INSTITUTO SAN ISIDRO

Curso de Java a distancia

Clase 5: Cadenas de caracteres en Java

En Java hemos visto que cuando queremos almacenar un valor entero definimos una variable de tipo int, si queremos almacenar un valor con decimales definimos una variable de tipo float. Ahora si queremos almacenar una cadena de caracteres (por ejemplo un nombre de una persona) debemos definir un objeto de la clase String.

Más adelante veremos en profundidad y detenimiento los conceptos de CLASE y OBJETO, por ahora solo nos interesa la mecánica para trabajar con cadenas de caracteres.

Problema 1:

Solicitar el ingreso del nombre y edad de dos personas. Mostrar el nombre de la persona con mayor edad.

Programa:

```
import java.util.Scanner;



public class CadenaDeCaracteres1 {

   public static void main(String[] ar) {

      Scanner teclado=new Scanner(System.in);

      String nombre1,nombre2;

      int edad1,edad2;

      System.out.print("Ingrese el nombre:");

      nombre1=teclado.next();
```

```
        System.out.print("Ingrese edad:");
        edad1=teclado.nextInt();
        System.out.print("Ingrese el nombre:");
        nombre2=teclado.next();
        System.out.print("Ingrese edad:");
        edad2=teclado.nextInt();
        System.out.print("La persona de mayor edad es:");
        if (edad1>edad2) {
           System.out.print(nombre1);
        } else {
           System.out.print(nombre2);
        }
    }
}
```

Para almacenar un nombre debemos definir una variable de tipo String y su ingreso por teclado se hace llamando al método next() del objeto teclado:

```
        nombre1=teclado.next();
```

La primera salvedad que tenemos que hacer cuando utilizamos el método next() es que solo nos permite ingresar una cadena de caracteres con la excepción del espacio en blanco (es decir debemos ingresar un nombre de persona y no su nombre y apellido separado por un espacio en blanco)

Veamos que existe otro método llamado nextLine() que nos permite cargar espacios en blanco pero para su uso se complica cuando cargamos otras valores de tipo distinto a String (por ejemplo int, float etc.)

Problema 2:

Solicitar el ingreso del apellido, nombre y edad de dos personas. Mostrar el nombre de la persona con mayor edad. Realizar la carga del apellido y nombre en una variable de tipo String.

Programa:

```
import java.util.Scanner;
```

```
public class CadenaDeCaracteres2 {

   public static void main(String[] ar) {

      Scanner teclado=new Scanner(System.in);

      String apenom1,apenom2;

      int edad1,edad2;

      System.out.print("Ingrese el apellido y el nombre:");

      apenom1=teclado.nextLine();

      System.out.print("Ingrese edad:");

      edad1=teclado.nextInt();

      System.out.print("Ingrese el apellido y el nombre:");

      teclado.nextLine();

      apenom2=teclado.nextLine();

      System.out.print("Ingrese edad:");

      edad2=teclado.nextInt();

      System.out.print("La persona de mayor edad es:");

      if (edad1>edad2) {

         System.out.print(apenom1);

      } else {

         System.out.print(apenom2);

      }

   }

}
```

Cuando se ingresa una cadena con caracteres en blanco debemos tener en cuenta en llamar al método nextLine()
Una dificultad se presenta si llamamos al método nextLine() y previamente hemos llamado al método nextInt(), esto debido a que luego de ejecutar el método nextInt() queda almacenado en el objeto de la clase Scanner el caracter "Enter" y si llamamos inmediatamente al método

nextLine() este almacena dicho valor de tecla y continúa con el flujo del programa. Para solucionar este problema debemos generar un código similar a:

```
System.out.print("Ingrese edad:");
edad1=teclado.nextInt();
System.out.print("Ingrese el apellido y el nombre:");
teclado.nextLine();
apenom2=teclado.nextLine();
```

Como vemos llamamos al método nextLine() dos veces, la primera retorna la tecla "Enter" y la segunda se queda esperando que ingresemos el apellido y nombre (tener en cuenta que esto es necesario solo si previamente se llamó al método nextInt() o nextFloat().

