UNIVERSIDAD DE CÓRDOBA
comprometida con el desarrollo regional

CENTRO DE INNOVACION EN TIC PARA APOYO A LA ACADEMIA
CINTIA

ARREGLOS EN UNA DIMENSIÓN (VECTORES)

DESARROLLO

CONCEPTOS DE ARREGLOS EN UNA DIMENSIÓN:

Los vectores también se conocen como Arreglos Unidimensionales, estos son estructuras de datos que contienen una colección de datos del mismo tipo. Se puede decir entonces que contienen una secuencia de datos finitos, pertenecientes al mismo dominio. Ejemplo: Las notas de los estudiantes de un curso, el sueldo neto de los empleados de una empresa, los precios de los productos de una tienda, etc.

Los datos que se almacenan dentro de un vector serán los elementos del arreglo y se enumeran desde 0 hasta N-1, siendo N el número de elementos (0, 1, 2, 3…. N-1). Esta enumeración se utiliza para localizar los elementos dentro del arreglo y se denominan índices. En Java los índices de un arreglo comienzan desde 0 para la primera posición y terminan con el "tamaño del arreglo – 1".

Lo anterior quiere decir que si el vector tiene N elementos se denotaran de la siguiente manera: vector[0], vector[2], vector[3], vector[N-1].

- **Propiedades de un arreglo:**

1. Los arreglos sirven de contenedores a la hora de almacenar datos que estén relacionados, de esta manera se pueden declarar arreglos de un determinado tipo de datos, en vez de variables por separado para cada uno de los elementos del vector.

2. Todos los datos almacenados en un arreglo son del mismo tipo. Se pueden crear arreglos de tipo entero (int) o de tipo real (float), pero no se pueden tener dentro de un mismo arreglo datos de tipo int y datos de tipo float (al menos que sea un vector de objetos).

3. A los elementos del vector se accede a través de la posición que ocupan dentro del conjunto de elementos del arreglo, es decir a través de los índices.

4. El tamaño del vector se establece cuando se crea dicho vector, y para esto en Java se utiliza el operador new. El tamaño del vector siempre será una cifra entra (10 elementos, 5 elementos).

- **Declaración de un vector en Java:**

Al declarar un vector se utilizan corchetes para especificar que se trata de un arreglo y no de una variable de algún tipo. Se puede declarar un vector en Java de la siguiente forma:

```
Tipo_Dato  Nombre_Vector [ ];

Tipo_Dato  [ ] Nombre_Vector;
```

Para definir o crear un vector con un número determinado de elementos en tiempo de diseño, se puede utilizar la siguiente sintaxis:

```
private int vectorEdad [ ];

private float vectorSueldos [ ];
```

- **Creación de un vector:**

Para la creación de un vector se utiliza el operador **new**, la sintaxis para definir un arreglo con un número determinado de elementos es la siguiente:

```
Tipo_Dato  Nombre_Vector [ ] = new  Tipo_Dato[numeroDeElementos];
```

Otra forma seria:

```
Tipo_Dato  Nombre_Vector [ ];

Nombre_Vector = new Tipo_Dato[numeroDeElementos];
```

Ejemplos:

```
float vectorNotas = new float [34];

int [ ] vectorNotas;
vectorNotas = new int[34];
```

- **Como acceder a los elementos del vector:**

Para acceder a los elementos de un vector, se utilizan los índices. Esto con el propósito de indicar la posición que ocupa un elemento dentro del vector:

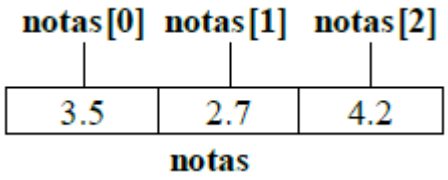
```
Nombre_Vector [índice]
```

La siguiente sintaxis haría referencia a lo que tiene el arreglo de nombre **vectorNotas** en la primera posición del vector.

```
vectorNotas[0]
```

Hay que tener en cuenta que en Java el primer elemento de un vector se encuentra en la posición cero. El índice del último elemento será el número de elementos -1.

Ejemplo: Si se define el siguiente vector **float notas = new float [3];**



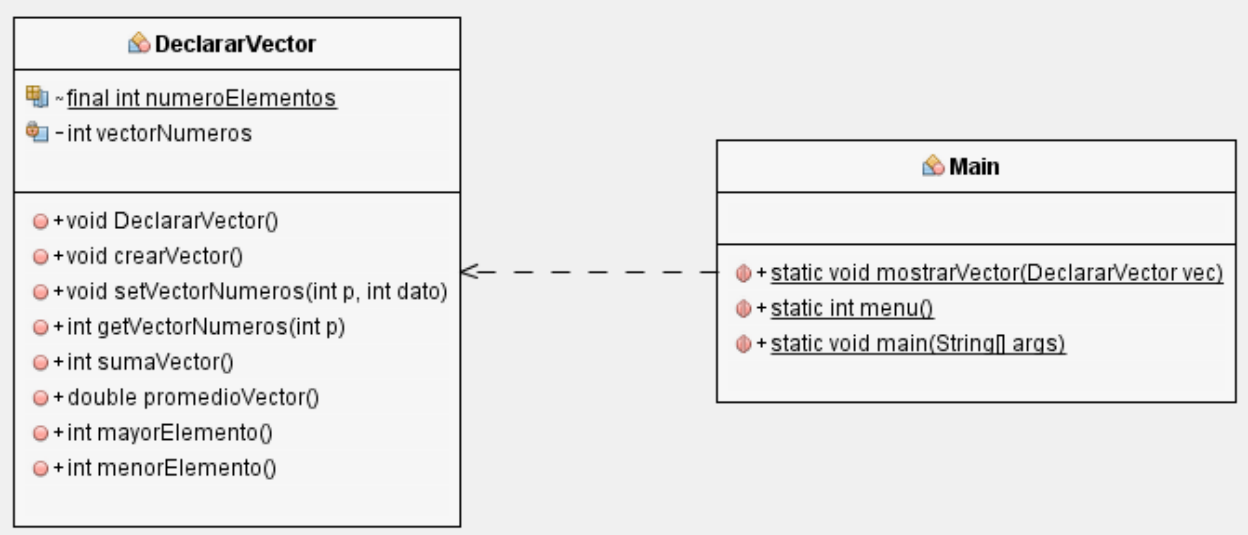
IMPLEMENTACIÓN DE ARREGLOS EN UNA DIMENSIÓN EN JAVA:

