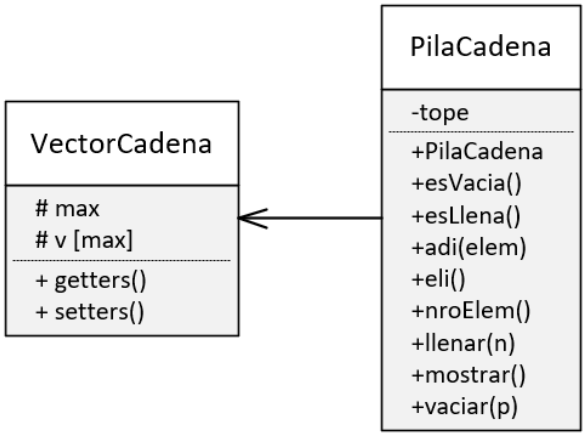
EJERCIOS DE PILA

EJERCICIO 1

UML



RESOLUCION

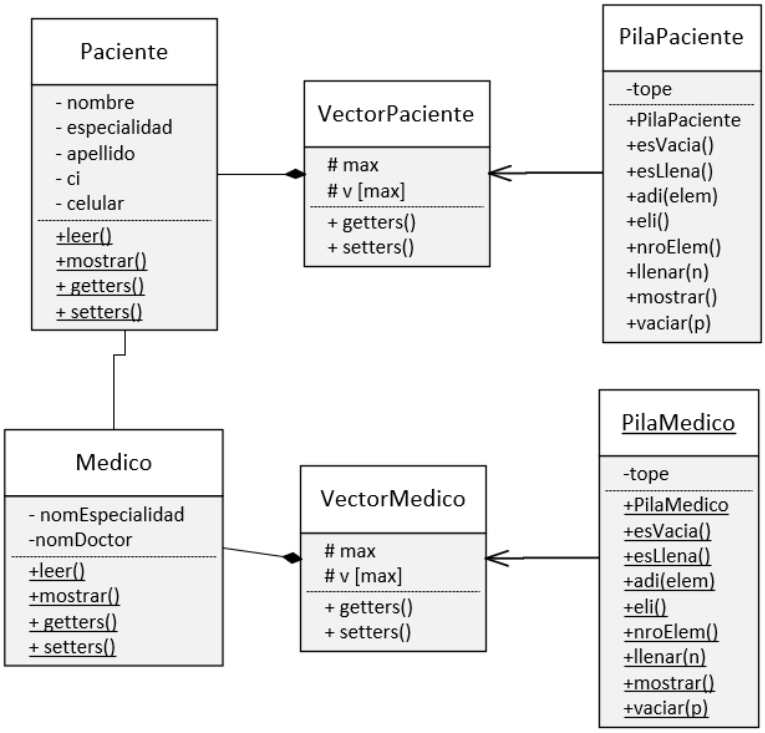
```
private static void notacion_polaca_inversa(PilaCadena a) {
    PilaCadena aux = new PilaCadena();
    PilaCadena aux1 = new PilaCadena();
    while (!a.esVacia()) {
        String elem = a.eli();
        elem=cambiarposicion(elem);
        aux.adi(elem);
    }
    a.vaciar(aux);
}
private static String cambiarposicion(String elem) {
    char [] car = elem.toCharArray();
    String numeros="",simbolos="",nudat="";
    for(char c:car) {
        if (c=='/'||c=='*'||c=='+'||c=='-') {
            String myString = Character.toString(c);
            simbolos+=myString;
        }else {
            String myString = Character.toString(c);
            numeros+=myString;
        }
    }
    nudat=numeros+simbolos;
    return nudat;
}
```

LLAMADO

```
public static void main(String[] args) {
    PilaCadena A = new PilaCadena();
    A.adi("2-8/9");
    A.adi("2+3*4");
    A.mostrar();
    System.out.println("****NOTACION POLACA INVERSA*******");
    notacion_polaca_inversa(A);
    A.mostrar();
}
```

EJERCICIO 3

UML



RESOLUCION

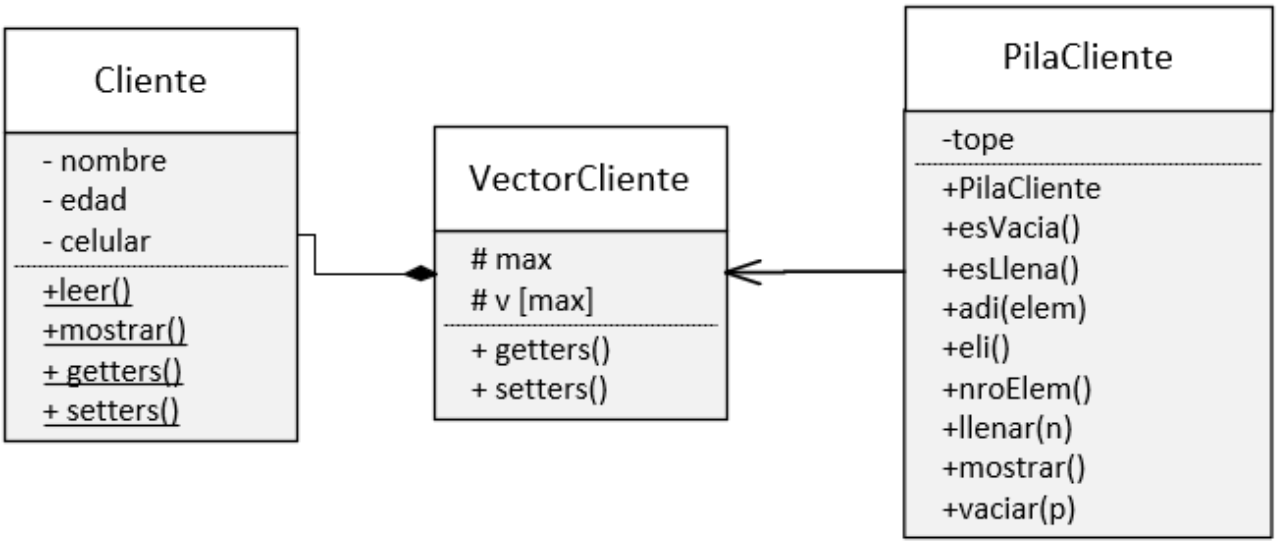
```
private static void incisoB(PilaPaciente a, PilaMedico b, String x, String y) {
    PilaPaciente aux = new PilaPaciente();
    while (!a.esVacia()) {
        Paciente elem = a.eli();
            if (elem.getNombre().equals(x)) {
                if (elem.getApellido().equals(y)) {
                    String espe = elem.getEspecialidad();
                    String atendido= doctor(espe,b);
                    System.out.println("El nombre del doctor que atendio a "
                            + ""+elem.getNombre()+" es "+atendido);
                }
            }
        aux.adi(elem);
    }
    a.vaciar(aux);
}

private static String doctor(String espe, PilaMedico b) {
    PilaMedico aux = new PilaMedico();
    String nombre = "";
    while (!b.esVacia()) {
        Medico m = b.eli();
        if (m.getNomEspecialidad().equals(espe)) {
            nombre=m.getNomDoctor();
        }
        aux.adi(m);
    }
    b.vaciar(aux);
    return nombre;
}

private static void incisoA(PilaPaciente a, String espx) {
    PilaPaciente aux = new PilaPaciente();
    while (!a.esVacia()) {
        Paciente elem = a.eli();
            if (elem.getEspecialidad().equals(espx)) {
                elem.mostrar();
            }
        aux.adi(elem);
    }
    a.vaciar(aux);
}
```

LLAMADO

```
PilaPaciente A = new PilaPaciente();
PilaMedico B = new PilaMedico();
A.llenar(5);
B.llenar(7);
System.out.println("\t*****PILA DE PACINTES******");
A.mostrar();
System.out.println("\t******PILA DE MEDICOS ******");
B.mostrar();
a. Mostrar todos los pacientes que fueron por la especialidad X
System.out.println("\t******INCISO A******");
System.out.println("Into especialidad X:");
Scanner sc = new Scanner(System.in);
//String espx=sc.next();
String espx="traumatología";
incisoA(A,espx);
System.out.println("\t******INCISO B******");
b. Añadir dos especialidades nuevas.
incisoB(B,2);
System.out.println("\t******AGREGADO******");
B.mostrar();
System.out.println("\t******INCISO C******");
c. Mostrar el nombre del doctor que atendió al paciente
de nombre X y apellido Y.
String x = "maria";
String y = "gomez";
incisoB(A,B,x,y);
```