Problema 3:

Solicitar el ingreso de dos apellidos. Mostrar un mensaje si son iguales o distintos.

Programa:

```
import java.util.Scanner;

public class CadenaDeCaracteres3 {
   public static void main(String[] ar) {
      Scanner teclado=new Scanner(System.in);
      String apellido1,apellido2;
      System.out.print("Ingrese primer apellido:");
      apellido1=teclado.next();
      System.out.print("Ingrese segundo apellido:");
      apellido2=teclado.next();
      if (apellido1.equals(apellido2)) {
         System.out.print("Los apellidos son iguales");
      } else {
         System.out.print("Los apellidos son distintos");
      }
   }
}
```

Para comparar si el contenido de dos String son iguales no podemos utilizar el operador ==. Debemos utilizar un método de la clase String llamado equals y pasar como parámetro el String con el que queremos compararlo:

```
if (apellido1.equals(apellido2)) {
```

El método equals retorna verdadero si los contenidos de los dos String son exactamente iguales, esto hace que se ejecute el bloque del verdadero.

Recordemos que hemos utilizado el método next() para la carga de los String, luego esto hace que no podamos ingresar un apellido con espacios en blanco (podemos probar que si

ingresamos por ejemplo "Rodriguez Rodriguez" en el primer apellido, luego se carga la cadena "Rodriguez" en la variable apellido1 y "Rodriguez" en la variable apellido2 (con esto hacemos notar que cada vez que ingresamos un espacio en blanco cuando utilizamos el método next() los caracteres que siguen al espacio en blanco son recuperados en la siguiente llamada al método next())

El método equals retorna verdadero si los contenidos de los dos String son exactamente iguales, es decir si cargamos "Martinez" en apellido1 y "martinez" en apellido2 luego el método equals retorna falso ya que no es lo mismo la "M" mayúscula y la "m" minúscula.

En el caso que necesitemos considerar igual caracteres mayúsculas y minúsculas podemos utilizar el método equalsIgnoreCase:

```
          if (apellido1.equalsIgnoreCase(apellido2)) {
        System.out.print("Los apellidos son iguales sin tener en cuenta mayúsculas y
minúsculas");
     } else {
        System.out.print("Los apellidos son distintos sin tener en cuenta mayúsculas y
minúsculas");
     }
```

Declaración de una clase y definición de objetos.

La programación orientada a objetos se basa en la programación de clases; a diferencia de la programación estructurada, que está centrada en las funciones.
Una clase es un molde del que luego se pueden crear múltiples objetos, con similares características.

Una clase es una plantilla (molde), que define atributos (variables) y métodos (funciones)

La clase define los atributos y métodos comunes a los objetos de ese tipo, pero luego, cada objeto tendrá sus propios valores y compartirán las mismas funciones.

Debemos crear una clase antes de poder crear objetos (instancias) de esa clase. Al crear un objeto de una clase, se dice que se crea una instancia de la clase o un objeto propiamente dicho.

La estructura de una clase es:

```
class [nombre de la clase] {
  [atributos o variables de la clase]
  [métodos o funciones de la clase]
  [main]
}
```

Problema 1:

Confeccionar una clase que permita carga el nombre y la edad de una persona. Mostrar los datos cargados. Imprimir un mensaje si es mayor de edad (edad>=18)

