**Introducción a la Programación Orientada a Objetos**

**Trabajo Práctico 2**

1. Implementar una clase **Persona** con los atributos: *nombre, apellido, tipo y número de documento*.
    a) Defina en la clase los siguientes métodos:
        1. Método constructor _ _*construct()* que recibe como parámetros los valores iniciales para los atributos de la clase.
        2. Los métodos de acceso de cada uno de los atributos de la clase.
        3. Redefinir el método _ _*toString()* para que retorne la información de los atributos de la clase.
        4. Crear un script *Test_Persona* que cree un objeto Persona e invoque a cada uno de los métodos implementados.
    b) Realizar las modificaciones necesarias en la clase **CuentaBancaria** (Ejercicio 14 del TP1) para que en vez de contener como atributo el *DNI* del dueño de la cuenta tenga una referencia a las clase **Persona**.
2. Implementar una clase **Disquera** con los atributos: *hora_desde* y *hora_hasta* (que indican el horario de atención de la tienda), estado (abierta o cerrada), dirección y dueño. El atributo dueño debe referenciar a un objeto de la clase **Persona** implementada en el punto 1. Defina en la clase los siguientes métodos:
    a) Método constructor _ _*construct ()* que recibe como parámetros los valores iniciales para los atributos de la clase.
    b) Los métodos de acceso de cada uno de los atributos de la clase.
    c) *dentroHorarioAtencion($hora,$minutos)*: que dada una hora y minutos retorna *true* si la tienda debe encontrarse abierta en ese horario y *false* en caso contrario.
    d) *abrirDisquera($hora,$minutos):* que dada una hora y minutos corrobora que se encuentra dentro del horario de atención y cambia el estado de la disquera solo si es un horario válido para su apertura.
    e) *cerrarDisquera($hora,$minutos):* que dada una hora y minutos corrobora que se encuentra fuera del horario de atención y cambia el estado de la disquera solo si es un horario válido para su cierre.
    f) Redefinir el método _ _*toString()* para que retorne la información de los atributos de la clase.
    g) Crear un script *Test_Disquera* que cree un objeto Disquera e invoque a cada uno de los métodos implementados.
3. Realizar las modificaciones necesarias en la clase **Libro** (Ejercicio 16 del TP1) para que en vez de contener como atributos nombre y apellido del autor del libro, tenga una referencia a las clase **Persona**. Además agregue como variables instancias de la clase la cantidad de páginas y sinopsis del libro.
    a) Método constructor _ _*construct ()* que recibe como parámetros los valores iniciales para los atributos de la clase.
    b) Los métodos de acceso de cada uno de los atributos de la clase.
    c) Redefinir el método _ _*toString()* para que retorne la información de los atributos de la clase
    d) Crear un script *Test_Libro* que cree un objeto Libro e invoque a cada uno de los métodos implementados.
4. Se desea implementar una clase **Lectura** que almacena información sobre la lectura de un determinado libro. Esta clase tiene como variable instancia un referencia a un objeto Libro y el número de la página que se esta leyendo. Por otro lado la clase contiene los siguientes métodos:
    a) Método constructor _ _*construct()* que recibe como parámetros los valores iniciales para los atributos de la clase.
    b) Los métodos de acceso de cada uno de los atributos de la clase.
    c) *siguientePagina():* método que retorna la página del libro y actualiza la variable que contiene la pagina actual.
    d) *retrocederPagina():* método que retorna la página anterior a la actual del libro y actualiza su valor.
    e) *irPagina(x):* método que retorna la página actual y setea como página actual al valor recibido por parámetro.
    f) Redefinir el método _ _*toString()* para que retorne la información de los atributos de la clase.
    g) Crear un script *Test_Lectura* que cree un objeto Lectura e invoque a cada uno de los métodos implementados.
5. Realizar las modificaciones que crea necesaria en la clase implementada en el punto 4 para poder almacenar información de los libros que va leyendo una persona. Implementar los siguientes métodos:
    a) *libroLeido($titulo):* retorna true si el libro cuyo título recibido por parámetro se encuentra dentro del conjunto de libros leídos y false en caso contrario.

b) *darSinopsis($titulo):* retorna la sinopsis del libro cuyo título se recibe por parámetro.
c) *leidosAnioEdicion($x):* que retorne todos aquellos libros que fueron leídos y su año de edición es un año X recibido por parámetro.
d) *darLibrosPorAutor($nombreAutor):* retorna todos aquellos libros que fueron leídos y el nombre de su autor coincide con el recibido por parámetro.