EJERCICIO 5

UML



RESOLUCION

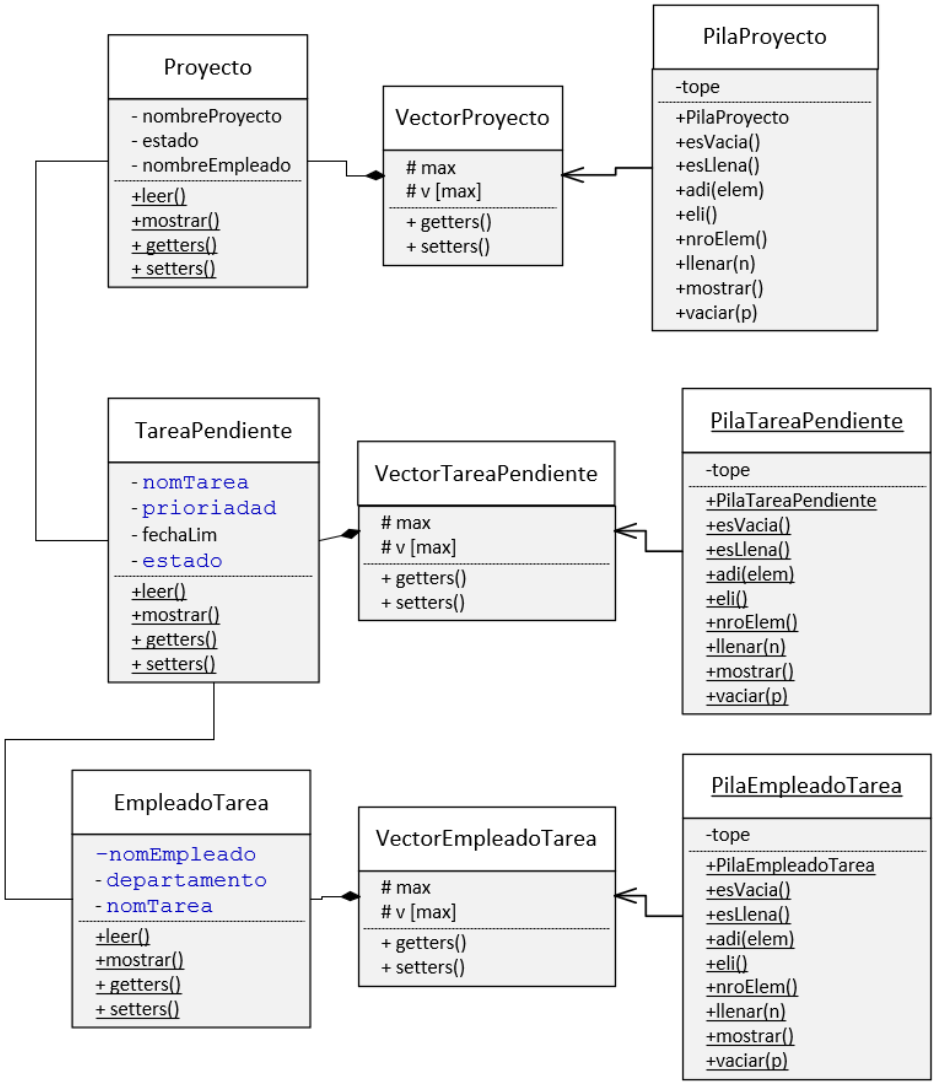
```
private static void inciso_c(PilaCliente c) {
    while (!c.esVacia()) {
        c.eli();
    }
}
private static void inciso_b(PilaCliente c, int i) {
    PilaCliente aux = new PilaCliente();
    while (!c.esVacia()) {
        Cliente cl = c.eli();
        int dig = primerDigitoClinte(cl);
        if (dig==i) {
            System.out.println("El clinte "+cl.getNombre()+" con el numero de celular: "+cl.getCelular());
        }
        aux.adi(cl);
    }
    c.vaciar(aux);
}
private static int primerDigitoClinte(Cliente cl) {
    String numeroStr =Integer.toString(cl.celular);
    char num = numeroStr.charAt(0);
    return Character.getNumericValue(num);
}

private static void inciso_a(PilaCliente c, int i, int j) {
    PilaCliente aux = new PilaCliente();
    PilaCliente aux2 = new PilaCliente();
    while (!c.esVacia()) {
        Cliente cl = c.eli();
        if (cl.edad>=i || cl.edad<=j) {
            aux.adi(cl);
        }else {
            aux2.adi(cl);
        }
    }
    c.vaciar(aux2);
    c.vaciar(aux);
}
```

LLAMADO

```
a. Una vez que ingresan todos a la panadería, serán reordenados y los primeros en
salir serán las personas mayores (>60 años) y niños (<12 años). Se pide
ordenarlos de esta manera.
System.out.println("\n\t*****INCISO A******");
inciso_a(c,60,12);
c.mostrar();
b. Mostrar a los clientes que su celular comience con el número 6.
System.out.println("\n\t*****INCISO B******");
inciso_b(c,6);
c. Vaciar la panadería, que ya no tenga clientes para atender.
System.out.println("\n\t*****INCISO C******");
System.out.println("-----PANADERIA CON CLIENTES---");
c.mostrar();
System.out.println("\n-----PANADERIA SIN CLIENTES---");
inciso_c(c);
c.mostrar();
System.out.println("Numero de clientes dentro de la panaderia: "+c.nroElem());
```

EJERCICIO 7

UML



RESOLUCION

```
private static void inciso_a(PilaProyecto z, String ne) {
    // TODO Auto-generated method stub
    PilaProyecto aux = new PilaProyecto();
    while(!z.esVacia()) {
        Proyecto est = z.eli();
        String x = est.getEstado();
        if(!(x.equals(ne))) {
            est.setEstado(ne);
        }
        aux.adi(est);
    }
    z.vaciar(aux);
}
```

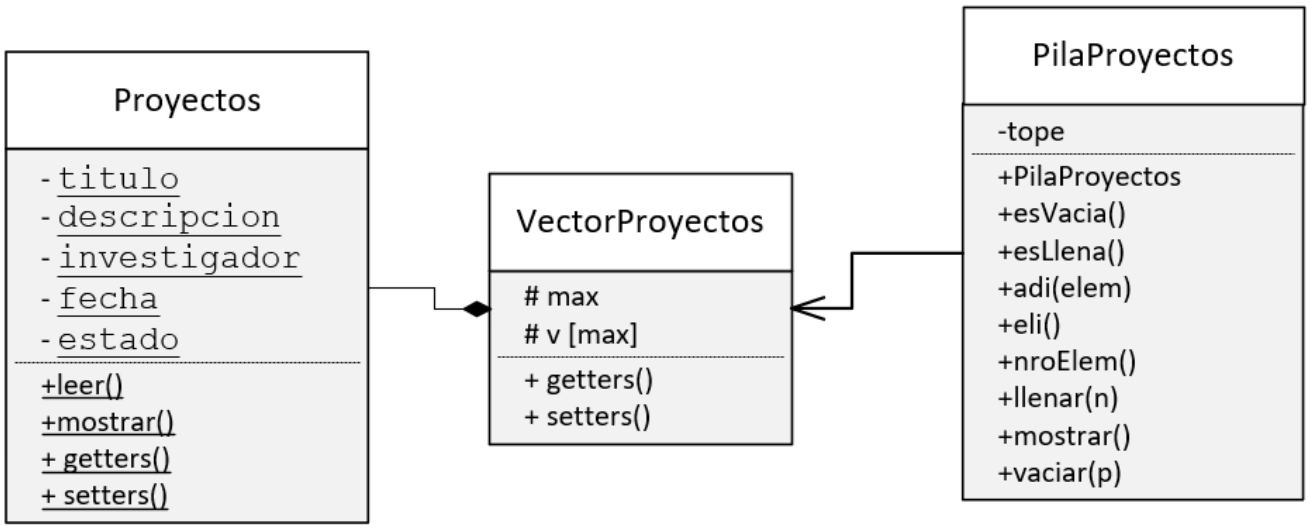
```
            private static void inciso_b(PilaTareaPendiente z) {
                // TODO Auto-generated method stub
                PilaTareaPendiente aux1 = new PilaTareaPendiente();
                PilaTareaPendiente aux2 = new PilaTareaPendiente();
                PilaTareaPendiente aux3 = new PilaTareaPendiente();
                String a ="Alta";
                String b = "Baja";
                while(!z.esVacia()) {
                    TareaPendiente elem = z.eli();
                    String x = elem.getPrioriadad();
                    if(x.equals(a)) {
                        aux1.adi(elem);
                    }
                    else {
                        if(x.equals(b)) {
                            aux2.adi(elem);
                        }
                        else {
                            aux3.adi(elem);
                        }
                    }
                }
                z.vaciar(aux2);
                z.vaciar(aux3);
                z.vaciar(aux1);
            }

    private static void inciso_c(PilaTareaPendiente a, PilaEmpleadoTarea b,String empleadoX) {
        // TODO Auto-generated method stub
        PilaTareaPendiente aux1 = new PilaTareaPendiente();
        PilaEmpleadoTarea aux2 = new PilaEmpleadoTarea();
        while(!b.esVacia()) {
            EmpleadoTarea x = b.eli();
            if(x.getNomEmpleado().equals(empleadoX)) {
                String nombreTarea = x.getNomTarea();
                while(!a.esVacia()) {
                    TareaPendiente y = a.eli();
                    if(y.getNomTarea().equals(nombreTarea)) {
                        String est = y.getEstado();
                        String flim = y.getFechaLim();
                        System.out.println("La tarea asignada a "+ empleadoX+" es --> "+ y.getNomTarea()+" con estado"
                                + " --> "+est+" Y fecha limite --> "+flim);
                        aux1.adi(y);
                    }
                    else {
                        aux1.adi(y);
                    }
                }
                aux2.adi(x);
                a.vaciar(aux1);
            }
            else {
                aux2.adi(x);
            }
        }
        b.vaciar(aux2);
    }
    private static void inciso_d(PilaEmpleadoTarea b, PilaProyecto c, String nomDesp) {
        // TODO Auto-generated method stub
        PilaEmpleadoTarea aux1 = new PilaEmpleadoTarea();
        PilaProyecto aux2 = new PilaProyecto();
        while(!b.esVacia()) {
            EmpleadoTarea x = b.eli();
            if(!(x.getNomEmpleado().equals(nomDesp))) {
                aux1.adi(x);
            }
        }
        b.vaciar(aux1);
        while(!c.esVacia()) {
            Proyecto y = c.eli();
            if(!(y.getnomEm().equals(nomDesp))) {
                aux2.adi(y);
            }
        }
        c.vaciar(aux2);

    }
```