Programa:

```
import java.util.Scanner;
public class Persona {
   private Scanner teclado;
```

```
   private String nombre;
   private int edad;

   public void inicializar() {
      teclado=new Scanner(System.in);
      System.out.print("Ingrese nombre:");
      nombre=teclado.next();
      System.out.print("Ingrese edad:");
      edad=teclado.nextInt();
   }

   public void imprimir() {
      System.out.println("Nombre:"+nombre);
      System.out.println("Edad:"+edad);
   }

   public void esMayorEdad() {
      if (edad>=18) {
         System.out.print(nombre+" es mayor de edad.");
      } else {
         System.out.print(nombre+" no es mayor de edad.");
      }
   }

   public static void main(String[] ar) {
      Persona persona1;
      persona1=new Persona();
      persona1.inicializar();
      persona1.imprimir();
      persona1.esMayorEdad();
   }
}
```

El nombre de la clase debe hacer referencia al concepto (en este caso la hemos llamado Persona):

```
public class Persona {
```

Los atributos los definimos dentro de la clase pero fuera de la main:

```
   private Scanner teclado;
   private String nombre;
   private int edad;
```

Veremos más adelante que un atributo es normalmente definido con la cláusula private (con esto no permitimos el acceso al atributo desde otras clases)

A los atributos se tiene acceso desde cualquier función o método de la clase (salvo la main)

Luego de definir los atributos de la clase debemos declarar los métodos o funciones de la clase. La sintaxis es parecida a la main (sin la cláusula static):

```
public void inicializar() {
   teclado=new Scanner(System.in);
   System.out.print("Ingrese nombre:");
   nombre=teclado.next();
   System.out.print("Ingrese edad:");
   edad=teclado.nextInt();
}
```

En el método inicializar (que será el primero que deberemos llamar desde la main) creamos el objeto de la clase Scanner y cargamos por teclado los atributos nombre y edad. Como podemos ver el método inicializar puede hacer acceso a los tres atributos de la clase Persona. El segundo método tiene por objetivo imprimir el contenido de los atributos nombre y edad (los datos de los atributos se cargaron al ejecutarse previamente el método inicializar:

```
public void imprimir() {
   System.out.println("Nombre:"+nombre);
   System.out.println("Edad:"+edad);
}
```

El tercer método tiene por objetivo mostrar un mensaje si la persona es mayor o no de edad:

```
public void esMayorEdad() {
   if (edad>=18) {
      System.out.print(nombre+" es mayor de edad.");
   } else {
      System.out.print(nombre+" no es mayor de edad.");
   }
}
```

Por último en la main declaramos un objeto de la clase Persona y llamamos a los métodos en un orden adecuado:

```
public static void main(String[] ar) {
   Persona persona1;
   persona1=new Persona();
   persona1.inicializar();
   persona1.imprimir();
   persona1.esMayorEdad();
}
```

Persona persona1; //Declaración del objeto
persona1=new Persona(); //Creación del objeto
persona1.inicializar(); //Llamada de un método

Problema 2:

Desarrollar un programa que cargue los lados de un triángulo e implemente los siguientes métodos: inicializar los atributos, imprimir el valor del lado mayor y otro método que muestre si es equilátero o no.

Programa:

```
import java.util.Scanner;
```

```
public class Triangulo {
   private Scanner teclado;
   private int lado1,lado2,lado3;

   public void inicializar() {
      teclado=new Scanner(System.in);
      System.out.print("Medida lado 1:");
      lado1=teclado.nextInt();
      System.out.print("Medida lado 2:");
      lado2=teclado.nextInt();
      System.out.print("Medida lado 3:");
      lado3=teclado.nextInt();
   }

   public void ladoMayor() {
         System.out.print("Lado mayor:");
      if (lado1>lado2 && lado1>lado3) {
         System.out.println(lado1);
      } else {
         if (lado2>lado3) {
            System.out.println(lado2);
         } else {
            System.out.println(lado3);
         }
      }
   }

   public void esEquilatero() {
      if (lado1==lado2 && lado1==lado3) {
         System.out.print("Es un triángulo equilátero");
      } else {
         System.out.print("No es un triángulo equilátero");
      }
   }

   public static void main(String []ar) {
      Triangulo triangulo1=new Triangulo();
      triangulo1.inicializar();
      triangulo1.ladoMayor();
      triangulo1.esEquilatero();
   }
}
```