- **Ejercicio propuesto:**

El siguiente ejemplo permite declara un vector de números enteros con un tamaño fijo, la aplicación permite almacenar los datos de tipo entero dentro del vector. Una vez lleno el vector se realiza operaciones sobre los elementos almacenados. Entre las operaciones que se realizan sobre los datos del vector están:

1. Calcular la suma y el promedio de los números enteros almacenados dentro del vector.
2. Determinar cuál es el elemento mayor y menor dentro del vector.
3. Imprimir los elementos del vector desde la última posición hasta la primera.

- **Diseño de clases UML de la solución**



- **Implementación de la clase DeclaraVector en el fichero DeclaraVector.java:**

En esta clase **DeclaraVector** se implementan los métodos selectores y modificadores para guardar datos dentro del vector y tomar datos del mismo. Se implementa la lógica de los métodos que permiten sumar todos los elementos del vector, calcular el promedio y determinar cuál es el elemento mayor y menor de los datos registrados.

```
public class DeclararVector {
//Declaración de una constante (numeroElementos) de tipo entero para asignar el tamaño.
    static final int numeroElementos = 5;
    private int vectorNumeros[];
    //Otra forma private int[] vectorNumeros;
    //Esta puede ser otra forma de declarar y crear el vector private int[] vectorNumeros = new int[N];

    //Método constructor de la clase DeclararVector, se asignan os valores iniciales a los atributos.
    public void DeclararVector(){
        vectorNumeros = null;
    }
//El siguiente método crea el vector y le asigna su respectivo tamaño según lo asignado en la
//constante numeroElementos.
    public void crearVector(){
        vectorNumeros = new int[numeroElementos];
    }

//Se implementa el método modificador para asignar los elementos del vector en cada una de las
//posiciones. El método modificador del vector es de tipo procedimiento (void), se le pasan dos
//parámetros: uno de tipo entero (p) correspondiente a la posición del vector en donde se
//almacenara el dato y otro de tipo entero (dato) correspondiente al elemento que se va a
//almacenar, en dicha posición.
    public void setVectorNumeros(int p, int dato){
        vectorNumeros[p] = dato;
    }
```

```
//Método selector del vector para obtener un elemento del vector que este almacenado en alguna
//posición, se pasa como parámetro la posición (p) del elemento que se desea obtener del vector.
    public int getVectorNumeros(int p){
        return vectorNumeros[p];
    }

//Método que suma los elementos del vector, devuelve un dato de tipo entero con el resultado
//de la suma.
    public int sumaVector(){
        int suma = 0;
        for (int i=0; i<=numeroElementos-1; i++){
            suma = suma + getVectorNumeros(i);
        }
        return suma;
    }
//Método que calcula el promedio de los elementos del vector, devuelve un dato de tipo double
//con el promedio.
    public double promedioVector(){
        int contador = 0;
        double suma = 0;
        for (int i=0; i<=numeroElementos-1; i++){
            suma = suma + getVectorNumeros(i);
            contador = contador + 1;
        }
        if(contador > 0){
            return (suma/contador);
        }else{
            return 0;
        }
    }
//Métodos que calculan cual es el elemento mayor y menor de los almacenados en el vector.
    public int mayorElemento(){
//Se asigna a la variable (mayor) el primer elemento del vector.
        int mayor = getVectorNumeros(0);
        for (int i=0; i<=numeroElementos-1; i++){
//Se compara si el elemento de la posición i-esima del vector es mayor que la variable (mayor).
//Si se cumple la condición se asigna a mayor, el elemento de la posición i-esima.
            if (getVectorNumeros(i) > mayor){
                mayor = getVectorNumeros(i);
            }
        }
        return mayor;
    }


    public int menorElemento(){
        int menor = getVectorNumeros(0);
        for (int i=0; i<=numeroElementos-1; i++){
            if (getVectorNumeros(i) < menor){
                menor = getVectorNumeros(i);
            }
        }
        return menor;
    }
}
```

