GRUPO 4:
BRUNO, LEO JOAQUÍN - FAI 3268
HERRERA, JEREMÍAS EZEQUIEL - FAI 3297
MONSERRAT VIDAL, MARIA ELVIRA - FAI 1829

5. Construir el TDA para tratar la abstracción de una línea área.
Para los aviones sus atributos son: identificador, modelo, cantidad de asientos, km recorridos, velocidad promedio.
Definir el diagrama UML, con Constructoras, Observadoras, Modificadoras y Propias del tipo.
Diseñar el UML Avion e implementarlo. Implementar un algoritmo de test.

<Avion>
<-> TEXTO identificador <-> TEXTO modelo <-> ENTERO cantidadDeAsientos <-> REAL kmRecorridos <-> REAL velocidadPromedio
Constructoras <Avion> (identificador, modelo, cantidadDeAsientos, kmRecorridos, velocidadPromedio) **Observadoras** <+> getIdentificador() : TEXTO <+> getModelo() : TEXTO <+> getCantidadDeAsientos() : ENTERO <+> getKmRecorridos() : REAL <+> getVelocidadPromedio() : REAL **Modificadoras** <+> setKmRecorridos **Propias** <+> toString() : TEXTO <+> equals(Avion objeto): LOGICO

8. Diseñar un algoritmo que almacene en una estructura adecuada la información de una cierta cantidad de aviones. El algoritmo debe realizar la carga de la estructura y permitir realizar las siguientes acciones:
• Mostrar el avión que tiene mayor velocidad (en caso de iguales devolver el último encontrado)
• Mostrar la cantidad de asientos que tiene un cierto avión (dado su identificador)
• Mostrar cuantos aviones de un cierto modelo tiene la empresa
• Mostrar el promedio de km recorrido por todos los aviones

Usamos el TDA del 5.

CLASE AVION.

```
public class Avion {

    //Atributos
    private String identificador;
    private String modelo;
    private int cantidadDeAsientos;
    private double kmRecorridos;
    private double velocidadPromedio;

    //Constructoras
    public Avion(String elIdentificador, String elModelo, int laCantidadDeAsientos, double
loskmRecorridos, double laVelocidadPromedio) {

        this.identificador = elIdentificador;
```

```
    this.modelo = elModelo;
    this.cantidadDeAsientos = laCantidadDeAsientos;
    this.kmRecorridos = loskmRecorridos;
    this.velocidadPromedio = laVelocidadPromedio;

}

//Contructor seteado con valores default.
public Avion(String elIdentificador) {
    this.identificador = elIdentificador;
    this.modelo = " ";
    this.cantidadDeAsientos = 0;
    this.kmRecorridos = 0;
    this.velocidadPromedio = 0;
}

//Observadores
public String getIdentificador() {
    return this.identificador;
}

public String getModelo() {
    return this.modelo;
}

public int getCantidadDeAsientos() {
    return this.cantidadDeAsientos;
}

public double getKmRecorridos() {
    return this.kmRecorridos;
}

public double getVelocidadPromedio() {
    return this.velocidadPromedio;
}
//Modificadores

public void setKmRecorridos(double losKmRecorridos) {
    this.kmRecorridos = losKmRecorridos;
}
   //Propias del tipo
   public boolean equals(Avion objeto){
         return (this.identificador.equalsIgnoreCase(objeto.getIdentificador()));
   }
   public static void toString(Avion objeto){
      System.out.println("Modelo: "+objeto.getModelo() +"\nIdentificador: "+ objeto.getIdentificador()
+ "\nCantidad de asientos: "+objeto.getCantidadDeAsientos() + "\nKm recorridos:
"+objeto.getKmRecorridos() + "\nVelocidad Promedio: "+objeto.getVelocidadPromedio());
   }


}
```