Todos los problemas que requieran la entrada de datos por teclado debemos definir un atributo de la clase Scanner:

```
   private Scanner teclado;
```

Este problema requiere definir tres atributos de tipo entero donde almacenamos los valores de los lados del triángulo:

```
   private int lado1,lado2,lado3;
```

El primer método que deberá llamarse desde la main es el inicializar donde creamos el objeto de la clase Scanner y cargamos los tres atributos por teclado:

```
   public void inicializar() {
      teclado=new Scanner(System.in);
      System.out.print("Medida lado 1:");
      lado1=teclado.nextInt();
      System.out.print("Medida lado 2:");
      lado2=teclado.nextInt();
      System.out.print("Medida lado 3:");
      lado3=teclado.nextInt();
   }
```

El método ladoMayor muestra el valor mayor de los tres enteros ingresados:

```
   public void ladoMayor() {
         System.out.print("Lado mayor:");
      if (lado1>lado2 && lado1>lado3) {
         System.out.println(lado1);
      } else {
         if (lado2>lado3) {
            System.out.println(lado2);
         } else {
            System.out.println(lado3);
         }
      }
   }
```

Como podemos observar cuando un problema se vuelve más complejo es más fácil y ordenado separar los distintos algoritmos en varios métodos y no codificar todo en la main.

El último método de esta clase verifica si los tres enteros ingresados son iguales:

```
   public void esEquilatero() {
      if (lado1==lado2 && lado1==lado3) {
         System.out.print("Es un triángulo equilátero");
      } else {
         System.out.print("No es un triángulo equilátero");
      }
   }
```

En la main creamos un objeto de la clase Triangulo y llamamos los métodos respectivos:

```
   public static void main(String []ar) {
      Triangulo triangulo1=new Triangulo();
      triangulo1.inicializar();
      triangulo1.ladoMayor();
      triangulo1.esEquilatero();
   }
```

Problema 3:

Desarrollar una clase que represente un punto en el plano y tenga los siguientes métodos: cargar los valores de x e y, imprimir en que cuadrante se encuentra dicho punto (concepto matemático, primer cuadrante si x e y son positivas, si x<0 e y>0 segundo cuadrante, etc.)

Programa:

```
import java.util.Scanner;
public class Punto {
   private Scanner teclado;
   int x,y;

   public void inicializar() {
      teclado=new Scanner(System.in);
      System.out.print("Ingrese coordenada x :");
      x=teclado.nextInt();
      System.out.print("Ingrese coordenada y :");
      y=teclado.nextInt();
   }

   void imprimirCuadrante() {
      if (x>0 && y>0) {
         System.out.print("Se encuentra en el primer cuadrante.");
      } else {
         if (x<0 && y>0) {
            System.out.print("Se encuentra en el segundo cuadrante.");
         } else {
            if (x<0 && y<0) {
               System.out.print("Se encuentra en el tercer cuadrante.");
            } else {
               if (x>0 && y<0) {
                  System.out.print("Se encuentra en el cuarto cuadrante.");
               } else {
                  System.out.print("El punto no está en un cuadrante.");
               }
            }
         }
      }
   }

   public static void main(String[] ar) {
      Punto punto1;
      punto1=new Punto();
      punto1.inicializar();
      punto1.imprimirCuadrante();
   }
}
```

Definimos tres atributos (el objeto de la clase Scanner y los dos enteros donde almacenamos la coordenada x e y del punto:

```
private Scanner teclado;
int x,y;
```

El método inicializar crea el objeto de la clase Scanner y pide cargar las coordenadas x e y:

```
public void inicializar() {
   teclado=new Scanner(System.in);
   System.out.print("Ingrese coordenada x :");
   x=teclado.nextInt();
   System.out.print("Ingrese coordenada y :");
   y=teclado.nextInt();
}
```