LLAMADO

```
System.out.println("\n******* Inciso A ******");
Scanner sc = new Scanner(System.in);
System.out.println("\nIngresa nuevo estado: ");
String nuevoEstado = sc.nextLine();

inciso_a(C, nuevoEstado);
System.out.println("--- PILA DE PROYECTOS EN CURSO ---");
C.mostrar();

System.out.println("\n****** Inciso B ******");
inciso_b(A);
System.out.println("---- PILA DE TAREAS PENDIENTES ----");
A.mostrar();

System.out.println("\n****** Inciso C **********");
System.out.println("Ingresa el nombre de un empleado en especifico: ");
String empleadoX = sc.next();
inciso_c(A,B, empleadoX);


System.out.println("\n******* Inciso D ***********");
System.out.println("Ingresa el nombre el empleado que se va a despedir: ");
String nomDesp = sc.next();
inciso_d(B,C, nomDesp);
System.out.println("+++ PILAS ACTUALIZADAS +++");
System.out.println("\n--- PILA DE EMPLEADOS CON TAREAS ASIGNADAS -------");
B.mostrar();
System.out.println("\n------ PILA DE PROYECTOS EN CURSO -------");
C.mostrar();
```

EJERCICIO 9

UML



RESOLUCION

```
private static void inciso_a(PilaProyectos a, String invX) {
    PilaProyectos aux = new PilaProyectos();
    System.out.println("*******Los proyectos asigandos al investigador "+invX+" son los siguintes*****");
    boolean sw= false;
    while (!a.esVacia()) {
        Proyectos e = a.eli();
        if (e.getInvestigador().equals(invX)) {
            e.mostrar();
            sw=true;
        }else {
            sw=false;
        }
        aux.adi(e);
    }
    if (sw==false) {
        System.out.println("No existe el investigador.....");
    }
    a.vaciar(aux);
}
private static void inciso_b(PilaProyectos a, int x) {
    PilaProyectos aux = new PilaProyectos();
    for (int i = 1; i <= x; i++) {
        Proyectos e = new Proyectos();
        System.out.println("\t*****PROYECTO ["+i+"]****");
        e.leer();
        a.adi(e);
    }
}

 private static void inciso_c(PilaProyectos a, String añox, String mesIn, String mesFin) {
     PilaProyectos aux = new PilaProyectos();
     while (!a.esVacia()) {
         Proyectos e = a.eli();
         String año=e.getFecha();
         if(verificarFecha(año,añox,mesIn,mesFin)) {
             e.mostrar();
         }
         aux.adi(e);

     }
     a.vaciar(aux);
 }
 private static boolean verificarFecha(String añoP, String añox,String mesIn, String mesFin) {
     String[] partes = añoP.split("/");
     String mes = partes[0];
     String año = partes[2];
     int mesP = Integer.parseInt(mes);
     int mesin = Integer.parseInt(mesIn);
     int mesfin = Integer.parseInt(mesFin);
     boolean sw=false;
     if (año.equals(añox)) {
         for (int i = mesin; i <= mesfin; i++) {
             if (mesP==i) {
                 sw=true;
             }
         }
     }
     if (sw) {
         return true;
     }else {
         return false;
     }

 }
```

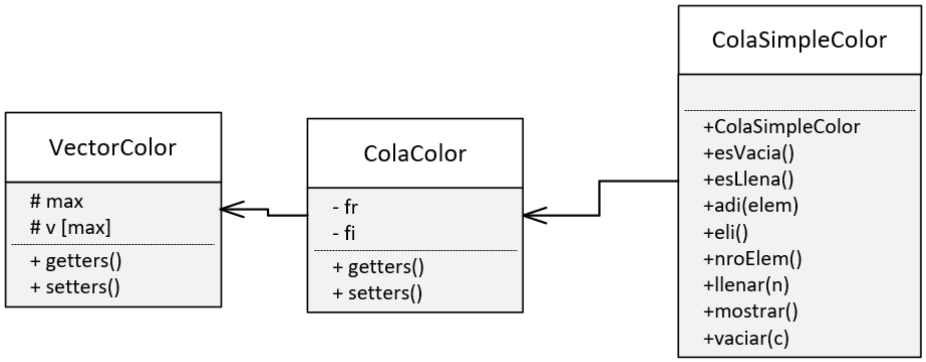
LLAMADO

```
public static void main(String[] args) {
    PilaProyectos A = new PilaProyectos();
    A.adi(new Proyectos("App móvil", "Gestión de tareas", "Juan", "10/04/2023", "En progreso"));
    A.adi(new Proyectos("App sofware", "Distribucion de ambintes", "Juan", "11/11/2023", "En progreso"));
    A.adi(new Proyectos("Inteligencia artificial", "Algoritmos de aprendizaje", "Ana", "20/07/2022", "Completado"));
    A.adi(new Proyectos("Inventario", "Gestión de inventario", "Carlos", "05/09/2022", "Pendiente"));
    A.adi(new Proyectos("Deportes", "Veneficios del deporte", "Carlos", "01/01/2020", "Pendiente"));
    A.adi(new Proyectos("Impacto ambiental", "Análisis de planta solar", "Elena", "15/11/2022", "Completado"));
    A.adi(new Proyectos("Falta de agua", "Análisis de lluvias", "Elena", "01/10/2021", "En revisión"));

    A.mostrar();
/       a. Mostrar todos los proyectos asignados a un investigador específico, incluyendo
/       el estado de cada proyecto y la fecha límite.
    Scanner sc = new Scanner(System.in);
    System.out.println("\t*****INCISO A *******");
    System.out.println("Introduce el nombre del investigador a busacar: ");
    String invX=sc.next();
    inciso_a(A,invX);
    b. Agregar nuevos proyectos a la pila de proyectos pendientes
    System.out.println("\t*****INCISO B *******");
    System.out.println("Ingrese la cantidad de nuevos proyectos a agregar :");
    int x = sc.nextInt();
    inciso_b(A,x);
    A.mostrar();
    c. Mostrar los proyectos del año x entre el intervalo de meses mi<mj

    System.out.println("\t*****INCISO B *******");
    System.out.println("Ingrese el año a buscar :");
    String añox = sc.next();
    System.out.println("Ingrese desde el mes incial: ");
    String mesInicial = sc.next();
    System.out.println("Ingrese el mes Final: ");
    String mesFinal = sc.next();
    inciso_c(A,añox,mesInicial,mesFinal);

}
```