- **Implementación de la clase Main en el fichero Main.java**

En la clase **Main** se implementa el método que recorre el vector y muestra sus elementos, igualmente se implementa un menú de opciones en donde permite registrar los datos en el vector y se muestran los resultados de los métodos implementados en la clase **DeclaraVector**.

```
public class Main {
//Se implementa el método que permite recorrer el vector y muestra los elementos.
   public static void mostrarVector(DeclararVector vec){
      String datosVector = "";
      for(int i=0; i<=vec.numeroElementos-1; i++){
         datosVector = datosVector+String.valueOf("Posición "+i+": "+vec.getVectorNumeros(i)+"\n");
      }
      JOptionPane.showMessageDialog(null, "============ ELEMENTOS DEL VECTOR
============"+"\n"+datosVector);
   }

//Se implementa el método con el menú de opciones de la aplicación para las diferentes
//operaciones sobre el vector.
   public static int menu(){
      int opcion = 0;
      do{
         opcion = Integer.parseInt(JOptionPane.showInputDialog("====== SELECCIONE UNA OPCIÓN
DEL MENÚ ====== \n"+
            "1. LLenar Vector \n"+"2. Suma y Promedio de los Elementos \n"+"3. Elemento Mayor y Menor
\n"+
               "4. Mostar Elementos de Vector \n"+"5. Salir \n"+ "\n \n Seleccione una opción del 1 al 5"));
      }while(opcion <= 0 || opcion > 5);
      return opcion;
   }
   public static void main(String[] args) {
      DeclararVector vec = new DeclararVector();
      int opcion;
      do{
         opcion = menu();
         switch(opcion) {
         case 1:
            vec.crearVector();
            for(int i=0; i<=vec.numeroElementos-1; i++){
               int dato = Integer.parseInt(JOptionPane.showInputDialog(null, "Digitar Elemento de la
Posición "+i+": "));
               vec.setVectorNumeros(i, dato);
            }
            JOptionPane.showMessageDialog(null,"Vector Lleno.... ");
            mostrarVector(vec);
            break;
         case 2:
            JOptionPane.showMessageDialog(null, "========= SUMA Y PROMEDIO DE LOS
ELEMENTOS ========= \n\n"+
                  "Suma de los Elementos: "+vec.sumaVector()+"\n\n Promedio de los Elementos:
"+vec.promedioVector()+
                  "\n\n"
                  + "");
            break;
         case 3:
```

```
            JOptionPane.showMessageDialog(null, "========= ELEMENTO MAYOR Y MENOR DEL
VECTOR ========= \n\n"+
                    "Elemento Mayor: "+vec.mayorElemento()+"\n\n Elemento Menor:
"+vec.menorElemento()+"\n\n"+ "");
            break;
          case 4:
            mostrarVector(vec);
            break;
           case 5:
            break;
          }
       }while(opcion != 5);
   }
}
```

VECTORES EN PARALELO

- **Definición de Vectores en Paralelo:**

Los vectores en paralelos se definen como un mecanismo que asocia la información de una entidad determinada bajo un mismo índice para más de un vector.

- **Características de un vector en paralelo:**

1. Utilizan un mismo índice para agrupar la información de una entidad determinada.
2. El ordenamiento de la información se refleja con base a uno o más atributos de la entidad.
3. Cada atributo de la entidad es un vector.
4. Disponen de un mismo tamaño en cada vector asociado a los atributos de la entidad.

Par entender mejor el concepto de vectores en paralelo, analicemos la siguiente situación: imaginemos que se requiere almacenar la información asociada a los productos de un supermercado, como son:

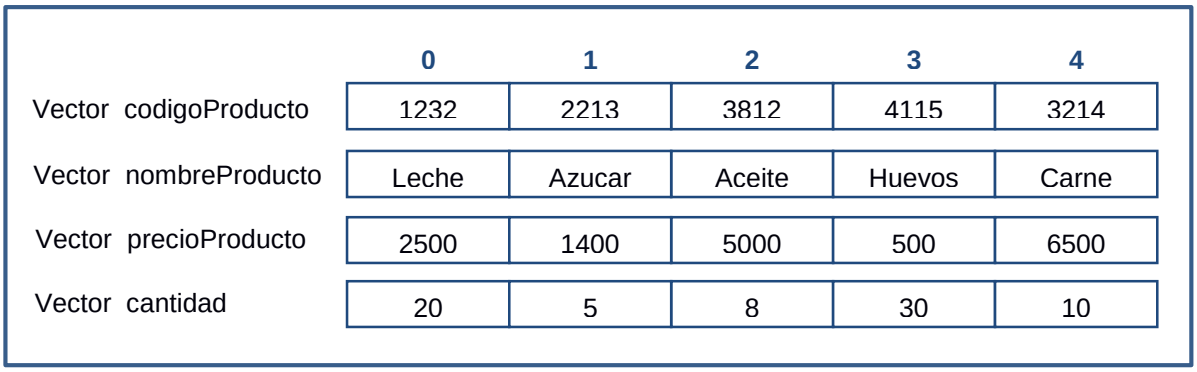
- Código del producto
- Nombre del producto
- Precio del Producto
- Cantidad en existencia

Para la solución se puede tener una clase llamada Productos con los siguientes atributos: codigoProducto, nombreProducto, precioProducto y cantidad. Sin embargo, qué ocurriría si deseamos almacenar la información de 5 productos; se tendrían que crear en nuestra vista 5 objetos de la clase, que representarían cada producto. Como podemos apreciar no se tendría una solución óptima desde el punto de vista del programador, debido a que la información se encuentra dispersa en distintos objetos y se hace más complicado hacer consultas o referencias específicas de la información de la entidad.

Una mejor solución al problema, es utilizar el concepto de **Vectores en Paralelo** permitiendo almacenar los diferentes atributos en vectores asociados a una entidad (Podrían utilizar el concepto de **Vectores de Objetos**, esta temática será tratada más adelante).