El segundo método mediante un conjunto de if verificamos en que cuadrante se encuentra el punto ingresado:

```
void imprimirCuadrante() {
   if (x>0 && y>0) {
      System.out.print("Se encuentra en el primer cuadrante.");
   } else {
      if (x<0 && y>0) {
         System.out.print("Se encuentra en el segundo cuadrante.");
      } else {
         if (x<0 && y<0) {
            System.out.print("Se encuentra en el tercer cuadrante.");
         } else {
            if (x>0 && y<0) {
               System.out.print("Se encuentra en el cuarto cuadrante.");
            } else {
               System.out.print("El punto no está en un cuadrante.");
            }
         }
      }
   }
}
```

La main no tiene grandes diferencias con los problemas realizados anteriormente, declaramos un objeto de la clase Punto, creamos el objeto mediante el operador new y seguidamente llamamos a los métodos inicializar e imprimirCuadrante en ese orden:

```
public static void main(String[] ar) {
   Punto punto1;
   punto1=new Punto();
   punto1.inicializar();
   punto1.imprimirCuadrante();
}
```

Problema 4:

Desarrollar una clase que represente un Cuadrado y tenga los siguientes métodos: cargar el valor de su lado, imprimir su perímetro y su superficie.

Programa:

```
import java.util.Scanner;
public class Cuadrado {
   private Scanner teclado;
   int lado;

   public void inicializar() {
      teclado=new Scanner(System.in);
      System.out.print("Ingrese valor del lado :");
      lado=teclado.nextInt();
   }

   public void imprimirPerimetro() {
      int perimetro;
      perimetro=lado*4;
      System.out.println("El perímetro es:"+perimetro);
   }

   public void imprimirSuperficie() {
      int superficie;
      superficie=lado*lado;
      System.out.println("La superficie es:"+superficie);
   }

   public static void main(String[] ar) {
      Cuadrado cuadrado1;
      cuadrado1=new Cuadrado();
      cuadrado1.inicializar();
      cuadrado1.imprimirPerimetro();
      cuadrado1.imprimirSuperficie();
   }
}
```

En este problema es interesante ver como no definimos dos atributos donde se almacenan la superficie y el perímetro del cuadrado, esto debido a que solo estos datos se los requiere en el método donde se imprimen:

```
   public void imprimirPerimetro() {
      int perimetro;
      perimetro=lado*4;
      System.out.println("El perímetro es:"+perimetro);
   }
```

Esto significa que la variable perimetro es una variable local al método imprimirPerimetro. Esta variable es local a dicho método y solo se la puede acceder dentro del método. La diferencia

fundamental entre una variable local y un atributo de la clase es que al atributo se lo puede acceder desde cualquier método de la clase y la variable local solo existe mientras se está ejecutando el método.

Problemas propuestos

1. Confeccionar una clase que represente un empleado. Definir como atributos su nombre y su sueldo. Confeccionar los métodos para la carga, otro para imprimir sus datos y por último uno que imprima un mensaje si debe pagar impuestos (si el sueldo supera a 3000)
2. Implementar la clase operaciones. Se deben cargar dos valores enteros, calcular su suma, resta, multiplicación y división, cada una en un método, imprimir dichos resultados.

