# *Ejercicios de Programacion Concurrente*

## *Ejemplo en netbeans 7*

**1. HILOS**

```
package ClaseHilos;


public class Hilos {

   public static void main (String args[]){

      ImprimaHilo InstanciaImprimaHilo1, InstanciaImprimaHilo2, InstanciaImprimaHilo3,
InstanciaImprimaHilo4;

      //Las variables creadas, instancia del objeto ImprimaHilo

      InstanciaImprimaHilo1 = new ImprimaHilo();

      InstanciaImprimaHilo2 = new ImprimaHilo();

      InstanciaImprimaHilo3 = new ImprimaHilo();

      InstanciaImprimaHilo4 = new ImprimaHilo();

      InstanciaImprimaHilo1.start();

      InstanciaImprimaHilo2.start();

      InstanciaImprimaHilo3.start();

      InstanciaImprimaHilo4.start();

   }

}


class ImprimaHilo extends Thread {

   int sleepTime;

   //constructor encargado de generar el tiempo de espera a paratir de la funcion

   //random, e imprime el hilo o thread actual y el tiempo aleatorio almacenado
```

```
   //en la variable sleepTime


   public ImprimaHilo(){

      //Tiempo de espera aleatorio entre  0 y 5 segundos

      sleepTime = (int)(Math.random()*5000);

      //Imprime el nombre del hilo actual y el tiempo aleatorio

      System.out.println ("Nombres del hilo que se ejecuatara: "+ getName()+ "; Tiempo que se
tardara: " + sleepTime);

   }


   public void run(){

      //pone el tiempo de espera aleatorio de espera en el hilo para su debida ejecucion

      //dependiendo del numero aleatorio arrojado en la parte de arriba demoraza

      //en ejecutarce el hilo

      try {

         Thread.sleep ( sleepTime );

      }

      catch (InterruptedException exception){

         System.err.println (exception.toString());

      }

      System.out.println("\nEspere    "+sleepTime+"   Milisgundos para ejecutar el   "+
getName());

   }

}
```

**2. INTERFACES**

```
package ClaseHilos;


interface animal1{
   public int edad = 10;
   public String nombre = "Bob";


   public void nace();
}


interface animal2{
   public void get_nombre();
}


interface animal3{
   void get_nombre(int i);
}


class ImplementInterfaces implements animal1,animal2,animal3{
   ImplementInterfaces(){
      get_nombre();
      get_nombre(8);
      //edad = 10; no podemos cambiar este valor
   }
```

```
    public void nace(){
        System.out.println("hola mundo");
    }

    public void get_nombre(){
        System.out.println(nombre );
    }

    public void get_nombre(int i){
        System.out.println(nombre +" " +i);
    }
}
```

**3. SOBRE CARGAR METODOS**

```
package ClaseHilos;


class SobrecargarMetodos {
  int x, y, radio;

  SobrecargarMetodos(int puntoX, int puntoY, int tamRadio) {
     this.x = puntoX;
     this.y = puntoY;
     this.radio = tamRadio;
     Resultado();
  }

  SobrecargarMetodos(int puntoX, int puntoY) {
     this(puntoX, puntoY, 2 ) ;
     Resultado();
  }

  final void Resultado(){
     int resultado = x*y*radio;
     System.out.println(resultado);
  }
  public static void main(String[] arguments) {
     SobrecargarMetodos Prueba = new SobrecargarMetodos(2,3,4);
     SobrecargarMetodos Prueba2 = new SobrecargarMetodos(2,3);
```

```
}          }
```