Entonces en la solución utilizando Vectores en Paralelo se tendrían que declarar cuatro vectores, para almacenar en cada uno de ellos la información asociada a los atributos de la clase Productos. Es decir, un vector de tipo entero para registrar los códigos de los productos, otro vector de tipo cadena para registrar los nombres de cada producto, uno para los precios y un vector para las cantidades.

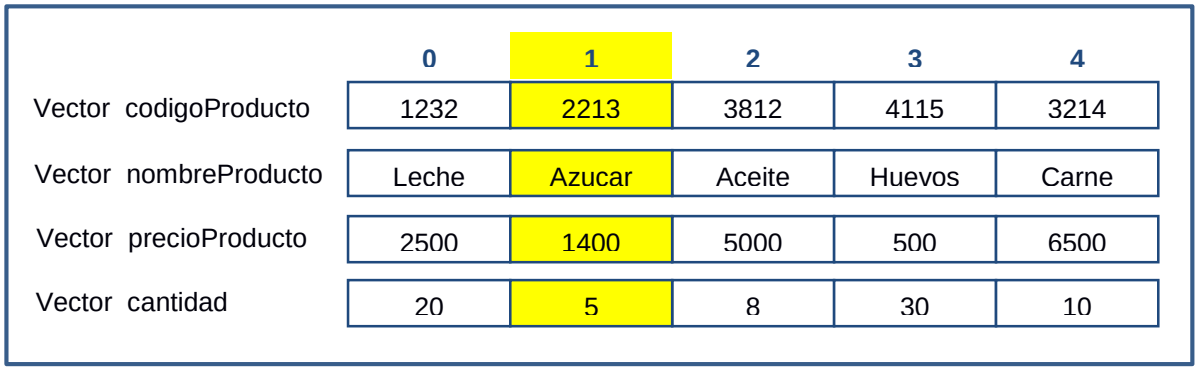
Los cuatro vectores tendrán un mismo tamaño y existirá una correspondencia entre los índices de cada vector para identificar la información correspondiente a un producto específico. En la siguiente figura se aprecia cómo se almacena la información bajo el esquema de vectores en paralelo para un producto.



	0	1	2	3	4
Vector codigoProducto	1232	2213	3812	4115	3214
Vector nombreProducto	Leche	Azucar	Aceite	Huevos	Carne
Vector precioProducto	2500	1400	5000	500	6500
Vector cantidad	20	5	8	30	10

Figura 1. Esquemas Vectores en Paralelo

Como se muestra en la Figura 1, la información de un producto específico se determina basado en los índices correspondientes; es decir la información del producto 1; está definida por los valores en los arreglos codigoProducto[1], nombreProducto[1], precioProducto[1] y cantidad[1] en la posición o índice igual a 1 (Ver Figura 2).



	0	1	2	3	4
Vector codigoProducto	1232	2213	3812	4115	3214
Vector nombreProducto	Leche	Azucar	Aceite	Huevos	Carne
Vector precioProducto	2500	1400	5000	500	6500
Vector cantidad	20	5	8	30	10

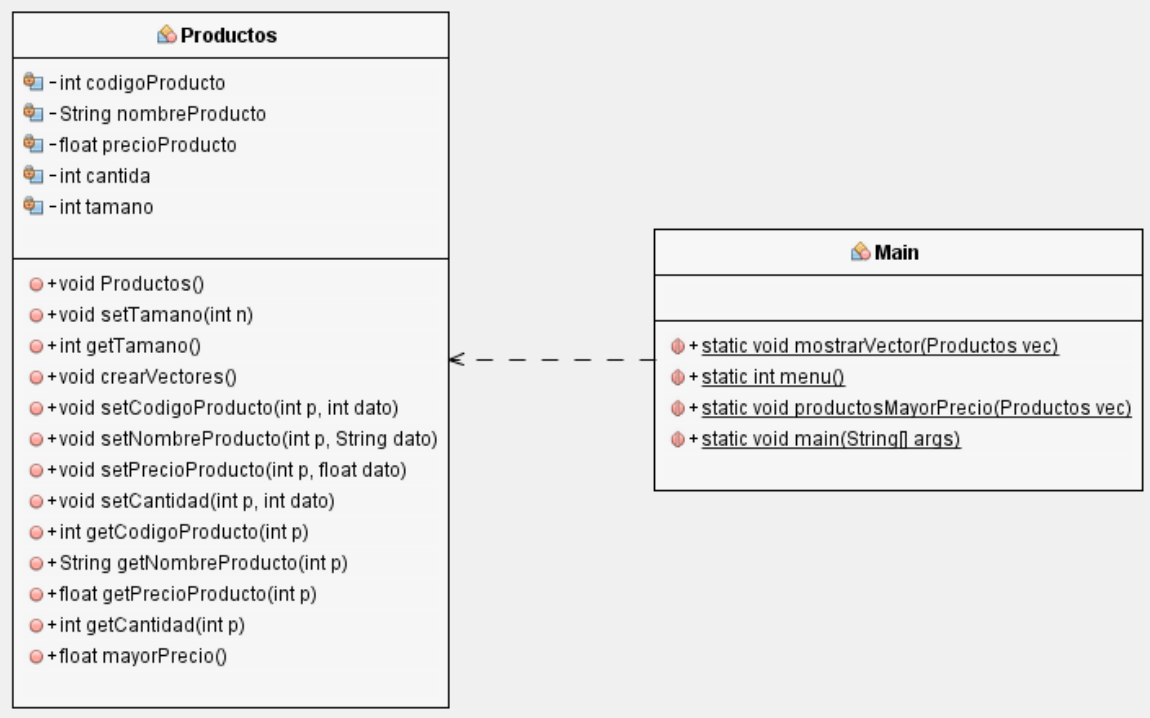
Figura 2. Producto específico con sub_índice 1

IMPLEMENTACIÓN DE VECTORES EN PARALELO

- **Ejercicio propuesto:**

Teniendo en cuenta la problemática plateada anteriormente se requiere registrar la información de los productos de un supermercado y obtener el nombre del producto o los productos de mayor valor.

