UNIDAD 0.

Concepto orientado a objetos (O.O.)

La importancia de la Programación Orientada a Objetos (POO) es que permite que los programadores diseñen software en forma muy parecida a como perciben el mundo real, de manera que se puedan entender entre sí los programadores y los no programadores.

La POO es el lenguaje para tratar y relacionar lo abstracto con los objetos reales.

Objetos: Son conjuntos de software, de datos y procedimientos, esta mezcla proporciona un medio más exacto de representar objetos del mundo real en el software.

Ejemplo de objetos reales.

Nombre (o Id): León.

Lo que es (color olor aspecto): Amarillo, grande con melena.

Lo que hace (conducta): duerme, ruge.

Ejemplo de objetos en programación:

Nombre: Nube.

Variables: string color=blue;

int tamaño=25;

int cant_agua: 25;

Funciones: llover ();

evapora ();

nevar ();

Esto es en general una clase.

Una clase es una descripción de atributos y funciones, y tiene la sintaxis como un tipo de dato.

Objeto es la materialización de una clase.

Ejemplo:

int a;

nube nube1;

nube1.color=blanco;

nube1.tamaño=100;

nube nube2;

nube2.color=gris;

nube2.tamaño=200;

nube1.llover ();

nube2.llover ();

UNIDAD 1

Programación en Java.

Origen del Java.

En un principio Java se llamó OAK y fue dirigido a electrodomésticos, pero fracasó. En 1995 se retomó este lenguaje ahora aplicado a Internet con el nombre de Java, y esta vez tuvo éxito.

El nombre de Java viene de una isla de Indonesia y su logotipo es una taza de café.

Java Virtual Machine.

Al realizar un programa en Java se realiza un archivo con extensión *.java, y al compilarlo e ejecutable tiene extensión *.class. En ese ejecutable el 80% se usa para hacer lo que se le programó, y el 20% para el manejo de plataforma (Windows, Linux, Solaris). De ese proceso se encarga el Java Virtual Machine; por ésta razón es que los virus disminuyen pero la compilación es lenta.

Entorno del Lenguaje.

Al instalar Java J2SDK1.4 crea las carpetas bin, demo, include, JRF, lib; todo esto muy similar a C++.

Editor.

El editor que usaremos será el Jcreator Integrated Development Enviroment, pero cabe señalar que no es el único, también existen como el JBuilder, IDE etc.

Glosario:

- API: Application Programming Interface (Ayuda).
- J2EE, J2SE, J2ME: Versiones de Java, Enterprise Edition, Estándar Edition, y Micro Edition.
- JDK: Java Development Kit.
- JRE: Java Runtime Enviroment.
- AWT: Abstract Window Toolkit.
- SWING: AWT mejorado.
- JDBC: Java Database Connectivity.
- Applet: Aplicación que se ejecuta desde un Browser.
- Servlet: Aplicación sin interfaz gráfica que se ejecuta en un servidor de Internet.
- Bean: Rutinas para transmisión entre aplicaciones.

Asistencia Web.

La página de Java.sun.com permite bajar el lenguaje J2SDK, la API, y demás completamente gratis.

Versiones de Java.

JDK 1.0

JSDK 1.0

J2SDK 1.2

J2SDK 1.3

J2SDK 1.4.0.1

J2SDK 1.4.0.2

UNIDAD II. CONVENCIONES DE ESCRITURA

Java utiliza las mayúsculas y minúsculas, pero convencionalmente usaremos las mayúsculas en:

Métodos: darHora

Clases: MiReloj

Interfaces: MiInterface

Los objetos se escriben al igual que los métodos y las constantes con mayúsculas.

Si la clase es declarada como public el nombre del archivo debe ser igual al nombre de la clase. Ejemplo: archivo: hola.java clase: class hola

Variables y Operadores.