TEST AVION.

```
    public static void main(String args[]) {
        // Crea y carga un arreglo del tipo Avion (clase) y dado un menu muestra lo solicitado.
        Scanner sc = new Scanner(System.in);
        int cantAviones, posicionArreglo;
```

```
        String idAvion, modelo, respuesta;

        cantAviones = verificarLongitudArreglo();
        Avion aviones[] = new Avion[cantAviones];
        cargarArreglo(aviones);

        System.out.println("--------------------------------------------------------------------------\n");
        do{
        switch (menu()) {
            case 1:
                posicionArreglo = maxVelocidad(aviones);
                System.out.println("\nEl avion con velocidad promedio mayor es el avion " +
aviones[posicionArreglo].getModelo()
                        + " \ny la velocidad maxima promedio es de " +
aviones[posicionArreglo].getVelocidadPromedio() + " km/h.");
                break;
            case 2:
                System.out.println("Ingrese el identificador del avion: ");
                idAvion = sc.next();
                mostrarAsientos(idAvion, aviones);
                break;
            case 3:
                System.out.println("Ingrese el modelo de avion que desea consultar: ");
                modelo = sc.next();
                mostrarCantModelo(modelo, aviones);
                break;
            case 4:
                mostrarPromedioKm(aviones);
                break;
        }
            System.out.println("¿Desea realizar otra operacion? s/n");
            respuesta= sc.next();
            respuesta= respuesta.toLowerCase();
        }while(respuesta.equals("s"));
        System.out.println("PROGRAMA TERMINADO.");
    }

    public static int verificarLongitudArreglo() {
        //Verifica si el número ingresado por el usuario es un número válido para la longitud de
arreglo
        Scanner sc = new Scanner(System.in);
        int longitud;
        do {
            System.out.println("¿Cuantos aviones desea cargar?");
            longitud = sc.nextInt();
            if (longitud <= 0) {
                System.out.println("El numero ingresado es incorrecto. Por favor, intentelo nuevamente.
");
            }
        } while (longitud <= 0);

        return longitud;
    }

    public static void cargarArreglo(Avion[] arregloAvion) {
        //Carga un arreglo de objeto Avion. Se estima que todos los datos ingresados son correctos.
        Scanner sc = new Scanner(System.in);
        String elIdentificador, elModelo;
        int i, laCantidadDeAsientos;
        double losKmRecorridos, laVelocidadPromedio;
```

```
        for (i = 0; i < arregloAvion.length; i++) {
            System.out.println((i + 1) + "° avion: ");
            System.out.println("Ingrese el identificador: ");
            elIdentificador = sc.next();
            System.out.println("Ingrese el modelo del avion: ");
            elModelo = sc.n                                          ext();
            System.out.println("Ingrese la cantidad de asientos: ");
            laCantidadDeAsientos = sc.nextInt();
            System.out.println("Ingrese la cantidad de km recorridos: ");
            losKmRecorridos = sc.nextDouble();
            System.out.println("Ingrese la velocidad promedio: ");
            laVelocidadPromedio = sc.nextDouble();
            System.out.println("");
            arregloAvion[i] = new Avion(elIdentificador, elModelo, laCantidadDeAsientos,
losKmRecorridos, laVelocidadPromedio);
        }
    }

    public static int menu() {
        //Muestra un menu y retorna la opcion elegida por el usuario.
        Scanner sc = new Scanner(System.in);
        int opcion;
        System.out.println("""
                           Opcion 1 - Mostrar el avión que tiene mayor velocidad.
                           Opcion 2 - Mostrar la cantidad de asientos que tiene un cierto avión.
                           Opcion 3 - Mostrar cuantos aviones de un cierto modelo tiene la empresa.
                           Opcion 4 - Mostrar el promedio de km recorrido por todos los aviones.
                           Cualquier otro numero para terminar.
                           Ingrese la opción:""");
        opcion = sc.nextInt();
        return opcion;
    }

    public static int maxVelocidad(Avion arregloAvion[]) {
        //Compara la velocidad promedio de todos los aviones y devuelve el de mayor velocidad
promedio.
        double velMax = -1;
        int a, posicion = -1;
        for (a = arregloAvion.length - 1; a >= 0; a--) {
            if (velMax < arregloAvion[a].getVelocidadPromedio()) {
                velMax = arregloAvion[a].getVelocidadPromedio();
                posicion = a;
            }
        }
        return posicion;
    }

    public static void mostrarAsientos(String idAvion, Avion[] arregloAvion) {
        //
        int i = 0;
        boolean continuar = true;
        do {
            if (arregloAvion[i].getIdentificador().equals(idAvion)) {
                continuar = false;
                System.out.println("La cantidad de asientos del avion es: " +
arregloAvion[i].getCantidadDeAsientos());
            }
            i++;
        } while (continuar);
    }
```

```
public static void mostrarCantModelo(String modelo, Avion[] arregloAvion) {
    //
    int i, longArr, cantModelo = 0;
    longArr = arregloAvion.length;

    for (i = 0; i < longArr; i++) {
        if (arregloAvion[i].getModelo().equals(modelo)) {
            cantModelo++;
        }
    }
    System.out.println("La cantidad de aviones del modelo " + modelo + " es: " + cantModelo);
}

public static void mostrarPromedioKm(Avion[] arregloAvion) {
    //
    int i, longArr;
    double totalKm = 0, totalKmRecorridos;
    longArr = arregloAvion.length;
        for (i = 0; i < longArr; i++) {
            totalKm += arregloAvion[i].getKmRecorridos();
        }
        totalKmRecorridos = totalKm / longArr;
        System.out.println("El promedio de km recorridos por todos los aviones es: " +
totalKmRecorridos);
    }

}
```