EJERCIOS DE PILA

EJERCICIO 1

UML



RESOLUCION

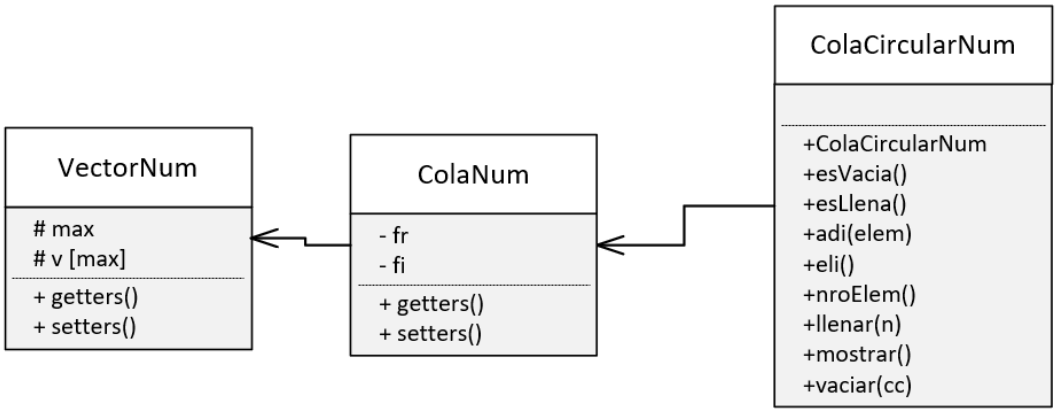
```
static void eliminarColorCentral(ColaSimpleColor Z) {
    ColaSimpleColor aux = new ColaSimpleColor();
    int nroElem = Z.nroElem();
    int cont = 0;
    if(nroElem % 2 == 0) {
        int m = nroElem/2;
        int k = nroElem/2 + 1;
        while(!Z.esVacia()) {
            String color = Z.eli();
            cont ++;
            if(cont!=m && cont!=k ) {
                aux.adi(color);
            }
        }
        Z.vaciar(aux);
    }
    else {
        int k = (int)(nroElem/2) + 1;
        while(!Z.esVacia()) {
            String color = Z.eli();
            cont ++;
            if(cont!=k ) {
                aux.adi(color);
            }
        }
        Z.vaciar(aux);
    }
}
```

LLAMADO

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    ColaSimpleColor A = new ColaSimpleColor();
    A.adi("Rojo");
    A.adi("Naranjado");
    A.adi("Amarillo");
    A.adi("Verde");
    A.adi("Azul");
    A.adi("Morado");
    A.adi("Turquesa");
    System.out.println("\n******* COLA SIMPLE DE COLORES ****");
    A.mostrar();
    System.out.println("\n----ELIMINAR LOS COLORES CENTRALES------");
    A.mostrar();
    System.out.println("NRO ELEM: "+A.nroElem());
    System.out.println("\n**** COLA SIMPLE DE COLORES CON LOS COLORES CENTRALES ELIMINADOS *****");
    eliminarColorCentral(A);
    A.mostrar();
    System.out.println("NRO ELEM: "+A.nroElem());

}
```

EJERCICIO 3

UML



RESOLUCION

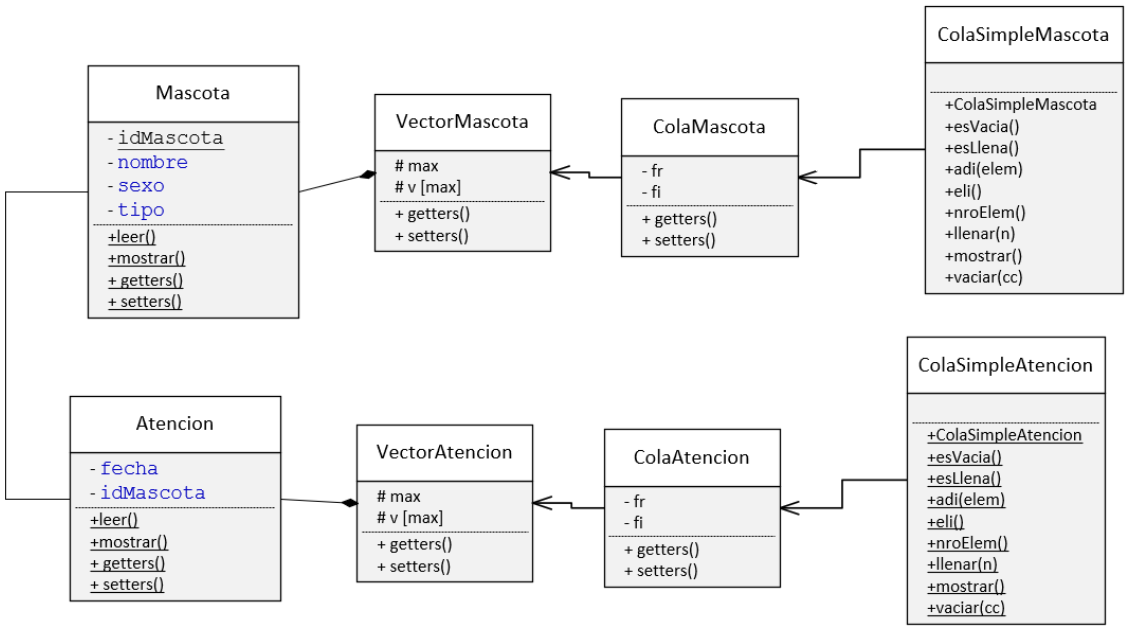
```
private static void intercambio_de_numeros(ColaCircularNum a, ColaCircularNum b, int i, int j) {
    int nroA = a.nroElem(),eleA=0;
    int nroB = b.nroElem(),eleB=0;
    for (int k = 1; k <= nroA; k++) {
        int elem = a.eli();
        if (k==i) {
            eleA=elem;
            System.out.println("este es el ["+k+"] elemento =>"+eleA);
        }
        a.adi(elem);
    }
    for (int k = 1; k <= nroB; k++) {
        int elem = b.eli();
        if (k==j) {
            eleB=elem;
            System.out.println("este es el ["+k+"] elemento =>"+eleB);
            elem=eleA;
        }
        b.adi(elem);
    }
    for (int k = 1; k <= nroA; k++) {
        int elem = a.eli();
        if (k==i) {
            elem=eleB;
        }
        a.adi(elem);
    }


}
```

LLAMADO

```
Dada una cola circular de números A y otra cola circular de números B, intercambiar el
i-esimo elemento de A con el j-esimo de B sin usar estructuras auxiliares.
System.out.println("\nIntro el nuemoro i a intercambiar: ");
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
System.out.println("Intro el numeor j a intercambiar: ");
int j = sc.nextInt();
intercambio_de_numeros(A,B,i,j);
System.out.println("\tINTERCAMBIADO LOS ELEMNTOS i y j DE LA COLA");
System.out.println("\t******COLA CIRCULAR A******");
A.mostrar();
System.out.println("\n\t******COLA CIRCULAR B******");
B.mostrar();
```

EJERCICIO 5

UML



RESOLUCION

```
private static int contarMascotasHembras(ColaSimpleMascota b, String sexo) {
    int cont=0;
    ColaSimpleMascota aux = new ColaSimpleMascota();
    while (!b.esVacia()) {
        Mascota m = b.eli();
        if (m.getSexo().equals(sexo)) {
            cont++;
        }
        aux.adi(m);
    }
    b.vaciar(aux);
    return cont;
}

private static void eliminarMascotaX(ColaSimpleMascota b, int id, String nombre) {
    ColaSimpleMascota aux = new ColaSimpleMascota();
    while (!b.esVacia()) {
        Mascota m = b.eli();
        if (m.getIdMascota()==id) {
            if (m.getNombre().equals(nombre)) {
                System.out.println("Se eliminara la Macota: "+m.getNombre()+" con la ID: "+m.getIdMascota());
            }

        }
        else {
            aux.adi(m);
        }
    }
    b.vaciar(aux);

}

  private static void mascotasAtendidasfechaX(ColaSimpleAtencion a, ColaSimpleMascota b, String fecha) {
      ColaSimpleAtencion aux = new ColaSimpleAtencion();
      while (!a.esVacia()) {
          Atencion at = a.eli();
          if (at.getFecha().equals(fecha)) {
              bucarId(at.getIdMascota(),b);
          }
          aux.adi(at);
      }
      a.vaciar(aux);

  }
```

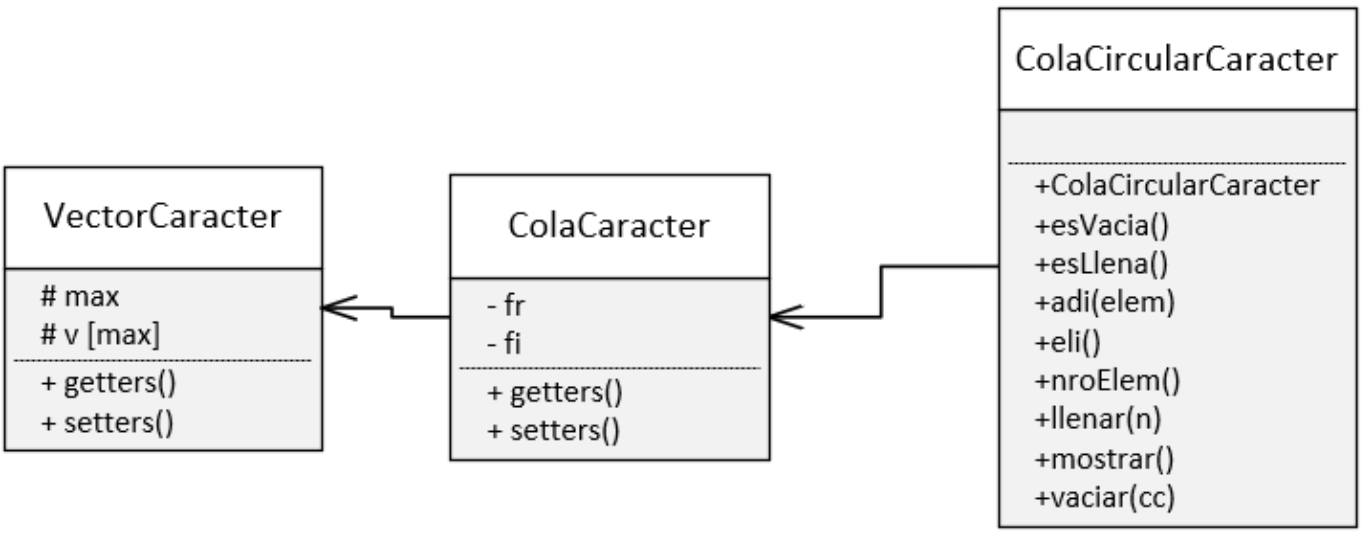
```
private static void bucarId(int idMascota, ColaSimpleMascota b) {
    ColaSimpleMascota aux = new ColaSimpleMascota();
    while (!b.esVacia()) {
        Mascota a = b.eli();
        if (a.getIdMascota()== idMascota) {
            a.mostrar();
        }
        aux.adi(a);
    }
    b.vaciar(aux);
}
```

LLAMADO