- **Diseño de clases UML de la solución**



- **Implementación de la clase Productos en el fichero Productos.java**

En la clase Productos se declaran los cuatro vectores para registrar la información de los productos, además se implementa el método que devuelve el mayor precio registrado.

```
public class Productos {
//Se declaran los cuatro vectores para registrar la información de los productos.
    private int codigoProducto[];
    private String nombreProducto[];
    private float precioProducto[];
    private int cantida[];
    private int tamano; //Atributo para asignar el tamaño de los 4 vectores.
//Método constructor de la clase Productos.
```

```
    public void Productos(){
        codigoProducto = null;
        nombreProducto = null;
        precioProducto = null;
        cantida = null;
        tamano = 0;
    }
//Método selector y modificador para asignar el tamaño de los vectores.
    public void setTamano(int n){
        tamano = n;
    }
    public int getTamano(){
        return tamano;
    }
//Método para crear los 4 vectores.
    public void crearVectores(){
        codigoProducto = new int[tamano];
        nombreProducto = new String[tamano];
        precioProducto = new float[tamano];
        cantida = new int[tamano];
    }
//Métodos modificadores para cada vector.
    public void setCodigoProducto(int p, int dato){
        codigoProducto[p] = dato;
    }
    public void setNombreProducto(int p, String dato){
        nombreProducto[p] = dato;
    }
    public void setPrecioProducto(int p, float dato){
        precioProducto[p] = dato;
    }
    public void setCantidad(int p, int dato){
        cantida[p] = dato;
    }

//Métodos selectores para cada vector.
    public int getCodigoProducto(int p){
        return codigoProducto[p];
    }
    public String getNombreProducto(int p){
        return nombreProducto[p];
    }
    public float getPrecioProducto(int p){
        return precioProducto[p];
    }
    public int getCantidad(int p){
        return cantida[p];
    }
```

```
//Método que devuelve el mayor precio.
   public float mayorPrecio(){
      float mayor = getPrecioProducto(0);
      for (int i=0; i<=getTamano()-1; i++){
         if (getPrecioProducto(i) > mayor){
            mayor = getPrecioProducto(i);
         }
      }
      return mayor;
   }
}
```

- **Implementación de la clase Main en el fichero Main.java**

```
public class Main {
//Método que muestra el contenido de los cuatro vectores, manejando una correlación en su posición.
   public static void mostrarVector(Productos vec){
      String datosVector = "";
      for(int i=0; i<=vec.getTamano()-1; i++){
         datosVector = datosVector+String.valueOf("CODIGO: "+vec.getCodigoProducto(i)+"   "+
               "NOMBRE: "+vec.getNombreProducto(i)+"   "+"PRECIO: "+vec.getPrecioProducto(i)+"   "+
               "CANTIDAD: "+vec.getCantidad(i)+"\n");
      }
      JOptionPane.showMessageDialog(null, "========== PRODUCTOS REGISTRADOS EN EL
VECTOR ==========="+"\n\n"+datosVector);
   }


//Método que muestra el menú de opciones al usuario.
   public static int menu(){
      int opcion = 0;
      do{
         opcion = Integer.parseInt(JOptionPane.showInputDialog("===== SELECCIONE SELECCIONE
UNA OPCIÓN DEL MENÚ =====\n\n"+
            "1. Registrar información de los productos \n"+"2. Mostar los productos más costosos \n"+
               "3. Mostrar los productos registrados \n"+"4. Salir"+ "\n \n Seleccione una opción del 1 al 4"));
      }while(opcion <= 0 || opcion > 4);
      return opcion;
   }

//Método que lista los nombres de los productos de mayor precio.
   public static void productosMayorPrecio(Productos vec){
      String datosVector = "";
            for(int i=0; i<=vec.getTamano()-1; i++){
               if(vec.getPrecioProducto(i) == vec.mayorPrecio()){
                  datosVector = datosVector+String.valueOf("NOMBRE: "+vec.getNombreProducto(i)+"\n");
               }
            }
```

```
      JOptionPane.showMessageDialog(null, "========== PRODUCTOS MAS COSTOSOS EN EL VECTOR ==========="+"\n"+datosVector);
   }
   public static void main(String[] args) {
      Productos vec = new Productos();
      int opcion;
      do{
         opcion = menu();
         switch(opcion) {
         case 1:
            int numeroElementos = Integer.parseInt(JOptionPane.showInputDialog(null, "Digite el Número de Productos a Registrar:"));
            vec.setTamano(numeroElementos);
            vec.crearVectores();
            for(int i=0; i<=vec.getTamano()-1; i++){
               int codigo = Integer.parseInt(JOptionPane.showInputDialog(null, "Digite el Código del Producto "+i+": "));
               String nombre = JOptionPane.showInputDialog(null, "Digite el Nombre del Producto "+i+": ");
               float precio = Float.parseFloat(JOptionPane.showInputDialog(null, "Digite el Precio del Producto "+i+": "));
               int cantidad = Integer.parseInt(JOptionPane.showInputDialog(null, "Digite la cantidad de Productos "+i+": "));
               vec.setCodigoProducto(i, codigo);
               vec.setNombreProducto(i, nombre);
               vec.setPrecioProducto(i, precio);
               vec.setCantidad(i, cantidad);
            }
            JOptionPane.showMessageDialog(null,"Productos Registrados.... ");
            mostrarVector(vec);
            break;
         case 2:
            productosMayorPrecio(vec);
            break;
         case 3:
            mostrarVector(vec);
            break;
         case 4:
            break;
         }
      }while(opcion != 4);
   }
}
```