Variable Primitiva: son las que tienen un nombre de variable y un solo valor. Se escriben en minúsculas. Ejemplo: int a=12;

Variable ADT (Abstract Data Tipe u Objeto): Contiene métodos, es un objeto preconstruido por Java. Empiezan con mayúsculas. Ejemplos:

a1.toString();

Integer a1;

Intb=a1.perseInt();

String s2;

String s3=s2.subString(2,2);

Operadores: Aritméticos: + , −, /, *, %

Relacionales: <>, <=, <, >

Lógicos: &&, ||

Declaración e Instanciación de Objetos

Hay tres formas:

- Forma Universal: clase objeto=new Clase(); Integer var=new Integer(); // declaración e instancia junta.
- Forma Orientada a Objetos:: Integer in; in=new Integer(58); // Se separa la declaración y la instancia.
- Forma Desechable: no lleva referencia del nombre, es como un efecto solamente. new miObjeto(), new String();

Clases, Métodos y Variables.

Existen 3 formas de diseñar un programa dependiendo de la complejidad del programa.

Modo Main, modo Objeto y Constructor y modo Objeto y Método.

Modo Main:

```
class Mayor

{ public statis void main (String arg[])

{

int a=12,b=13, c=5, d=4, m, n, o;

System.out.println(prueba +a);

m=Math.max(a,b);

n=Math.max(c,d);

o=Math.max(m,n);

System.out.println(mayor +o);

System.out.println(Math.max( Math.max(a,b) , Math.max(c,d) ); }

}
```

Constructor: es un método del mismo nombre de la clase y se utiliza normalmente para inicializar a las variables.

Modo Objeto y Constructor:

```
class Mayor

{ int a,b,c,d;

Mayor(int x, int y,int z, int d)

{ a=x; b=y; c=z; this.d=d; }

public static void main (String arg[])

{ Mayor m=new Mayor(12,13,4,5);

System.out.printl(Math.max(Math.max(a,b),Math.max(c,d)));

}

}
```

Public: se define como la única clase de ese tipo, y si se define public deberá ser igual que el nombre del archivo, si no marcará error.

Modo Objeto y Método:

```
public class Mayor

{ calcula (int a, int b, int c, int d)

{ int r=(Math.max(Math.max(a,b),Math.max(c,d)));

return r;

}

public static void main (String arg[])

{ int a=12, b=14, c=15, d=8;

Mayor ma;

ma=new Mayor();

System.out.println(El mayor es: +ma.calcula(a,b,c,d));

}

}
```

Programas que calculan el área de un círculo de las tres formas:

```
class Circulo

{ public static void main (String arg[])
```

```
{ int r=9, a;
a= Math.PI * (r * r);
System.out.println(area: +a);
}
}
class Circulo
{ int r;
Circulo(int x)
{ r = x; }
public static void main (String arg[])
{ Circulo d= new Circulo(9);
System.out.println(area: +Math.PI * d.r * d.r);
}
}
public class Circulo
{ area ( int r );
{ int rad=r * r * Math.PI;
return rad;
}
public static void main (String arg[])
{ int x=9;
Circulo z;
z = new Circulo();
System.out.printl(El mayor es: +z.area(x));
}
}
```

Packages e Imports.

Packages: es un grupo de clases compiladas que se utilizan posteriormente con un import.

Para crear el propio Package se pondrá al inicio del programa Pakage(nombre).

Los packages mas comunes son:

import java.AWT; //ventanas

import java.io; //archivos

import java.util; //utilerías

import java.Applet; //aplicaciones de Internet

import java.Graphics; //gráficos

import java.NET; //usa protocolos de red TCP/IP

Modificadores de Acceso

Clases:

Sintaxis: <modificador> class nombreClase{ }

Tipos de modificadores:

- Public: hace a la clase accesible desde otra clase o package.
- Abstract: se define como clase base para heredar, no se puede instanciar. Ejemplo: Math, Graphics.
- Final: evita subclases, no hereda, es una medida de seguridad contra hackers.

Metodos:

Sintaxis: nombre_del_metodo(<parámetro>){ }

Tipos de modoficadores:

- Public.
- Protected
- Private
- Synchronizable: se utiliza para eventos concurrentes o multitareas.
- Static: se ejecuta sin necesidad de hacer una instancia de un objeto.
- Final
- Transient: atributos que no se graban cuando se archiva un objeto, no forma parte de estado permanente del mismo.
- Volatile: son variables modificadas asíncronamente por objetos en diferentes trees.
- Native: se utiliza para usar código de otro lenguaje.