6. En un banco existen varios mostradores. Cada mostrador puede atender cierto tipo de trámites y tiene una cola de clientes, que no puede superar un número determinado para cada cola, de cada cola se conoce el número actual de personas que hay en ella. Cada cliente concurre al banco para realizar un solo trámite. Un trámite tiene un horario de creación y un horario de resolución. Implemente los siguientes métodos:
   a) Método constructor _ _*construct()* que recibe como parámetros los valores iniciales para los atributos de las clases.
   b) Los métodos de acceso de cada uno de los atributos de las clases.
   c) Redefinir el método _ _*toString()* para que retorne la información de los atributos de las clases.
   d) *mostrador–>atiende($unTramite)*: devuelve true o false indicando si el tramite se puede atender o no en el mostrador; note que el tipo de trámite correspondiente a *unTramite* tiene que coincidir con alguno de los tipos de trámite que atiende el mostrador.
   e) *banco–>mostradoresQueAtienden($unTramite)*: retorna la colección de todos los mostradores que atienden ese trámite.
   f) *banco–>mejorMostradorPara($unTramite)*: retorna el mostrador con la cola más corta con espacio para al menos una persona más y que atienda ese trámite; si ningun mostrador tiene espacio, retorna null.
   g) *banco–>atender($unCliente)*: cuando llega un cliente al banco se lo ubica en el mostrador que atienda el trámite que el cliente requiere, que tenga espacio y la menor cantidad de clientes esperando; si no hay lugar en ningún mostrador debe retornar un mensaje que diga al cliente que "será antendido en cuanto haya lugar en un mostrador".

7. Realizar las modificaciones necesarias en la clase **Banco** para poder gestionar colas de trámites además de colas de clientes. De cada trámite se conoce el cliente que le da inicio. Asimismo del trámite también se conoce: la fecha de ingreso, la fecha de cierre, el horario de ingreso y el horario de cierre.

   Implementar los siguientes métodos:

   a) *ingresarTramite:* esta etapa es cuando la persona ya esta en el mostrador explicando el trámite para que sea tratado. Ya salió de la cola de trámites y está siendo atendido en el mostrador correspondiente.
   b) *cerrarTramite:* es cuando el trámite ha sido resuelto. Además, debe validar que el tramite a cerrar está abierto y setearlo en estado cerrado.
   c) *mostrador–> cantTramitesAtendidosMes():* el método retorna la cantidad promedio de trámites resueltos por día en este mes.
   d) *mostrador–> porcentajeTramitesResuelto()*: método que da el porcentaje de tramites resueltos sobre el total de recibidos.
   e) *tramite–>cantTramitesAbiertos():* método que retorna la cantidad de trámites abiertos de un cliente.
   f) *tramite–>cantTramitesCerrados():* método que retorna la cantidad de trámites cerrados de un cliente.
   g) *banco–>promTramitesIngresadosDia():* método que retorna el promedio de trámites ingresados por día.
   h) *banco–>promTramitesCerradosDia():* método que retorna el promedio de trámites cerrados por día.
   i) *banco–>mostradorResuelveMasTramites()*: método que retorna el mostrador con mayor porcentaje de tramites resueltos (sobre el total recibido) en el mes actual (o en un rango de fechas - pueden usar el tda fecha o clases de php, por ejemplo usar el getdate() de Php).

8. Modificar la clase **Teatro** (Ejercicio 15 TP 1) para que ahora las funciones sean un objeto que tenga las variables nombre, horario de inicio, duración de la obra y precio. El teatro ahora, contiene una referencia a una colección de funciones; las cuales pueden variar en cantidad y en horario. Volver a implementar las operaciones que permiten modificar el nombre y el precio de una función. Luego implementar la operación que carga las funciones en un teatro especifico, solicitando por consola la información de las mismas. También se debe verificar que el horario de las funciones, no se solapen para un mismo teatro.