VECTORES DE OBJETOS

- **Definición de Vectores de Objetos:**

Un vector o arreglo de objetos es aquel en el que se pueden almacenar un conjunto de objetos, es decir instancias de una clase cualquiera. De esta manera el tipo de datos almacenado en el vector, no será de tipo convencional (entero, real, carácter, cadena, etc) como en los casos anteriores, sino que esta vez su tipo de dato será del tipo de una clase definida anteriormente o provista por el lenguaje de programación. En Java la sintaxis para declarar vectores de objetos es la siguiente:

```
private NombreClase NombreVector[ ];

private Estudiantes vectorEstudiantes[ ];
```

Donde **NombreClase** es el tipo de la clase que le corresponde a cada uno de los objetos que se almacenaran dentro del vector. En relación al proceso de creación del vector, es igual a como se explicó para vectores de tipos de datos primitivos. A continuación se presenta en Java la sintaxis para crear un vector de tipo de dato no primitivo (Estudiantes), donde Estudiantes indica una clase ya definida:

```
vectorEstudiantes = new Estudiantes[tamano];
```

La anterior línea de código genera la creación de una estructura de datos vector llamado **vectorEstudiantes**, donde se pueden almacenar objetos de la clase **Estudiantes**. Una vez se reserva el espacio para los objetos en el vector, cada uno de dichos objetos se inicializa en nulo (null) de forma automática.

En cuanto al encapsulado de un vector de objetos el diseño e implementación de los métodos modificadores y selectores es el mismo, puesto que cada uno de ellos recibe un parámetro de tipo entero (int) que especifica la posición del elemento (objeto de la clase estudiante) dentro del vector. Lo que cambia, es el tipo de datos de elemento al cual se hace referencia en el vector; pues ya no se trata de un valor de tipo primitivo sino de una instancia de clase, por lo cual al método modificador (set) se le pasa por parámetro: un dato de tipo entero para hacer referencia a la posición y una instancia del tipo de clase Estudiantes.

```
public void setVectorEstudiantes(int p, Estudiantes dato){
        vectorEstudiantes[p] = dato;
}
```

El método selector del vector (get) retornara un objeto de la clase Estudiantes y se declara de la siguiente forma:

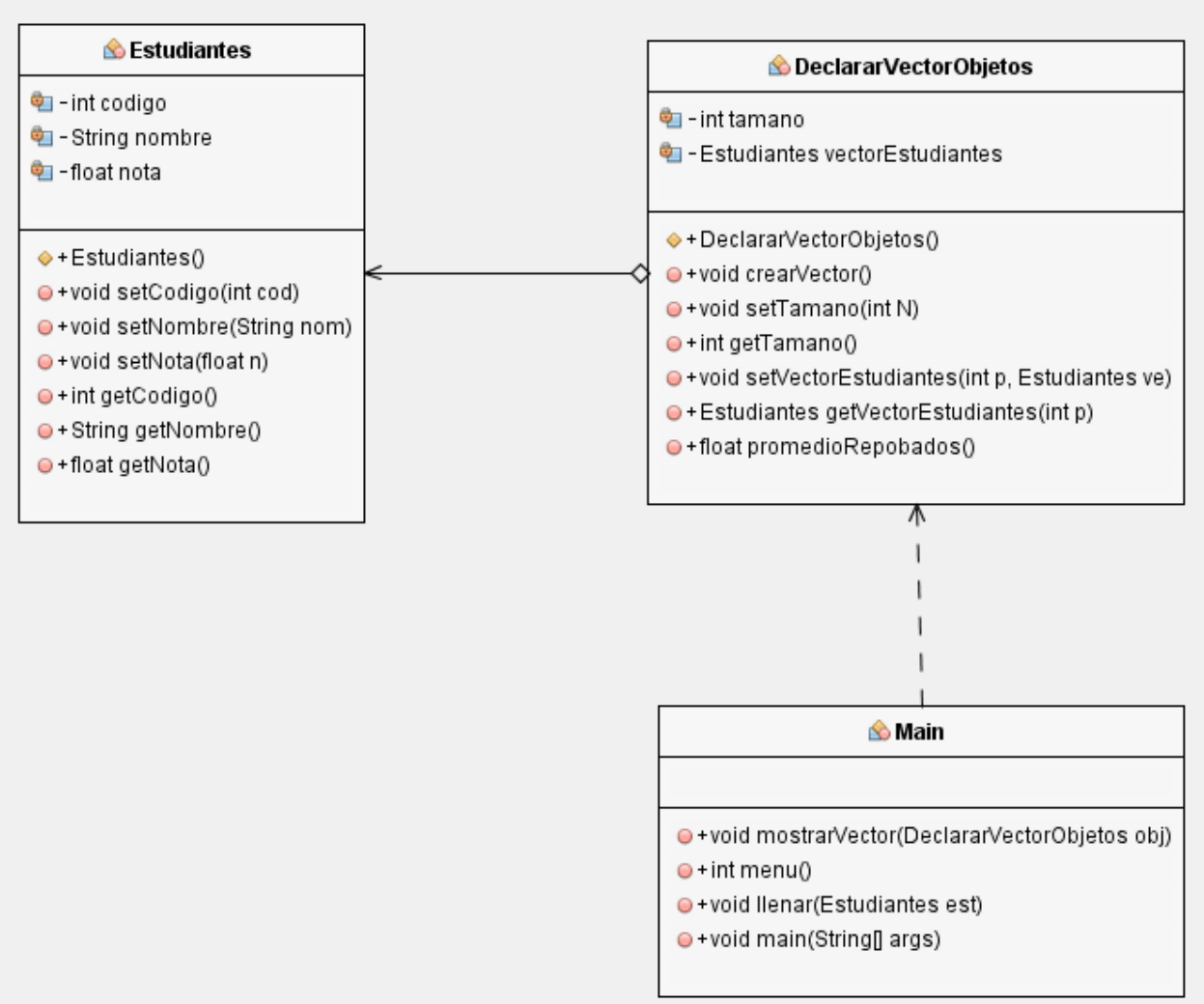
```
public Estudiantes getVectorEstudiantes(int p){
        return vectorEstudiantes[p];
}
```