Solución

```
import java.util.Scanner;
public class Empleado {
   private Scanner teclado;
   String nombre;
   float sueldo;

   public void inicializar() {
      teclado=new Scanner(System.in);
      System.out.print("Ingrese el nombre del empleado:");
      nombre=teclado.next();
      System.out.print("Ingrese su sueldo:");
      sueldo=teclado.nextFloat();
   }

   public void pagaImpuestos() {
      if (sueldo>3000) {
         System.out.print("Debe abonar impuestos");
      } else {
         System.out.print("No paga impuestos");
      }
   }

   public static void main(String[] ar) {
      Empleado empleado1;
      empleado1=new Empleado();
      empleado1.inicializar();
      empleado1.pagaImpuestos();
   }
}
```

```
import java.util.Scanner;
public class Operaciones {
   private Scanner teclado;
   int valor1,valor2;

   public void inicializar() {
      teclado=new Scanner(System.in);
      System.out.print("Ingrese primer valor:");
      valor1=teclado.nextInt();
      System.out.print("Ingrese segundo valor:");
      valor2=teclado.nextInt();
```

```
    }

    public void sumar() {
        int suma;
        suma=valor1+valor2;
        System.out.println("La suma es:"+suma);
    }

    public void restar() {
        int resta;
        resta=valor1-valor2;
        System.out.println("La resta es:"+resta);
    }

    public void multiplicar() {
        int multiplicacion;
        multiplicacion=valor1*valor2;
        System.out.println("La multiplicación es:"+multiplicacion);
    }

    public void dividir() {
        int division;
        division=valor1/valor2;
        System.out.println("La división es:"+division);
    }

    public static void main(String[] ar) {
        Operaciones opera;
        opera=new Operaciones();
        opera.inicializar();
        opera.sumar();
        opera.restar();
        opera.multiplicar();
        opera.dividir();
    }
}
```

Declaración de métodos.

Cuando uno plantea una clase en lugar de especificar todo el algoritmo en un único método (lo que hicimos en los primeros pasos de este tutorial) es dividir todas las responsabilidades de las clase en un conjunto de métodos.

Un método hemos visto que tiene la siguiente sintaxis:

```
public void [nombre del método]() {
  [algoritmo]
}
```

Veremos que hay varios tipos de métodos:

Métodos con parámetros.

Un método puede tener parámetros:

```
public void [nombre del método]([parámetros]) {
  [algoritmo]
}
```

Los parámetros los podemos imaginar como variables locales al método, pero su valor se inicializa con datos que llegan cuando lo llamamos.

Problema 1:

Confeccionar una clase que permita ingresar valores enteros por teclado y nos muestre la tabla de multiplicar de dicho valor. Finalizar el programa al ingresar el -1.

Programa:

```
import java.util.Scanner;
public class TablaMultiplicar {
   public void cargarValor() {
      Scanner teclado=new Scanner(System.in);
      int valor;
      do {
         System.out.print("Ingrese valor:");
         valor=teclado.nextInt();
         if (valor!=-1) {
            calcular(valor);
         }
      } while (valor!=-1);
   }

   public void calcular(int v) {
      for(int f=v;f<=v*10;f=f+v) {
         System.out.print(f+"-");
      }
   }
```

```
    public static void main(String[] ar) {
        TablaMultiplicar tabla;
        tabla=new TablaMultiplicar();
        tabla.cargarValor();
    }
}
```

En esta clase no hemos definido ningún atributo, ya que el objeto de la clase Scanner lo requerimos en un solo método, por ello lo definimos como una variable local.

El método calcular recibe un parámetro de tipo entero, luego lo utilizamos dentro del método para mostrar la tabla de multiplicar de dicho valor, para esto inicializamos la variable f con el valor que llega en el parámetro. Luego de cada ejecución del for incrementamos el contador f con el valor de v.

```
    public void calcular(int v) {
        for(int f=v;f<=v*10;f=f+v) {
            System.out.print(f+"-");
        }
    }
```

Un método puede no tener parámetros como hemos visto en problemas anteriores o puede tener uno o más parámetros (en caso de tener más de un parámetro los mismos se separan por coma)

El método cargarValores no tiene parámetros y tiene por objetivo cargar un valor entero por teclado y llamar al método calcular para que muestre la tabla de multiplicar del valor que le pasamos por teclado:

```
    public void cargarValor() {
        Scanner teclado=new Scanner(System.in);
        int valor;
        do {
            System.out.print("Ingrese valor:");
            valor=teclado.nextInt();
            if (valor!=-1) {
                calcular(valor);
            }
        } while (valor!=-1);
    }
```

Como vemos al método calcular lo llamamos por su nombre y entre paréntesis le pasamos el dato a enviar (debe ser un valor o variable entera)

En este problema en la main solo llamamos al método cargarValor, ya que el método calcular luego es llamado por el método cargarValor:

```
    public static void main(String[] ar) {
        TablaMultiplicar tabla;
        tabla=new TablaMultiplicar();
        tabla.cargarValor();
    }
```

Métodos que retornan un dato.