11. Un lavadero de autos ofrece diferentes servicios a sus clientes:
a. Externa: limpieza externa, solo el chasis
b. Básica: limpieza externa e interna básica
c. Full: limpieza con motor.

Se debe diseñar un TDA para guardar los datos de los autos que están en el lavadero, de los cuales se almacena: **la patente, el nombre del propietario del auto, el teléfono del propietario, el servicio que ha contratado**. Definir los Constructoras, Observadoras, Modificadoras y Propias del tipo.

Implementar un algoritmo de test en el que se carguen 3 autos (en cada ejecución podemos cargar diferente información) y luego se muestre por pantalla los datos de los autos ordenados de la siguiente manera: primero listar los autos que han solicitado la limpieza Full, luego los que han contratado la limpieza Básica y por último la Externa. Si no hay en alguna de las clases de limpieza no se lista nada.

<Auto>
<-> TEXTO patente <-> TEXTO nombrePropietario <-> TEXTO telefono <-> TEXTO servicioContrato
Constructoras <Auto> (patente, nombrePropietario, telefono, sevicioContrato) **Observadoras** <+> getPatente() : TEXTO <+> getNombrePropietario() : TEXTO <+> getTelefono() : TEXTO <+> getServicioContrato() : TEXTO **Modificadoras**

CLASE AUTO.

```
public class Auto {

    //Atributos
    String patente;
    String nombrePropietario;
    String telefono;
    String servicioContrato;

    //Constructor
    public Auto(String laPatente, String elNombrePropietario, String elTelefono, String
elServicioContrato) {
        this.patente = laPatente;
        this.nombrePropietario = elNombrePropietario;
        this.telefono = elTelefono;
        this.servicioContrato = elServicioContrato;
    }

    public Auto(String laPatente) {
        this.patente = laPatente;
        this.nombrePropietario = " ";
        this.telefono = " ";
        this.servicioContrato = " ";
    }

    //Observadores
    public String getPatente() {
        return this.patente;
    }

    public String getNombrePropietario() {
        return this.nombrePropietario;
    }

    public String getTelefono() {
        return this.telefono;
    }

    public String getServicioContrato() {
        return this.servicioContrato;
    }

    //Modificadores
    public void setTelefono(String elTelefono){
        this.telefono=elTelefono;
    }

    public void setServicio(String newServicio){
        this.servicioContrato=newServicio;
    }

    //Propias del tipo
```

```
        public boolean igualA(Auto objeto){
            return (this.patente.equalsIgnoreCase(objeto.getPatente()));
        }

        public static void toString(Auto objeto){
            System.out.println("Nombre del propietario: "+objeto.getNombrePropietario() +"\nPatente: "+
objeto.getPatente() + "\nTelefono: "+objeto.getTelefono() + "\nServicio: "+objeto.getServicioContrato());
        }


        }

    }
```