IMPLEMENTACIÓN DE VECTORES DE OBJETOS

- **Ejercicio propuesto:**

Consideremos entonces el siguiente problema, en donde se quiere registrar la información de los estudiantes del curso de programación, correspondiente a su código, nombre y nota definitiva. Además se requiere de un informe con el promedio de los estudiantes que reprobaron la nota y los nombres de los estudiantes que reprobaron la nota.

- **Diseño de clases UML de la solución**



- **Implementación de la clase Estudiantes en el fichero Estudiantes.java**

```
public class Estudiantes {
    int codigo;
    String nombre;
    float nota;
    public Estudiantes(){
        codigo = 0;
        String nombre = "";
        float nota = 0;
    }
    public void setCodigo(int cod){
        codigo = cod;
    }
    public void setNombre(String nom){
        nombre = nom;
    }
    public void setNota(float n){
        nota = n;
    }
    public int getCodigo(){
        return codigo;
    }
    public String getNombre(){
        return nombre;
    }
    public float getNota(){
        return nota;
    }
}
```

- **Implementación de la clase DeclararVectorObjetos en el fichero DeclararVectorObjetos.java**

```
public class DeclararVectorObjetos {
    private int tamano; //Se define un atributo para asignar el tamaño que tendrá el vector.
    private Estudiantes vectorEstudiantes[];//Se declara el vector de tipo Estudiantes.
    public DeclararVectorObjetos(){
        tamano = 0;
        vectorEstudiantes = null;
    }
    public void crearVector(){
        vectorEstudiantes = new Estudiantes[tamano];
    }

//Métodos modificadores y selectores para asignar y obtener el tamaño del vector.
    public void setTamano(int N){
        tamano = N;
    }
    public int getTamano(){
        return tamano;
    }
```

```
//Métodos modificadores y selectores para agregar información y obtener elementos del vector.
//Se pasan como parámetros la posición (dato entero) y un objeto de la clase Estudiantes (ve).
   public void setVectorEstudiantes(int p, Estudiantes ve){
      vectorEstudiantes[p] = ve;
   }
   public Estudiantes getVectorEstudiantes(int p){
      return vectorEstudiantes[p];
   }
//Método que calcula el promedio de los estudiantes que reprobaron la nota.
   public float promedioRepobados(){
      float suma = 0;
      int contador = 0;
      for(int i=0; i<=getTamano()-1; i++){
         if(getVectorEstudiantes(i).getNota() < 3){
            suma = suma + getVectorEstudiantes(i).getNota();
            contador = contador + 1;
         }
      }
      if(contador > 0){
         return (suma/contador);
      }else{
         return 0;
      }
   }
}
```