**4. HERENCIA**

```
package herencia;

import java.io.*;

class claseSuperMadre{
   //protected int cedula;
   int cedula;
   String nombre;

   public claseSuperMadre(int cedula, String nombre){
      this.cedula=cedula;
      this.nombre=nombre;
   }

   public void saludo(){
      System.out.println("Gracias. Este es solo un ejemplo");
   }

   /*public void obtener_nombre(){
      System.out.println("Tu nombre es: "+nombre);
   }*/

   public void obtener_nombre(int i){
```

```
        System.out.println("Tu nombre es: " + nombre + " " + "y tu cedula es: " + cedula);
    }

    public void obtener_cedula(){
        System.out.println("Tu numero de cedula es: "+cedula);
    }
}

class herenciaClase extends claseSuperMadre{
    herenciaClase(int cedula, String nombre){
        super(cedula,nombre);
    }
    static void saludo(herenciaClase InstanciaClaseHija){
        InstanciaClaseHija.saludo();
    }

    static void obtener_Datos(herenciaClase InstanciaClaseHija){
        InstanciaClaseHija.obtener_cedula();
    }

    @SuppressWarnings("static-access")
    public static void main(String[] args){
        int cedula = 0;
        String nombre = " ";
        IngresarDatosTeclado instanciaIngresarDatosTeclado = new IngresarDatosTeclado();
```

```
        System.out.println("Programa para trabajar herencias");
        System.out.print("Digite su cedula: ");
        cedula = Integer.parseInt(instanciaIngresarDatosTeclado.IngresarDatosTeclado());

        System.out.print("Digite su nombre: ");
        nombre = instanciaIngresarDatosTeclado.IngresarDatosTeclado();

        herenciaClase InstanciaClaseHija = new herenciaClase(cedula,nombre);
        InstanciaClaseHija.obtener_Datos(InstanciaClaseHija);
        InstanciaClaseHija.obtener_nombre(cedula);

        InstanciaClaseHija.saludo();
    }
}

class IngresarDatosTeclado{
    String IngresarDatosTeclado(){
        String dato=" ";
        try{
            BufferedReader flujo = new BufferedReader(new InputStreamReader(System.in));
             dato = flujo.readLine();
        }
        catch (Exception e){
            System.out.println("Lo siento digito un dato erroneo");
            System.out.println(e);
        }
```

```
      return (dato);
   }
}
```

**5. MENUS**

```
package menus;
import java.io.*;

//Lectura instantanea a traves de fichero, de dato
//entero ingresado por teclado.
//Ojo que la instancia del objeto global o de clase llamaMenu
//debe ser del tipo estatico, de lo contrario arroja error
//Dado que siempre las variables de clase deben ser estaticas por seguridad

public class menus{

   static menu llamaMenu = new menu();

   public static void main(String[] args){

      llamaMenu.menu();
   }

   //declaro varibles a usar
```

```
  static class menu{

    void menu(){

      char letra=' ';

      //creando un objeto llamado teclado especializado en capturas

      BufferedReader teclado = new BufferedReader(new InputStreamReader(System.in));

      //Capturando datos

      try{
        System.out.println("\n"+"*** MENU DE OPCIONES PARA LECTURA DE DATOS POR
TECLADO ***");
        System.out.println("\n"+"Elija una de las siguientes opcion: ");