TEST AUTO.

```
    public class TestAuto {
        public static void main(String[] args) {
            //
            Scanner sc = new Scanner(System.in);
            int cantAutos;

            cantAutos = verificarLongitudArreglo();
            Auto autos[] = new Auto[cantAutos];
            cargarArreglo(autos);
            serviciosContrato(autos, cantAutos);
        }

        public static int verificarLongitudArreglo() {
            //Verifica si el número ingresado por el usuario es un número válido para la longitud de
arreglo
            Scanner sc = new Scanner(System.in);
            int longitud;
            do {
                System.out.println("¿Cuantos autos desea cargar?");
                longitud = sc.nextInt();
                if (longitud <= 0) {
                    System.out.println("El numero ingresado es incorrecto. Por favor, intentelo
nuevamente. ");
                }
            } while (longitud <= 0);

            return longitud;
        }

        public static void cargarArreglo(Auto[] arregloAuto) {
            //Carga un arreglo de objeto Avion. Se estima que todos los datos ingresados son
correctos.
            Scanner sc = new Scanner(System.in);
            String laPatente, elPropietario, elTelefono, elServicio;
            int i;

            for (i = 0; i < arregloAuto.length; i++) {
                System.out.println((i + 1) + "° auto: ");
                System.out.println("Ingrese la patente: ");
                laPatente = sc.next();
                System.out.println("Ingrese el nombre del propietario: ");
                elPropietario = sc.next();
                System.out.println("Ingrese el telefono del propietario: ");
                elTelefono = sc.next();
                System.out.println("Ingrese el servicio contratado: ");
```

```
                elServicio = sc.next();
                elServicio = elServicio.toLowerCase();
                System.out.println("");
                arregloAuto[i] = new Auto(laPatente, elPropietario, elTelefono, elServicio);
            }
        }

    public static void serviciosContrato(Auto[] arregloAuto, int cantAutos) {
        //Crea arreglos unidimensionales de cada tipo de servicio y muestra por pantalla cada
auto segun el servicio.
        int i, j = 0, k = 0, l = 0, longArr;
        longArr = arregloAuto.length;

        Auto autosFull[] = new Auto[cantAutos];
        Auto autosBasica[] = new Auto[cantAutos];
        Auto[] autosExterno = new Auto[cantAutos];

        for (i = 0; i < longArr; i++) {
            switch (arregloAuto[i].getServicioContrato()) {
                case "full":
                    autosFull[j] = arregloAuto[i];
                    j++;
                    break;
                case "basico":
                    autosBasica[k] = arregloAuto[i];
                    k++;
                    break;
                case "externo":
                    autosExterno[l] = arregloAuto[i];
                    l++;
                    break;
            }
        }
        j = 0;
        k = 0;
        l = 0;
        System.out.println("Los Autos con servicio full son: ");
        while (autosFull[j] != null && j < cantAutos) {
            Auto.toString(autosFull[j]);
            j++;
        }
        System.out.println("");
        System.out.println("Los autos con servicio basica son:");
        while (autosBasica[k] != null && k < cantAutos) {
            Auto.toString(autosBasica[k]);
            k++;
        }
        System.out.println("");
        System.out.println("Los autos con servicio externo:");
        while (autosExterno[l] != null && l < cantAutos) {
            Auto.toString(autosExterno[l]);
            l++;
        }
    }

}
```