar la lista de todos los libros con su título, año de publicación y nombre del autor.

3. Actualización de Datos:

- Permitir la actualización del año de publicación de un libro específico.

4. Eliminación de Datos:

- Permitir la eliminación de un autor y todos sus libros asociados.

5. Consulta Específica:

- Mostrar la lista de libros publicados en un año determinado, ordenados alfabéticamente por título.