Nombre: juan camilo parra sanchez

```
    #-------------------------------------------------------------------------
-----------
    # Tarea 04 - 12/11/24
    # Para cada uno de los problemas que se plantean a continuación, escriba el
    # método que lo resuelve. No olvide identificar primero el patrón de
algoritmo que se
    # necesita y usar las guías que se dieron en secciones anteriores
    #-------------------------------------------------------------------------
-------------

    # Calcular el numero de sillas ejecutivas ocupadas en el avion
    def contarSillasEjecutivasOcupadas(self):
        totalOcupadas = 0
        for silla in self.sillas:
            if silla.darClase() == Silla.Clase.EJECUTIVA and
silla.sillaAsignada():
                totalOcupadas += 1
        return totalOcupadas

    # Localizar la silla en la que se encuentra el pasajero identificado con la
cédula que
    # se entrega como parámetro. Si no hay ningún pasajero en clase ejecutiva con
esa
    # cédula, el método retorna null
    def buscarPasajeroEjecutivo(self, cedula:str):
        for silla in self.sillas:
            if silla.darClase() == Silla.Clase.EJECUTIVA and
silla.sillaAsignada() and silla.darPasajero().darCedula() == cedula:
                return silla
        return None

    # Localizar una silla económica disponible, en una localización dada
(ventana,
    # centro o pasillo). Si no existe ninguna, el método retorna null :
    def buscarSillaEconomicaLibre (self, ubicacion: Silla.Ubicacion, pasajero:
Pasajero):
        for silla in self.sillas:
            if not silla.sillaAsignada() and silla.darClase() ==
silla.clase.ECONOMICA and silla.darUbicacion() == ubicacion:
                return silla
        return None
```

```
    # Asignar al pasajero que se recibe como parámetro una silla en clase
económica
    # que esté libre (en la ubicación pedida). Si el proceso tiene éxito, el
método retorna
    # verdadero. En caso contrario, retorna falso:
    def asignarSillaEconomica(self, ubicacion, pasajero):
        for silla in self.sillas:
            if not silla.sillaAsignada() and silla.darClase() ==
Silla.Clase.ECONOMICA and silla.darUbicacion() == ubicacion:
                silla.asignarPasajero(pasajero)
                return True
        return False

    # Anular la reserva en clase ejecutiva que tenía el pasajero con la cédula
dada.
    # Retorna verdadero si el proceso tiene éxito
    def anularReservaEjecutivo(self, cedula: str) :
         for silla in self.sillas:
             if silla.darClase() == silla.clase.EJECUTIVA and
silla.sillaAsignada() and silla.darPasajero().darCedula() == cedula:
                 silla.desasignarsilla
                 return True
             return False

    # Contar el número de puestos disponibles en una ventana, en la zona
económica
    # del avión:
    def contarVentanasEconomica(self):
        ventanasLibres = 0
        for silla in self.sillas:
            if silla.darClase() == Silla.Clase.ECONOMICA and not
silla.sillaAsignada() and silla.darUbicacion() == Silla.Ubicacion.VENTANA:
                ventanasLibres += 1
        return ventanasLibres

    # Informar si en la zona económica del avión hay dos personas que se llamen
igual.
    # Patrón de doble recorrido:
    def hayDosHomonimosEconomica(self):
        for i in range(len(self.sillas)):
            if self.sillas[i].sillaAsignada():
                nombreSilla = self.sillas[i].darPasajero().darNombre()
```

```
                for j in range(i + 1, len(self.sillas)):
                    if self.sillas[j].sillaAsignada():
                        nombreSilla2 = self.sillas[j].darPasajero().darNombre()
                        if nombreSilla == nombreSilla2:
                            return True
        return False
```