- **Implementación de la clase Main en el fichero Main.java**

```
public class Main {
//Implementación del método que muestra los elementos almacenados en el vector, en este caso
//los objetos de la clase Estudiantes.
   public static void mostrarVector(DeclararVectorObjetos obj){
      String datosVector = "";
      for(int i=0; i<=obj.getTamano()-1; i++){
         datosVector = datosVector+String.valueOf("CODIGO: "+obj.getVectorEstudiantes(i).getCodigo()+
               " NOMBRE: "+obj.getVectorEstudiantes(i).getNombre()+
               " NOTA: "+ obj.getVectorEstudiantes(i).getNota()+"\n");
      }
      JOptionPane.showMessageDialog(null, "======= ELEMENTOS DEL VECTOR DE OBJETOS
=======+"\n"+datosVector);
   }

//Implementación del menú de opciones que tendrá la aplicación.
    public static int menu(){
      int opcion = 0;
      do{
         opcion = Integer.parseInt(JOptionPane.showInputDialog("====== MENÚ DE OPCIONES -
VECTOR DE OBJETOS ======\n\n"+
            "1. Registrar la información de los estudiantes\n"+"2. Mostrar la información de los
estudiantes\n"+
               "3. Promedio de los estudiantes que reprobaron\n"+
               "4. Listado de los estudiantes que reprobaron\n"+"5. Salir\n"+
               "\n"+"Seleccione una opción del 1 al 5:"));
      }while(opcion <= 0 || opcion > 5);
```

```
        return opcion;
    }

//Método que asigna valores a los atributos de la Estudiantes.
    public static void llenar(Estudiantes est){
        int cod = Integer.parseInt(JOptionPane.showInputDialog("Digite CODIGO del Estudiante: "));
        est.setCodigo(cod);
        String nom = JOptionPane.showInputDialog("Digite NOMBRE del Estudiante: ");
        est.setNombre(nom);
        float note = Float.parseFloat(JOptionPane.showInputDialog("Digite La NOTA del Estudiante:"));
        est.setNota(note);
    }
    public static void main(String[] args) {
        DeclararVectorObjetos obj = new DeclararVectorObjetos();
        Estudiantes est;
        int opcion;
        do{
            opcion = menu();
            switch(opcion) {
            case 1:
                int numeroElementos = Integer.parseInt(JOptionPane.showInputDialog(null, "Digite el Número
de Elementos del Vector:"));
//Se pasa el dato capturado al respectivo método modificador.
                obj.setTamano(numeroElementos);
//Se llama al método que crea el vector en tiempo de ejecución.
                obj.crearVector();
                for(int i=0; i<=obj.getTamano()-1; i++){
                    est = new Estudiantes();
                    JOptionPane.showMessageDialog(null, "============ ELEMENTOS DEL VECTOR
============"+"\n \n"+
                    "Ingresar la Informacón del estudiante ---- "+(i+1));
                    llenar(est);
                    obj.setVectorEstudiantes(i, est);
                }
                mostrarVector(obj);
                break;
            case 2:
                mostrarVector(obj);
                break;
            case 3:
                JOptionPane.showMessageDialog(null, "======== PROMEDIO DE LOS ESTUDIANTES QUE
REPROBARON ========"+"\n\n"+
                    "Promedio Reprobados: "+obj.promedioRepobados());
                break;
            case 4: //La opción cuatro muestra los nombres de los estudiante que reprobaron.
                String datosVector = "";
                for(int i=0; i<=obj.getTamano()-1; i++){
                    if (obj.getVectorEstudiantes(i).getNota() < 3){
                        datosVector = datosVector+String.valueOf("NOMBRE:
"+obj.getVectorEstudiantes(i).getNombre()+"\n");
                    }
                }
                JOptionPane.showMessageDialog(null,"===== NOMBRE DE LOS ESTUDIANTES QUE
REPROBARON ====="+"\n"+datosVector);
                break;
            case 5:
```

```
                break;
            }
        }while(opcion != 5);
    }
}
```

EJERCICIOS/ACTIVIDADES

1. Se quiere almacenar en un arreglo las N notas de los estudiantes del curso de programación y se desea saber cuántos estudiantes obtuvieron una nota entre 3.0 y 4.0; además de los estudiantes con nota superior a 4.0, el total de estudiantes que reprobaron la materia, el promedio de los reprobados y el promedio general del curso.

2. Se tiene un vector de números enteros para el cual se desea obtener el total de números pares e impares del arreglo, el promedio de los números que son múltiplos de 5, la suma de los números pares y el promedio de los impares.

3. Teniendo en cuenta el ejercicio implementado en el tema de **Vectores en Paralelo**, agregar la siguiente funcionalidad a la aplicación:

    - Generar un Informe con los nombres de los productos que superen un precio de 2500 y tengan menos de 5 productos en existencia.

    - Buscar la información de un producto pasando como paramento su código. Se recomienda implementar un método que devuelva la posición del correspondiente índice del producto buscado.

    - Utilizando otro vector auxiliar, almacenar el valor total que cobrara el supermercado por la venta cada producto en existencia. Se debe tener en cuenta el precio de cada producto y la cantidad en existencia.

4. El bienestar infantil departamental quiere implementar el programa de niños bajos en peso. Para cumplir con este objetivo, se quiere recolectar la información de los niños entre 1 y 5 años de los municipios de Montería, Lorica y Sahagún. Entre los datos generales de los niños están: el número de registro civil y el nombre del niño.

Entre los datos relevantes de cada niño están su talla y peso, necesarios para generar los informes requeridos por el director de la entidad. Para la implementación de la aplicación se deben diseñar en Java las clases necesarias que permitan registrar los datos de cada niño en un Vector de Objetos, se recomienda aplicar el concepto de herencia en la solución (es opcional).

- El bienestar infantil requiere que la aplicación realice las siguientes operaciones y genere informes sobre los datos registrados en el vector de objetos:

- Registrar los datos de los niños que pertenecen al programa.

- Buscar la información de un niño pasando como dato de búsqueda el número de registro civil.

- Generar un listado por municipios, con el nombre de cada niño. Se debe tener un consolidado de cuántos niños tiene el programa en cada municipio.

- Determinar la estatura promedio de los niños entre 3 y 5 años del municipio de Montería. Además determinar el peso promedio de los niños entre 1 y 2 años de edad de Lorica.

- Se considera que los niños entre 3 y 5 años con un peso menor de 20 kilos están bajos de peso, lo mismo para los niños de 1 y 2 años que pesan menos de 10 kilos. La dirección del bienestar infantil quiere un listado con la información (nombre, talla y peso) de los niños que estén en estas condiciones y saber cuántos son en cada municipio para enviar los nutricionistas y mercados con una dieta especial.

BIBLIOGRAFÍA

- Zahonero, I., y Joyanes Aguilar, L. (1999). Estructura de Datos - Algoritmos, Abstracción y Objetos. España: McGraw-Hill.

- Allen Weiss, M. (2004). Estructuras de datos en Java. España: Addison Wesley - Pearson. 776 pp.