```
a. Mostrar el nombre de las mascotas que fueron atendidas
en la clínica veterinaria
en la fecha X
System.out.println("\n\t****INCISO A****");
Scanner sc = new Scanner(System.in);
System.out.println("intro la fecha X: ");
//String fecha =sc.next();
String fecha = "25-feb";
mascotasAtendidasfechaX(A,B,fecha);
b. Eliminar la mascota con el idMascota X si y sólo si su
nombre es "Tavo".
System.out.println("\n\t****INCISO B****");
System.out.println("intro ID-Mascota X: ");
//int id=sc.nextInt();
int id=6789;
eliminarMascotaX(B,id,"tavo");
B.mostrar();
c. Contar mascotas "hembra" que fueron atendidas en la
clínica veterinaria.
System.out.println("\n\t****INCISO C****");
String sexo="hembra";
int cant = contarMascotasHembras(B,sexo);
System.out.println("La cantidad de macotsa "+sexo+" que fueron atendidas son: "+cant);
```

EJERCICIO 7

UML



RESOLUCION

```
private static void InvertirLaCola(ColaCircularCaracter a) {
    if (!a.esVacia()) {
        char primero = a.eli();
        InvertirLaCola(a);
        a.adi(primero);
    }
}
private static void existetodasVocales(ColaCircularCaracter c) {
    ColaCircularCaracter aux = new ColaCircularCaracter();
    int cont=0;
    boolean a=true,e=true,i=true,o=true,u=true;
    while (!c.esVacia()) {
        char car = c.eli();
        if (car=='a' && a==true ) {
            cont++;
            a=false;
        }
        if (car=='e' && e==true ) {
            cont++;
            e=false;
        }
        if (car=='i' && i==true ) {
            cont++;
            i=false;
        }
        if (car=='o' && o==true ) {
            cont++;
            o=false;
        }
        if (car=='u' && u==true ) {
            cont++;
            u=false;
        }
        aux.adi(car);
    }
    c.vaciar(aux);
    if (cont==5) {
        System.out.println("La Cola contine todas las vocales");
    }else {
        System.out.println("La cola no contine todas las vocales ");
    }

}
```

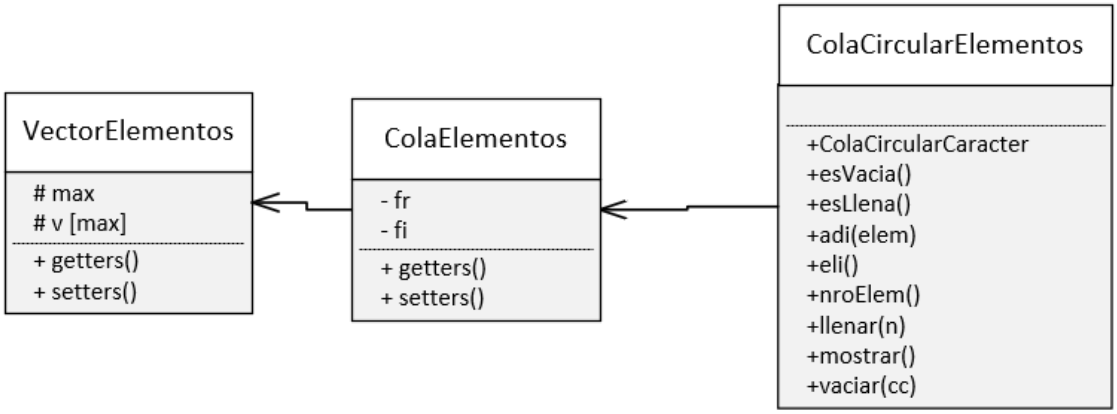
LLAMADO

```
public static void main(String[] args) {
    ColaCircularCaracter A = new ColaCircularCaracter();
    A.adi('j');
    A.adi('a');
    A.adi('b');
    A.adi('e');
    A.adi('i');
    A.adi('k');
    A.adi('l');
    A.adi('p');
    A.adi('o');
    A.adi('q');
    A.adi('u');
    A.adi('u');
    A.adi('j');
```

```
        System.out.println("\t****COLA CON CARACTERES****");
        A.mostrar();
        a. Ver si existen en la cola todas las vocales.
        System.out.println("\n\t*****INCIOS A*****");
        existetodasVocales(A);
        b. Invertir el orden de la cola sin el uso de
        estructuras auxiliares.
        System.out.println("\n\t*****INCIOS B*****");
        System.out.println("\t----COLA SIN INVERTIR----");
        A.mostrar();
        InvertirLaCola(A);
        System.out.println("\n\t----COLA INVERTIDA----");
        A.mostrar();


    }
```

EJERCICIO 9

UML



RESOLUCION

```
private static void inntroducir_datos_iesima(ColaCircularElementos a, int n, int iesimo)
        Scanner sc = new Scanner(System.in);
        int nro = a.nroElem();
        for (int i = 0; i < nro; i++) {
            String elem = a.eli();
            System.out.println(elem+" "+i);
            String e;
            if (i == iesimo) {
                for (int j = 0; j < n; j++) {
                    System.out.println("Ingrese para adicionar un elemento");
                    e = sc.nextLine();
                    a.adi(e);
                }
            }else {
                a.adi(elem);
            }
        }
    }
```

LLAMADO

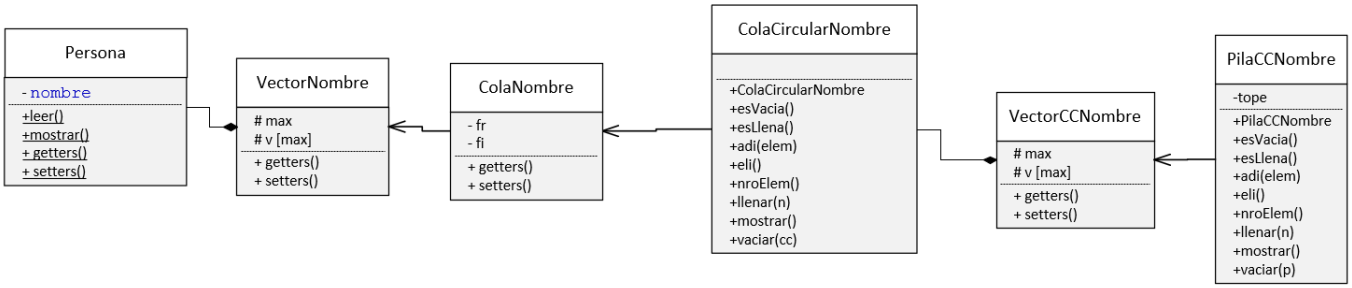
```
public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        ColaCircularElementos A= new ColaCircularElementos();
        int a,b,iesimo,j,n,m,k;
        System.out.println("Ingrese un numero de elementos");
        a=sc.nextInt();
        A.llenar(a);
        A.adi("1");
        A.adi("2");
        A.adi("3");
        A.adi("4");
        A.adi("5");
        A.adi("6");
        System.out.println("Elementos Ingresados");
        A.mostrar();
        System.out.println("\nIngrese los N elementos que quiere ingresar:");
        n=sc.nextInt();
        System.out.println("Antes de que i-esimo quiere ingresarlos:");
        iesimo=sc.nextInt();

        inntroducir_datos_iesima(A,n,iesimo);
        A.mostrar();