        System.out.println("\n"+"a) Procedimiento para leer datos enteros por teclado");
        System.out.println("b) Procedimiento para leer datos tipo caracter por teclado");
        System.out.print("\n"+"Elija una opcion : ");

        letra = teclado.readLine().charAt(0);
      }
      catch (IOException e){
        System.out.print("\n"+"Error de captura en: "+e);
      }
```

```
        switch(letra){
           case 'a': case 'A':
              //Aqui llamo a los archivos donde estan las clases para ejecutar programas


              DatosEnteros leaEnteros = new DatosEnteros();
              leaEnteros.DatosEnteros();
              llamaMenu.menu();


           case 'b': case 'B':
              DatosCadena leaCadena = new DatosCadena();
              leaCadena.DatosCadena();


              //Fijese que puedo utilizar la palabra resercada this para
              //referirme a la clase actual en la que estoy trabajando,
              //es decir this.menu(); puede reemplazar a llamaMenu.menu();


              this.menu();


           default:
              System.out.println("\n"+"Lo siento la opcion digitada no existe");
              llamaMenu.menu();
        }
     }
}
```

```
static class DatosEnteros{

   void DatosEnteros(){
      //Crea el conducto para transportar la corriente de datos
      //desde el programa hasta el sumidero.
      //El sumidero es la entrada standard
      //crear un objeto del tipo InputStreamReader
      //(lectura de corriente entrante)

      InputStreamReader flujo;
      BufferedReader in;
      flujo = new InputStreamReader(System.in);
      in = new BufferedReader(flujo);

      String linea;
      int entrada;
      System.out.println("\n"+"Ejemplo de lectura de datos enteros por teclado, uno por linea.");
      System.out.print("Digite un dato entero: ");

      try{
         while((linea=in.readLine())!=null){
            entrada = Integer.parseInt(linea);
            System.out.println("\n"+"El dato entero ingresado por teclado es: " + entrada);
            System.out.print("Digite un dato entero: ");
         }
```

```
        }
        catch(Exception e){
           System.out.println("\n"+"Lo siento no digito un dato tipo entero");
        }

        //Lectura instantanea atraves de fichero, de dato tipo
        //cadena ingresado por teclado.

    }
}

static class DatosCadena{

    //Crea el conducto para el tranportar los datos desde el programa
    //hasta el sumidero. El sumidero es la entrada standard

    void DatosCadena(){
        InputStreamReader flujo;
        BufferedReader in;
        flujo = new InputStreamReader(System.in);
        in = new BufferedReader(flujo);

        String linea;

        System.out.println("\n"+"Ejemplo de lectura de datos String, uno por linea");
        System.out.print("Digite un caracter: ");
```

```
        try{
           while((linea=in.readLine())!=null){
              System.out.println("\n"+"El dato caracter ingresado por teclado es: " + linea);
              System.out.print("\n"+"Digite un caracter: ");
           }
        }
        catch(Exception e){
           System.out.println("Lo siento no digito un dato tipo entero");
        }
     }
  }
}
```

