Guía 8

1) Crea una clase "Coche" con las propiedades "marca", "modelo" y "precio".

Crea clase de servicio con los métodos "ingresarCoche" y "mostrarCoche".

2) Crea una clase "Estudiante" con las propiedades "nombre", "edad" y "nota media".

Crea un método en la clase de servicios para crear al estudiante, uno para mostrarlo y otro para mostrar por pantalla si el estudiante aprobó o no.

3) Crea una clase "Mascota" con las propiedades "nombre", "edad", "raza" y "tipo" (perro, gato, etc.).

Crea un constructor para inicializar los valores de las propiedades y en la clase servicio crear un método para agregar la mascota y otro para mostrar por pantalla la información de la mascota.

También crea un método "cumplirAnios" que aumente en uno la edad de la mascota.

4) Crea una clase "Ascensor" con las propiedades "pisoActual" y "maximoPisos".

Crea un constructor para inicializar los valores de las propiedades.

Crea un método "subir" que aumente en uno el valor de la propiedad "pisoActual" si aún no se ha alcanzado el "maximoPisos".

Crea un método "bajar" que disminuya en uno el valor de la propiedad "pisoActual" si no se ha alcanzado el piso "0".

Crea un método "irA" que reciba como parámetro el número de piso al que se desea ir y que modifique la propiedad "pisoActual" para que tenga ese valor.

Crea un programa que cree un objeto "Ascensor" con un "maximoPisos" de 10. Simula el comportamiento del ascensor moviéndolo hacia arriba y hacia abajo, yendo a un piso específico, y mostrando en pantalla el piso actual después de cada movimiento.

5) Crear la clase “Tarea” con los siguientes atributos título, descripción y estado (boolean). Agregar constructor vacío, con parámetros, getters y setters.

Luego crear la clase “TareaServicio”, que contenga:

* Método para crear una Tarea: se pedirá al usuario que ingrese el título y la descripción. Por defecto, al crear una tarea se tiene que asignar el valor de true al atributo **estado**, esto indicará que la tarea está en proceso.
* Método para modificar el estado a false, esto indicará que la tarea está terminada.
* Método para analizar si el estado de la tarea y que muestre si la misma está en proceso o terminada.
* Método para visualizar los datos de la Tarea.

6) Crear la clase “Producto” con los siguientes atributos: nombre, descripción, precio de venta y stock. Agregar constructores, getters y setters.

Luego crearemos la clase “ProductoServicio”, que contendrá los siguientes métodos:

* Método crearProducto(), solicitando los datos al usuario.
* Método comprarProducto(int cantidad, double precio), recibe por parámetros la cantidad de productos comprados y el precio unitario pagado. Debe modificar el stock de productos e imprimir por pantalla el monto total de la compra.
* Método unidadesDisponibles(), debe imprimir el stock del producto.
* Método venderProducto(int cantidad), recibe por parámetro las unidades que solicita el cliente. Debe descontar las unidades vendidas del stock. Tener en cuenta que debe analizar las unidades disponibles, y en el caso que las mismas sean inferiores a la cantidad solicitada por el cliente, debe notificar al cliente que no tienen stock suficiente y solamente venderle las unidades disponibles hasta agotar stock.
* Método visualizarProducto(), debe imprimir los datos del producto.

7) Crear la clase “Autobús” con los siguientes atributos: identificación, capacidad máxima de pasajeros, capacidad actual de pasajeros y cantidad de paradas. Luego agregar constructores, setters y getters.

Crear la clase “AutobúsServicio”, la misma contendrá los siguientes métodos:

* Método para crear un Autobús: se pedirá al usuario que ingrese los datos.
* Método inicioRecorrido(int cantidad), recibirá por parámetro la cantidad de personas que suben al autobús en el inicio del recorrido. La misma no debe superar la capacidad máxima del autobús.
* Método finRecorrido(), debe simular la finalización del recorrido y por lo cual el autobús debe quedar vacío.
* Método subirPasajeros(int cantidad, int parada), recibe por parámetro la cantidad de personas a subir y el número de parada.

En primer lugar debe analizar el número de parada, si la misma corresponde a la primera parada debe ejecutarse el método inicioRecorrido(int cantidad). En el caso que sea la última parada, debe informar al usuario que se encuentra en la finalización del recorrido y no pueden subir más pasajeros.

Si el número de parada es distinto al número de inicio y de finalización, debe subir a los pasajeros indicados, analizando la capacidad actual y verificando que no sobrepase la capacidad máxima. Solamente podrán subir pasajeros hasta cubrir la capacidad máxima.

* Método bajarPasajeros (int cantidad, int parada), recibe por parámetro la cantidad de personas a bajar del autobús y el número de paradas.

En primer lugar debe analizar el número de parada, si la misma corresponde a la primera parada, se debe informar al usuario que el autobús aún se encuentra vacío. En el caso que sea la última parada, se debe ejecutar el método finRecorrido().

Si el número de parada es distinto al número de inicio y de finalización, deben bajar los pasajeros indicados.