    }
```

```
Ingrese un numero de elementos
Elementos Ingresados
1 2 3 4 5 6
Ingrese los N elementos que quiere ingresar:
3
Antes de que i-esimo quiere ingresarlos:
2
Ingrese para adicionar un elemento
9
Ingrese para adicionar un elemento
8
Ingrese para adicionar un elemento
7
1 2 9 8 7 4 5 6
```

EJERCIOS DE MÚLTIPLES PILAS/COLAS

EJERCICIO 1

UML



RESOLUCION

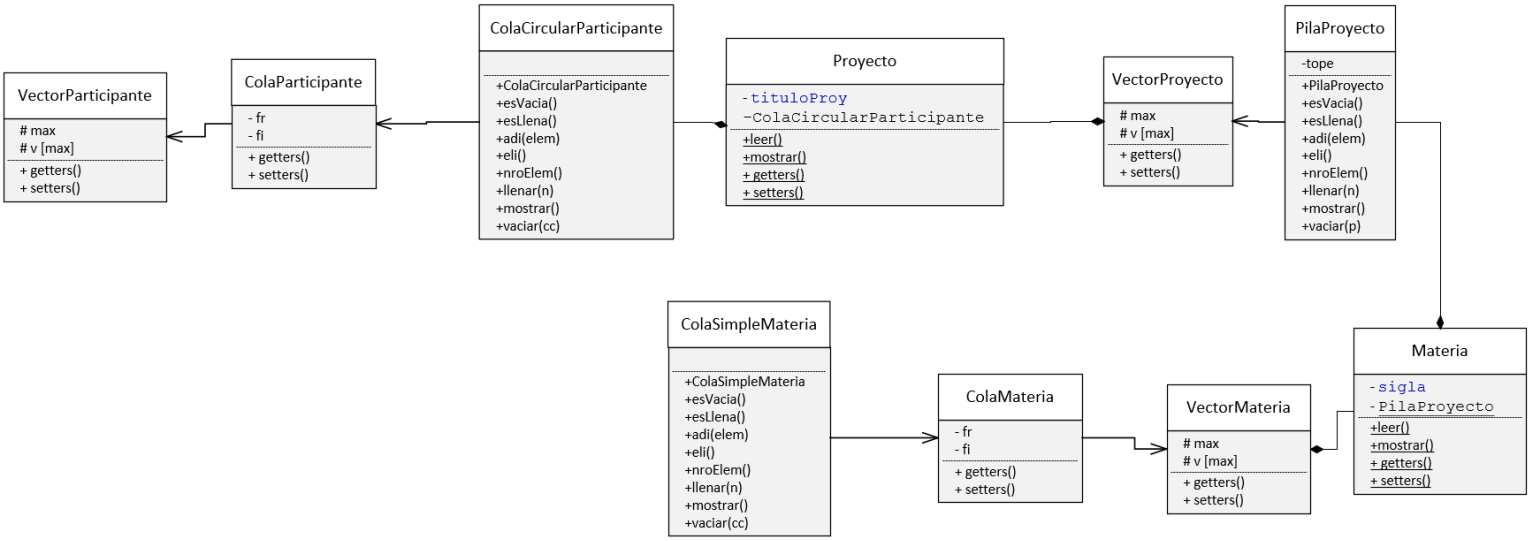
```
private static void inciso_b(Pila p) {
    Pila aux1 = new Pila();
    Pila aux2 = new Pila();
    while (!p.esVacia()) {
        ColaCircularNombre elem = (ColaCircularNombre) p.eleminar();
        aux1.adicionar(elem);
    }
    while (!aux1.esVacia()) {
        int minNroElem = Integer.MAX_VALUE;
        ColaCircularNombre minCola = null;
        while (!aux1.esVacia()) {
            ColaCircularNombre elem = (ColaCircularNombre) aux1.eleminar();
            int nroElem = elem.nroElem();
            if (nroElem < minNroElem) {
                minNroElem = nroElem;
                minCola = elem;
            }
            aux2.adicionar(elem);
        }
        while (!aux2.esVacia()) {
            ColaCircularNombre elem = (ColaCircularNombre) aux2.eleminar();
            if (elem.nroElem() == minNroElem) {
                p.adicionar(elem);
            } else {
                aux1.adicionar(elem);
            }
        }
    }
}

private static void inciso_a(Pila p) {
    Pila aux = new Pila();
    while (!p.esVacia()) {
        ColaCircularNombre cola = (ColaCircularNombre) p.eleminar();
        int nroE = cola.nroElem();
        String[] elementos = new String[nroE];
        for (int i = 0; i < nroE; i++) {
            elementos[i] = cola.eli().getNombre();
        }
        cola.vaciar(cola);
        for (int i = nroE - 1; i >= 0; i--) {
            cola.adi(new Persona(elementos[i]));
        }
        aux.adicionar(cola);
    }
    while (!aux.esVacia()) {
        p.adicionar((ColaCircularNombre) aux.eleminar());
    }
}
```

LLAMADO

```
Pila P = new Pila();

P.adicionar(B);
P.adicionar(A);
P.adicionar(D);
P.adicionar(C);
P.mostrar();
a. Invertir cada cola sin usar estructuras auxiliares
System.out.println("\t****INCISO A*****");
inciso_a(P);
P.mostrar();
b. Ordenar la pila según el número de elementos de
las colas, ascendentemente.
System.out.println("\t****INCISO B*****");
inciso_b(P);
P.mostrar();
```

EJERCICIO 3

UML



RESOLUCION

```
static void cantidadProyectoMateria(ColaSimpleMateria Z) {
    ColaSimpleMateria auxC = new ColaSimpleMateria();
    PilaProyecto auxP = new PilaProyecto();
    while(!Z.esVacia()) {
        int c = 0;
        Materia mat = Z.eli();
        PilaProyecto A = mat.getListaProyecto();
        while(!A.esVacia()) {
            Proyecto proyX = A.eli();
            c++;
            auxP.adi(proyX);
        }
        System.out.println("La materia con sigla ["+mat.getSigla()+"] tiene "
                + ""+c+" proyecto(s)");
        A.vaciar(auxP);
        auxC.adi(mat);
    }
    Z.vaciar(auxC);

}
```

```java
static void mostrarMateriasIgual_3Participantes(ColaSimpleMateria Z) {
    ColaSimpleMateria auxC = new ColaSimpleMateria();
    PilaProyecto auxP = new PilaProyecto();
    boolean sw = false;
    while(!Z.esVacia()) {
        Materia mat = Z.eli();
        PilaProyecto B = mat.getListaProyecto();
        while(!B.esVacia()) {
            Proyecto proyX = B.eli();
            ColaCircularParticipante cp = proyX.getListaParticipante();
            int nroPart = cp.nroElem();
            if(nroPart == 3) {
                System.out.println("Materia [Sigla= "+mat.getSigla()+", "
                        + "TituloProyecto= "+proyX.getTituloProy()+"]");
                sw  = true;
            }
            auxP.adi(proyX);
        }
        B.vaciar(auxP);
        auxC.adi(mat);
    }
    if(sw==false) {
        System.out.println("No hay pryectos con solo 3 participantes");
    }
    Z.vaciar(auxC);
}
```

```java
static void mostrarSiglaMateriasMenosParticipantes(ColaSimpleMateria Z) {
    int n = Z.nroElem();
    PilaProyecto aux = new PilaProyecto();
    int menor = buscarMenorParticipante(Z);
    System.out.println("Siglas de las materias con menos participantes: ");
    if(menor != -1) {
        for (int i = 0; i < n; i++) {
            int contP = 0;
            Materia mat = Z.eli();
            PilaProyecto C = mat.getListaProyecto();
            while(!C.esVacia()) {
                Proyecto proyX = C.eli();
                ColaCircularParticipante cp = proyX.getListaParticipante();
                int nroPart = cp.nroElem();
                contP = contP + nroPart;
                aux.adi(proyX);
            }
            C.vaciar(aux);
            if(contP == menor) {
                System.out.println(mat.getSigla());
            }
            Z.adi(mat);
        }
    }
    else {
        System.out.println("Las materias tienen la misma cantidad de participantes.");
    }
}
```

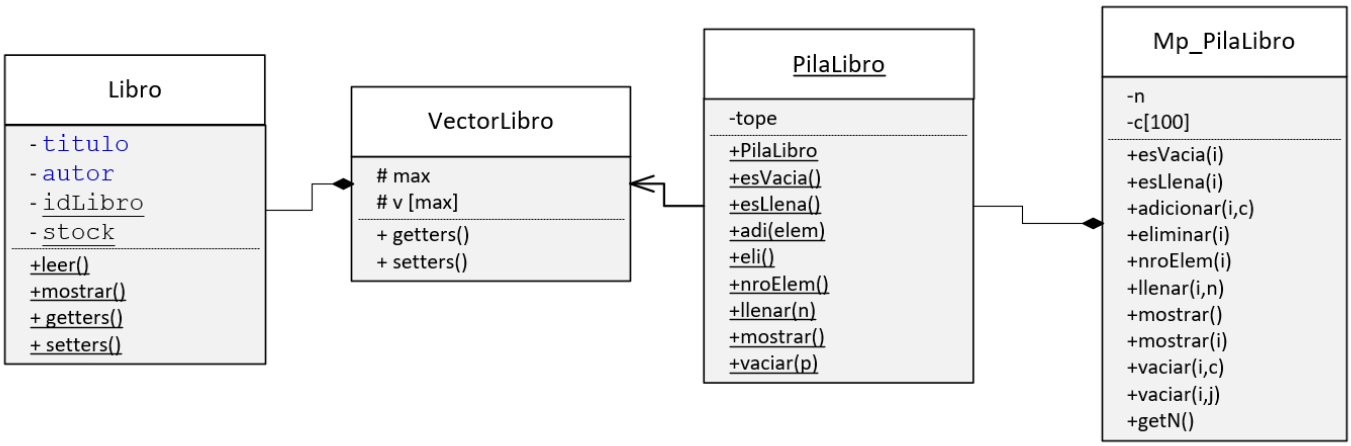
```
static int buscarMenorParticipante(ColaSimpleMateria Z) {
    int n = Z.nroElem();
    int k = 0;
    PilaProyecto aux = new PilaProyecto();
    int men = Integer.MAX_VALUE;
    for (int i = 0; i < n; i++) {
        int cont = 0;
        Materia mat = Z.eli();
        PilaProyecto C = mat.getListaProyecto();
        while(!C.esVacia()) {
            Proyecto proyX = C.eli();
            ColaCircularParticipante cp = proyX.getListaParticipante();
            int nroP = cp.nroElem();
            cont = cont + nroP;
            aux.adi(proyX);
        }
        C.vaciar(aux);
        if(cont < men) {
            men = cont;
            k++;
        }
        Z.adi(mat);
    }
    if(k == 0) {
        men = -1;
        return men;
    }
    return men;
}
```

LLAMADO