Estructura Básica de un Programa.

```
package pak; // opcional

import java.AWT.*;

public class Prueba

{ //atributos

int altura;

Button boton1;

//metodo constructor

Prueba()

{ new boton1; }

class uno

{ metodos() }

metodos();

public static vois main (String arg[])

{ new Prueba(); }// instancia de la clase

}
```

Tipos de Clases.

- Anónima: Se utiliza para manejo de eventos, en vez de definir el nombre de la clase se pasa directamente a la implementación del código.
- Adapter: Es la versión abreviada de la interfaz gráfica.
- Principal: es la clase contenedora o más grande (Public).
- Internas: Es cada una de las clases que se encuentran dentro de la clase pincipal.
- Interface: Es un conjunto de declaraciones de métodos vacíos (sin código), el programador inserta el código de esos métodos, su objetivo es establecer flujos de control y para programar más genéricamente.

Comandos Super y This.

Super: llama al constructor de la clase padre o clase escendente (también llama a métodos de la clase ascendente).

This: señala a variables o métodos de la clase presente o actual.

Ejemplos:

```
class A
{ int i, j;
A(int a, int b)
{ I=a; j=b; }
void show()
{ System.out.println(ij+I+ +j); }
class B extends A
{ int k;
B (int a, int b, int c)
{ super(a,b); k=c; }
void show()
{ System.out.println(k+k); }
void otro()
{ super.show();
this.show();
System.out.println(super i +super.i);
}
}
calss Overviide
{ public static void main (String arg[])
{ B obj=new b(1,2,3);
obj.otro();
}
}
}
```

Método Main.

Se declara e instancia la clase principal, es el inicio de la ejecución.

Interface.

```
import java.awt.*;

import java.awt.event.*;

public class dameMes extends Frame implements ActionListener

{ Textfield a,b;

Button enter,salida;

public dameMes()

{ setTitle (Lee número y da Estación);

a= new TextField(4);

a=new TextField(mes,14);

enter=new Button(enter);

salida=new Button(exit);

enter.addActionListener(this);

salida.addActionListener(this);

add(a); add(enter); add(b); add(salida);

setLayout(new FlowLayout());

setVisible(true);

setSize(400,200);

}

public void actionPerformed (ActionEvent e)

{ if(e.getSource()==enter)

{ String s=new String(a.getText());

Integer c3=new Integer();

int c2 = c3.parseInt(s);

switch(c2)
```

```
{ case 12: case 1: case 2: estación=Invierno;

break; .

.

.

}

b.getText(estacion);

}

if(e.getSource()==salida)

System.exit(0);

}

public static void main (String arg[])

{ dameMes d=new dameMes();

}

}
```

Manejo de Excepciones.

Consiste en monitorear condiciones factibles de generar un error y transferir el control del programa a un código específico para manejar dichas excepciones.

Utilización del Try−catch:

1.− Enciérrese el código que se quiere monitorear dentro del bloque Try.

2.− Inmediatamente después se incluye un Catch y un bloque con la excepción adecuada para ese error.

Ejemplo:

```
Class Exc

{ public static void main (String arg[])

{ int d=0;

try{ int a=42/d;

System.out.println(¿imprimirá?);

}
```

```
catch(ArithmeticException e)
{System.out.println(Error al dividir entre cero);}
}
}
```

La excepción generada es un objeto que viene directa o indirectamente del Throwable.

Error: Virtual Machine Error: Stack Overflow Error

Out of Memory Error

Exception: Runtime Exception: Arithmetic Exception

NullPointer Exception

Array IndexOutofBound Exception

IO Exception: EOF Exception

FileNotFound Exception

Ejemplo de un Try y varios Catch.

```
calss x{ public static void main (String arg[])
{ int c[]={1};
try{ int b=47 / 0;
c[42]=99; }
catch(ArithmeticException e)
{ System.out.println(Error div 0); }
catch(ArrayIndexOutOfBoundsException a)
{ System.out.println(Rebaso Array+a); }
finally
{ System.out.println(otro error que no se);}
}
}
```

Comando throw

```
class manda{

static void metodo1(){

try{ throw new NullPointerException(demo); }

catch (NullPointerException e)

{ System.out.println(catched);

throw e;

}

}

public static void main (String arg[]){

try{ metodo1(); }

catch(NullPointerException e)

{ System.out.println(rechazada+e); }

}

}
```

Comentarios en Java.