Un método puede retornar un dato:

```
public [tipo de dato] [nombre del método]([parámetros]) {
 [algoritmo]
 return [tipo de dato]
}
```

Cuando un método retorna un dato en vez de indicar la palabra clave void previo al nombre del método indicamos el tipo de dato que retorna. Luego dentro del algoritmo en el momento que queremos que finalice el mismo y retorne el dato empleamos la palabra clave return con el valor respectivo.

Problema 2:

Confeccionar una clase que permita ingresar tres valores por teclado. Luego mostrar el mayor y el menor.

Programa:

```
import java.util.Scanner;
public class MayorMenor {
   public void cargarValores() {
         Scanner teclado=new Scanner(System.in);
      System.out.print("Ingrese primer valor:");
      int valor1=teclado.nextInt();
      System.out.print("Ingrese segundo valor:");
      int valor2=teclado.nextInt();
      System.out.print("Ingrese tercer valor:");
      int valor3=teclado.nextInt();
      int mayor,menor;
      mayor=calcularMayor(valor1,valor2,valor3);
      menor=calcularMenor(valor1,valor2,valor3);
      System.out.println("El valor mayor de los tres es:"+mayor);
      System.out.println("El valor menor de los tres es:"+menor);
   }

   public int calcularMayor(int v1,int v2,int v3) {
      int m;
      if(v1>>v2 && v1>v3) {
        m=v1;
      } else {
         if(v2>v3) {
            m=v2;
         } else {
          m=v3;
         }
      }
      return m;
   }
```

```
   public int calcularMenor(int v1,int v2,int v3) {
      int m;
      if(v1<v2 && v1<v3) {
        m=v1;
      } else {
         if(v2<v3) {
            m=v2;
         } else {
          m=v3;
         }
      }
      return m;
   }

   public static void main(String[] ar) {
      MayorMenor maymen=new MayorMenor();
      maymen.cargarValores();
   }
}
```

Si vemos la sintaxis que calcula el mayor de tres valores enteros es similar al algoritmo visto en conceptos anteriores:

```
   public int calcularMayor(int v1,int v2,int v3) {
      int m;
      if(v1>v2 && v1>v3) {
        m=v1;
      } else {
         if(v2>v3) {
            m=v2;
         } else {
          m=v3;
         }
      }
      return m;
   }
```

Lo primero que podemos observar que el método retorna un entero y recibe tres parámetros:

```
   public int calcularMayor(int v1,int v2,int v3) {
```

Dentro del método verificamos cual de los tres parámetros almacena un valor mayor, a este valor lo almacenamos en una variable local llamada "m", al valor almacenado en esta variable lo retornamos al final con un return.

La llamada al método calcularMayor lo hacemos desde dentro del método cargarCalores:

```
      mayor=calcularMayor(valor1,valor2,valor3);
```

Debemos asignar a una variable el valor devuelto por el método calcularMayor. Luego el contenido de la variable mayor lo mostramos:

```
System.out.println("El valor mayor de los tres es:"+mayor);
```

La lógica es similar para el cálculo del menor.

Muchas gracias hasta la próxima clase.

Alsina 16 [B1642FNB] San Isidro | Pcia. De Buenos Aires |Argentina |

TEL.: [011] 4742-1532 o [011] 4742-1665 |

www.institutosanisidro.com.ar info@institutosanisidro.com.ar