TALLER Nº 3
IMPLEMENTACIÓN DE CLASES Y CREACIÓN DE OBJETOS

Implemente las aplicaciones que permitan resolver los enunciados siguientes:

1. Crear una clase Trabajador con los siguientes atributos: nombre, salario por hora y cantidad de horas laboradas. La clase cuenta con métodos que permiten calcular: el sueldo bruto (salario por hora * cantidad de horas laboradas), el descuento equivalente al 12% del sueldo bruto y el sueldo neto (sueldo bruto menos descuento). La aplicación debe permitir capturar los datos de un trabajador y luego mostrar los cálculos realizados.

2. Crear una clase Rectángulo, con los atributos longitud y ancho, ambos con valor por omisión de uno. La clase cuenta con métodos que calculan el perímetro y el área del rectángulo, así como métodos set y get para longitud y ancho. Los métodos set deben verificar que tanto la longitud como el ancho sean números reales mayores que 0 y menores que 20.

3. Crear una clase Factura con los atributos: nombre del cliente, la cantidad vendida, el precio de venta. La clase cuenta con métodos para calcular la venta bruta (cantidad vendida * precio de venta), el IGV (19% de la venta bruta) y la venta líquida (venta bruta – IGV).

4. Crear una clase MiFecha con las siguientes características:
   a. Exhibir la fecha en múltiples formatos, como:
      - ✓ MM/DD/AA
      - ✓ DD/MM/AA
      - ✓ Octubre 13, 2005
      - ✓ Octubre 2005
   b. Utilice constructores sobrecargados para crear objetos MiFecha inicializados con fechas en los formatos del punto a.

5. Crear una clase MiTiempo que permita registrar el tiempo (hora, minuto y segundo). Además de los métodos set y get debe contener un método IncrementarSegundo que incrementa el tiempo almacenado en un objeto MiTiempo en un segundo. Implemente también el método IncrementarMinuto para incrementar un minuto y el método IncrementarHora para incrementar una hora. Deben probarse los siguientes casos:
   a. Incrementar un segundo pasando al siguiente minuto.
   b. Incrementar un minuto pasando a la siguiente hora.
   c. Incrementar una hora pasando al siguiente día (por ejemplo, 11:59 p.m. a 12:00 a.m.).

6. Modificar la clase MiFecha de modo que detecte errores en los valores de los inicializadores para las variables de ejemplar (atributos) día, mes y año. También debe proporcionarse un método SiguienteDía para incrementar un día a la fecha guardada. Deben probarse los siguientes casos:
   a. Incrementar un día pasando al siguiente mes.
   b. Incrementar un mes pasando al siguiente año.

7. Crear una clase CuentaAhorro. Utilice una variable de clase static para almacenar la tasa de interés anual para cada uno de los ahorristas. Cada objeto de clase contiene una variable de ejemplar saldo de ahorros que indica la cantidad que el ahorrista tiene actualmente en depósito. Implemente el método que calcule el interés del mes multiplicando el saldo de ahorros por la tasa de interés anual dividida entre 12; estos intereses deberán sumarse al saldo. Incluir un método static que permita modificar el valor asignado a la tasa de interés anual.

8. Crear una clase MiFraccion para realizar aritmética de fracciones. Utilizar variables enteras para representar las variables de ejemplar privadas de la clase: el numerador y el denominador. Proporcionar un método constructor que permita inicializar un objeto de esta clase cuando se declara. El constructor deberá almacenar la fracción en forma reducida, es decir, la fracción 2/4 se almacenará como 1 en el numerador y 2 en el denominador. Incluir un constructor sin argumentos con valores por omisión en caso de que no se proporcionen valores. Incluir métodos públicos para cada una de las siguientes operaciones:
   a. Suma de dos números de la clase MiFraccion. El resultado de la suma debe almacenarse en forma reducida.
   b. Resta de dos números de la clase MiFraccion. El resultado de la resta debe almacenarse en forma reducida.
   c. Multiplicación de dos números de la clase MiFraccion. El resultado de la multiplicación debe almacenarse en forma reducida.
   d. División de dos números de la clase Difracción. El resultado de la división debe almacenarse en forma reducida.
   e. Reporte de las fracciones en la forma a/b, donde a es el numerador y b el denominador.
   f. Reporte de números de la clase Difracción en forma de punto flotante. (Considerar ofrecer capacidades de formateo que permita al usuario de la clase especificar el número de dígitos de precisión a la derecha del punto decimal.

9. Crear una clase ListaEnteros, la cual tenga como atributo un arreglo que puede contener hasta 20 números. La clase debe contener los siguientes métodos:
   a. Agregar un número entero.
   b. Eliminar un número, si ingresa su posición.
   c. Obtener un entero dada su posición.
   d. Dado un número entero cualquiera, verificar si se encuentra en el arreglo.
   e. Insertar un entero dada su posición.
   f. Listar los números ordenados en forma descendente.

10. De manera similar al problema anterior, genere una clase ListaNombres, la cual tenga como atributo un arreglo que puede contener hasta 20 números. En la clase se deben implementar los siguientes métodos:
   a. Agregar un nombre.
   b. Eliminar un nombre, si ingresa su posición.
   c. Obtener un nombre dada su posición.
   d. Dado un nombre cualquiera, verificar si se encuentra en el arreglo.
   e. Insertar un nombre dada su posición.
   f. Listar los nombres ordenados en forma ascendente.

11. Implementar una clase coleccionadora ListaTrabajadores. Reutilice la clase Trabajador, implementada en el ejercicio 1. La clase debe ser capaz de:
   a. Ingresar un trabajador dado.
   b. Eliminar un trabajador dada su posición.
   c. Obtener el trabajador dada su posición.
   d. Buscar si un trabajador existe dado su nombre.
   e. Reportar la relación de trabajadores ordenados en forma ascendente por el nombre.
   f. Reportar la relación de trabajadores ordenados en forma descendente por el sueldo neto.

12. Implementar una clase Lista de Movimientos de Productos de un Almacén que refleje entradas y salidas de productos de dicho almacén. De cada movimiento de producto se conoce el código del producto, la cantidad y el tipo de movimiento (0: entradas, 1: salidas). La clase debe ser capaz de:
   a. Ingresar un movimiento de un producto.
   b. Devolver el stock actual de un producto dado.
   c. Reportar la relación de productos con su respectivo stock.

d. Reportar la relación de productos por stock de mayor a menor.

13. Un hotel tiene 30 habitaciones. Se desea crear un programa que controle dichas habitaciones. Para ello debe implementarse una clase ListaHabitaciones. De cada habitación se conoce su número, disponibilidad (true: disponible; false: ocupada) y nombre del huésped. El programa debe permitir:
    a. Crear las habitaciones. No se permiten número de habitaciones repetidas.
    b. Ingresar un huésped en una habitación dada.
    c. Reportar la relación de habitaciones. En el caso de que la habitación esté ocupada se debe mostrar el nombre del huésped. Si está vacía, saldrá el mensaje de disponible.

    Implemente un menú de opciones para probar la clase ListaHabitaciones.

14. Crear una clase ListaAlumnos. De cada alumno se conoce su código, apellidos, nombres y la lista de los cursos matriculados que lleva. De cada curso matriculado se conoce el nombre del curso, número de créditos y promedio final. Hacer un programa con un menú que permita:
    a. Ingresar un alumno dado.
    b. Agregar un curso determinado a un alumno.
    c. Reportar la relación de alumnos con sus respectivos cursos.
    d. Dado un código de alumno reportar su relación de cursos y el total de créditos que lleva.