```
ColaSimpleMateria CM = new ColaSimpleMateria();
CM.adi(M1);
CM.adi(M2);
CM.adi(M3);
CM.adi(M4);
System.out.println("\n**** COLA SIMPLE DE MATERIAS **");
System.out.println("");
CM.mostrar();

System.out.println("\n---- INCISO (A) -----");
cantidadProyectoMateria(CM);
System.out.println("\n---- INCISO (B) -----");
mostrarMateriasIgual_3Participantes(CM);
System.out.println("\n---- INCISO (c) -----");
mostrarSiglaMateriasMenosParticipantes(CM);
```

EJERCICIO 5

UML



RESOLUCION

```
private static void numero_de_Pila_TituloX(Mp_PilaLibro a, String tituloX, String autor) {
    int n= a.getNp();
    Libro e;
    for (int i = 0; i < n ; i++) {
        PilaLibro aux = new PilaLibro();
        while (!a.esvacia(i)) {
            e = a.eliminar(i);
            if (e.getTitulo().equals(tituloX)) {
                if (e.getAutor().equals(autor)) {
                    System.out.println("El libro "+e.getTitulo()+" del autor "+e.getAutor()+
                            " está en la pila [" + i + "]");
                }
            }
            aux.adi(e);
        }
        a.vaciar(i, aux);
    }
}
private static void mostrarAutorX(Mp_PilaLibro a, String autor) {
    int n = a.getNp();
    Libro e;
    for (int i = 0; i < n; i++) {
        PilaLibro aux = new PilaLibro();
        boolean sw = false;
        while (!a.esvacia(i)) {
            e = a.eliminar(i);
            if (e.getAutor().equals(autor)) {
                System.out.println(autor + " está en la pila [" + i + "]");
                sw = true;
            }
            aux.adi(e);
        }
        if (sw) {
            System.out.println("\t*****Datos de la pila [" + i + "]*****");
            aux.mostrar();
        }
        a.vaciar(i, aux);
    }
}
```

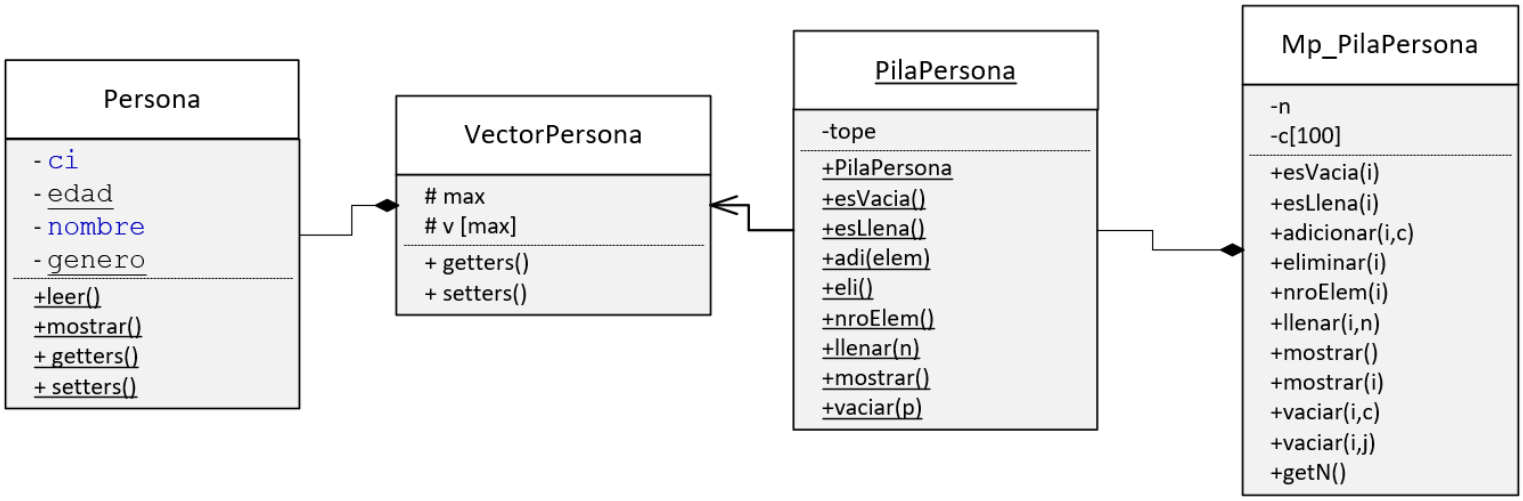
```java
private static void eliminarStockIgualCero(Mp_PilaLibro a, int stockCero) {
    int n = a.getNp();
    Libro e = new Libro();
    for (int i = 0; i < n; i++) {
        PilaLibro aux = new PilaLibro();
        while (!a.esvacia(i)) {
            e = a.eliminar(i);
            if (e.getStock()==stockCero) {
                System.out.println("El libro '"+e.getTitulo()+"' sera eliminado por Stock: "+stockCero);
            }
            else {
                aux.adi(e);
            }
        }
        a.vaciar(i, aux);
    }
}
```

LLAMADO

```java
  a. Eliminar de la MultiPila todos los libros con stock igual a 0.
System.out.println("\n\t******INCISO A******");
eliminarStockIgualCero(A,0);
A.mostrar();
  b. Mostrar el contenido de todas las pilas en las que existan al menos un libro del
  autor X.
System.out.println("\n\t******INCISO B******");
System.out.println("MOSTRAR LIBRO DEL AUTOR X: ");
//String autor=sc.next();
String autor="Jose";
mostrarAutorX(A,autor);
  c. Mostrar el número de la pila en la que esté el libro de título X del autor Y
  (asumiendo que solo puede existir el libro solicitado en una sola pila).
System.out.println("\n\t******INCISO C******");
System.out.println("MOSTRAR EL LIBRO DE TITULO X: ");
//String tituloX = sc.next();
String tituloX = "El señor de los anillos";
numero_de_Pila_TituloX(A,tituloX,autor);
```

EJERCICIO 7

UML



RESOLUCION

```
private static void inciso_a(Mp_PilaPersona a) {
    int n = a.getNp();
    Persona elem;
    for (int i = 0; i < n; i++) {
        PilaPersona aux = new PilaPersona();
        int contMasculino=0,contFemenino=0;
        while (!a.esvacia(i)) {
            elem = a.eliminar(i);
            if (elem.getGenero().equals("masculino")) {
                contMasculino++;
            }else {
                contFemenino++;
            }
            aux.adi(elem);
        }
        System.out.println("En la pila ["+i+"] la cantidad de Varones ["+contMasculino+"] "
                + "de Mujeres ["+contFemenino+"]");
        a.vaciar(i, aux);
    }