**6. MATRICES**

```
//Matriz por referencia
package menus;


public class matriz {


  void matriz(){


    try{
      int MatrizNum [][] =  {
                    {1,2,3},
                    {6,7,8},
                    {9,0,11}
                    };


      String Cadena [][] = {
                   {"edgar ","leandro ","munoz"},
                   {"Es ","una ","matriz"},
                   {"Esta ","es ","prueba"}
                  };
      mostrarMatr imprimir = new mostrarMatr();
      imprimir.mostrarMatr(MatrizNum,Cadena);
    }
    catch(Exception e){
      System.out.println("Error: "+ e);
    }
```

```
    }

    static class mostrarMatr{

        void mostrarMatr(int[][] Numeros, String[][] Nombres) {

            for(int i=0;i<3;i++){
                for(int j=0;j<3;j++){
                    System.out.print(" "+Numeros[i][j]+" ");
                }
            }
            System.out.println("\n");
            for(int i=0;i<3;i++){
                for(int j=0;j<3;j++){
                    System.out.print(" "+Nombres[i][j]+" ");
                }
            }
        }
    }
      public static void main(String[] args){
        matriz leerMatriz = new matriz();
        leerMatriz.matriz();
    }
}
```

**7. CIBER CAFÉ**

```
/*Cafe Internet*/
package quiz;
import java.io.*;

public class quiz {

  public static void main(String[] args) throws IOException{

    BufferedReader lectura = new BufferedReader(new InputStreamReader(System.in));
    try{
      int tam;
      System.out.print("Digite el numero de usuarios: ");
      tam = Integer.parseInt(lectura.readLine());

      int [] usuarios = new int [tam];
      int [] horas = new int [7];
      char dias[] ={'L', 'M', 'M', 'J', 'V', 'S','D'};
      int i, c=1,b,sum=0,min=0;

      for(i=0;i<tam;i++){
        System.out.print("\n"+"Digite el numero de minutos del usuario Nº "+c+"\n");

        for(b=0;b<7;b++){
          System.out.println("en el dia "+dias[b]+" :");
          horas[b] = Integer.parseInt(lectura.readLine());
```

```
    if(horas[b]>=0){
        min=min+horas[b];
        int hora=horas[b];
        char dia;

        dia = dias[b];

        valorDeLasHoras Costos = new valorDeLasHoras();
        Costos.valorDeLasHoras(hora,dia);
    }
    else{
        System.out.print("\n"+"Recuerde que no puede digitar numeros negativos"+"\n");
        b--;
    }
  }

  if (min>=60)
        sum=min/60;

  usuarios[i]=sum;
  sum=0;
  min=0;
  c++;
}
```

```
        valorReferencia OperacRefer = new valorReferencia();
        OperacRefer.valorReferencia(usuarios);

        imprimirVectores imprimir = new imprimirVectores();
        imprimir.imprimirVectores(usuarios,horas);
    }
    catch(NumberFormatException ex){
        System.out.println("Ocurrio algun error en: " + ex);
    }
}

static class valorReferencia {

    void valorReferencia(int datos []) {
        int i,t=datos.length,c=1;
        for(i=0;i<t;i++){
            if(datos[i]>=5 && datos[i]<=7 )
                System.out.print("\n"+"El usuario Nº "+c+" ha ganado 1 hora gratis");

            else if(datos[i]>7 && datos[i]<=10)
                System.out.print("\n"+"El usuario Nº "+c+" ha ganado 2 hora gratis");

            c++;
        }
    }
}
```

```
static class valorDeLasHoras {


   void valorDeLasHoras(int val, char dia) {
      int valor;
      valor=val;
      if(valor==0){
         System.out.print("minutos consumidos"+"\n\n");
      }


      if(valor>0 && valor<=15){
         System.out.print("minutos consumidos y tuvo que pagar $ 250 pesos."+"\n\n");
      }


      if(valor>15 && valor<=30){
         System.out.print("minutos consumidos y tuvo que pagar $ 500 pesos"+"\n\n");
      }


      if(valor>30 && valor<=45){
         System.out.print("minutos consumidos y tuvo que pagar $ 750 pesos"+"\n\n");
      }


      if(valor>45 && valor<=60){
         System.out.print("minutos consumidos y tuvo que pagar $ 1000 pesos"+"\n\n");
      }
```

```
        if(valor>60){
           if(dia != 'D'){
              costoMinutos OperValor = new costoMinutos();
              int costo;
              costo=OperValor.costoMinutos(valor);
              System.out.print("minutos consumidos y tuvo que pagar $ "+costo+" pesos"+"\n\n");
           }
           if(dia == 'D'){
              costoMinutos OperValor = new costoMinutos();
              double costo,obt;
              costo=OperValor.costoMinutos(valor);
              obt=costo*0.1;
              costo=costo-obt;
              System.out.print("minutos consumidos y tuvo que pagar $ "+ costo +" pesos con un
descuento del 10%"+"\n\n");
           }
        }
     }
  }


  static class costoMinutos {


     int costoMinutos(int mins) {
        int res,coste=0,vuelto;
        res=mins;
        while(res>=60){
```

```
            res=res-60;
            coste++;
        }
        if(res==0)
            return coste*1000;

        else {
            valorDelosMinutos Costos = new valorDelosMinutos();
            vuelto=Costos.valorDelosMinutos(res);
            return ((coste*1000)+vuelto);
        }
    }
}

static class valorDelosMinutos {

    int valorDelosMinutos(int num) {
        int valor;
        valor=num;
        if(valor>=0 && valor<=15){
            return 250;
        }

        else if(valor>15 && valor<=30){
            return 500;
        }
```

```
        else if(valor>30 && valor<=45){
            return 750;
        }
        else if(valor>45 && valor<=60){
            return 1000;
        }
        return 0;
    }
  }

  static class imprimirVectores{
    void imprimirVectores(int[] Us, int[] Ho){
        int ta=Us.length;
        System.out.println("\n\n");
        for(int i=0;i<ta;i++){
            System.out.println("Vector Usuario["+Us[i]+"]");
        }
        System.out.println("\n\n");
        for(int i=0;i<Ho.length;i++){
            System.out.println("Vector horas["+Ho[i]+"]");
        }
    }
  }
}
```