}
```

```
 private static void inciso_b(Mp_PilaPersona a) {
     int n = a.getNp();
     Persona elem;

     for (int i = 0; i < n; i++) {
         PilaPersona aux1 = new PilaPersona();
         PilaPersona aux2 = new PilaPersona();
         while(!a.esvacia(i)) {
             int men = 0;
         while (!a.esvacia(i)) {
             elem = a.eliminar(i);

             if (elem.getEdad() > men) {
                 men = elem.getEdad();
             }
             aux2.adi(elem);
         }
         a.vaciar(i, aux2);
         while (!a.esvacia(i)) {
             elem = a.eliminar(i);
             if (elem.getEdad()==men) {
                 aux1.adi(elem);
             }else {
                 aux2.adi(elem);
             }
         }
         a.vaciar(i, aux2);
         }
         a.vaciar(i, aux1);
     }
 }
```

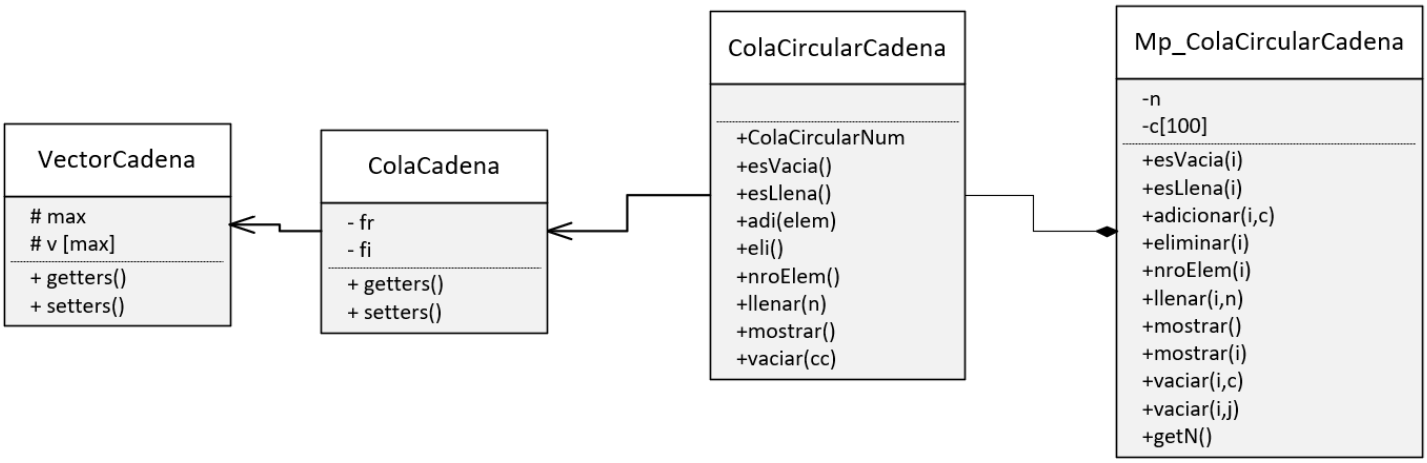
```
private static void inciso_c(Mp_PilaPersona a, int iesima, int jesima) {
    int n = a.getNp();
    Persona elem;
    for (int i = 0; i < n; i++) {
        PilaPersona aux = new PilaPersona();
        if (i == iesima) {
            int nElem = a.nroelem(i);
            int cont=0;
            for (int j = nElem; j >=1; j--) {
                elem = a.eliminar(i);
                cont++;
                System.out.println(elem+" "+j);
                if (j==jesima) {
                    Persona p = new Persona();
                    System.out.println("ANADIENDO UNA PERSONA EN LA POSCION ["+jesima+"] "
                            + "DE LA PILA ["+iesima+"]");
                    p.leer();
                    aux.adi(p);
                }
                aux.adi(elem);
            }

            a.vaciar(i, aux);
        }

    }

}
```

LLAMADO

```
A.adicionar(0, p1);
A.adicionar(0, p2);
A.adicionar(0, p3);
A.adicionar(1, p4);
A.adicionar(1, p5);
A.adicionar(1, p6);
A.adicionar(2, p7);
A.adicionar(2, p8);
A.adicionar(3, p9);
A.adicionar(3, p10);

A.setNp(4);
A.mostrar();
a. Mostrar la cantidad de varones y mujeres de cada pila.
System.out.println("\n\t******INCISO A*******");
inciso_a(A);
b. Ordenar los elementos de la pila con respecto a su edad.
System.out.println("\n\t******INCISO B*******");
System.out.println("\t----------DATOS DE LA PILA SIN ORDENAR POR EDAD---------");
A.mostrar();
inciso_b(A);
System.out.println("\n\t----------DATOS DE LA PILA ORDENADA POR EDAD---------");
A.mostrar();
c. Insertar una nueva persona en la i-ésima posición de la múltiple pila y en la
j-ésima posición de la pila i-ésima.
System.out.println("\n\t******INCISO C*******");
System.out.println("Introduce Iesima posion de la pila multiple: ");
Scanner sc = new Scanner(System.in);
int iesima = sc.nextInt();
System.out.println("Introduce la posion de la pila Jesima a agregar: ");
int jesima = sc.nextInt();
inciso_c(A,iesima,jesima);
System.out.println("\n\t******MULTIPILA INSERTADO UNA PERSONA******");
A.mostrar();
```

EJERCICIO 9

UML



RESOLUCION

```
private static void MoverCaracteres(Mp_ColaCirCadena a) {
    int n = a.getN();
    String elem;
    for (int i = 0; i < n; i++) {
        int numC = a.nroElem(i);
        for (int j = 0; j < numC; j++) {
            elem = a.eliminar(i);
            char[] c = elem.toCharArray();
            boolean swNum = false, swlet = false;
            for (char car : c) {
                if (Character.isDigit(car)) {
                    swNum = true;
                }else {
                    swNum =false;
                }
                if (Character.isAlphabetic(car)) {

                } else {
                    swlet = true;
                }
            }
            if (swNum) {
                a.adicionar(0, elem);
            } else if (swlet) {
                a.adicionar(2, elem);
            } else if (!swlet && !swNum) {
                a.adicionar(1, elem);
            }
        }
    }
}
```

```java
public static boolean esPalindromo(String cadena) {
    char[] caracteres = cadena.toCharArray();
    int izquierda = 0;
    int derecha = caracteres.length - 1;

    while (izquierda < derecha) {
        if (caracteres[izquierda] != caracteres[derecha]) {
            return false;
        }
        izquierda++;
        derecha--;
    }
    System.out.println("SE elimino a: "+cadena);
    return true;
}
```

```java
private static void Eliminar_Palindromos(Mp_ColaCirCadena a, int iesima) {
    int n = a.getN();
    String elem;
    for (int i = 0; i < n; i++) {
        if (i == iesima) {
            int numC = a.nroElem(i);
            for (int j = 0; j < numC; j++) {
                elem = a.eliminar(i);
                if (!esPalindromo(elem)) {
                    a.adicionar(i, elem);
                }
            }
        }
    }
}
```

LLAMADO

```java
public static void main(String[] args) {
    Mp_ColaCirCadena A = new Mp_ColaCirCadena();

    A.adicionar(0, "GG");
    A.adicionar(0, "abCD");
    A.adicionar(0, "1203");
    A.adicionar(0, "Hola");

    A.adicionar(1, "+-+");
    A.adicionar(1, "131B");
    A.adicionar(1, "0123");
    A.adicionar(1, "ana");
    A.adicionar(1, "012210");

    A.adicionar(2, "aBc");
    A.adicionar(2, "ABC");
    A.adicionar(2, "10a");
    A.adicionar(2, "+-*F");

    A.setN(3);
    A.mostrar();
```

```
a. De todas las colas circulares mover a la primera cola circular todas las cadenas
que solo tengan caracteres entre el '0' al '9', mover a la segunda cola circular
todas las cadenas que solo tengan caracteres alfabéticos (tanto mayúsculas
como minúsculas) y en la cola circular 3 las cadenas que no cumplan las
anteriores condiciones.
MoverCaracteres(A);
System.out.println("\t-------INCISO A------");
A.mostrar();
b. Eliminar todas las cadenas que sean palíndromos de la i-ésima cola.
System.out.println("\t-------INCISO B------");
System.out.println("Ingrese la I-esima cola: ");
Scanner sc = new Scanner(System.in);
int iesima=sc.nextInt();
Eliminar_Palindromos(A,iesima);
A.mostrar();
```

```
        *****Datos de la ColaSimple [0]*****
GG
abCD
1203
Hola
        *****Datos de la ColaSimple [1]*****
+-+
131B
0123
ana
012210
        *****Datos de la ColaSimple [2]*****
aBc
ABC
10a
+-*F
        -------INCISO A------
        *****Datos de la ColaSimple [0]*****
1203
0123
012210
        *****Datos de la ColaSimple [1]*****
ana
GG
abCD
Hola
aBc
ABC
        *****Datos de la ColaSimple [2]*****
10a
+-*F
+-+
131B
        -------INCISO B------
Ingrese la I-esima cola:
1
SE elimino a: ana
SE elimino a: GG
        *****Datos de la ColaSimple [0]*****
1203
0123
012210
        *****Datos de la ColaSimple [1]*****
abCD
Hola
aBc
ABC
        *****Datos de la ColaSimple [2]*****
10a
+-*F
+-+
131B
```