Al igual que en C, los comentarios se hacen con // y con /**/

UNIDAD 3

Clase Object, Number, Math, String y Format

3.1 Datos Primitivos.

Son aquellos que los proporciona el lenguaje y existen diferentes tipos:

Numéricos enteros.

Nombre del Tipo	**Tamaño en Bytes**	**Rango**
Byte	1	−128 a 127
Short	2	−32768 a 32767
Int	4	-2^{31} a 2^{31}
Long	8	-2^{3} a 2

Numéricos decimales.

Nombre del Tipo	**Tamaño en Bytes**	**Rango**

Float	4	$-3.4x10^3$ a $3.4x10^3$
Double	8	$-1.7x10^3$ a $1.7x10^3$

Tipo carácter.

Nombre del Tipo	**Tamaño en Bytes**	**Rango**
Char	2	Conjunto de caracteres

Tipo lógico(booleano).

Nombre del Tipo	**Tamaño en Bytes**	**Rango**
boolean	1	True, false

3.2 ADT Abstract Data Type, datos en envoltura.

Generan objetos y tienen métodos y atributos. Un método abstracto es un método que se declara, pero no se define. Entre ellos tenemos: Integer, String, Byte.....

3.3 Cast (conversión de datos)

```
float b; // modo cast

int a=(int) b; //cambiar flotante a entero.

Integer i=(Integer)a; //se combierte a integer.
```

Modo de conversión.

```
//Primitivo a ADT

int a=45;

Integer i=new Integer(a);

//ADT a primitivo

int c= i.intValue(); //si parámetros

//Char primitivo a String ADT

char v='4';

Character c= new Character(v);

String s= new String(c.toString());

//String a primitivo

char arr[]= s.toCharArray();

char uno=arr[0];
```

```
// Textfield a arreglo de caracteres

char arr[]= x.getText.toString.toCharArray();

//String a numérico primitivo

String s= "18";

float f= Float.parseFloat(s);

int i= Integer.parseInt(s);
```

3.4 Clase Object.

```
//Ejemplo

import java.awt.*;

import java.awt.event.*;

public class Juego1 extends Frame implements ActionListener

{

long m, n;

long m1=0, n1=0;

int c1=0, c2=0, c3=0;

TextField a,b,c,d,e;

Button enter, salida;

public Juego1() {

setTitle("Jueguito");

a=new TextField (4);

b=new TextField (4);

c=new TextField (4);

d=new TextField (4);

e=new TextField (22);

enter=new Button ("enter");

salida=new Button ("exit");
```

```
enter.addActionListener(this);
salida.addActionListener(this);
add(a); add(b); add(enter); add(c); add(d); add(e); add(salida);
setLayout(new FlowLayout());
setVisible(true);
setSize(400,400);
}
public void actionPerformed(ActionEvent p){
if (p.getSource()==enter)
{
m=Math.round(Math.random()*10);
n=Math.round(Math.random()*10);
m1=m1+m;
n1=n1+n;
if(m1>n1)
{
c1++;
}
if(m1<n1)
{
c2++;
}
if(m1==n1)
{
c3++;
}
```

```
int m2=(int)m;
int n2=(int)n;
int m3=(int)m1;
int n3=(int)n1;
Integer i=new Integer(m2);
a.setText(i.toString());
Integer j=new Integer(n2);
b.setText(j.toString());
Integer k=new Integer(m3);
c.setText(k.toString());
Integer l=new Integer(n3);
d.setText(l.toString());
if((c1>2)||(c2>2))
{
enter.disable();
if((c3>c1)&&(c3>c2))
{
e.setText("Esto es un empate");
}
if(c1>c2)
{
e.setText("El ganador es el Jugador 1");
}
if (c1<c2){
e.setText("El ganador es el jugador 2");
}
```

```
}
}
if (p.getSource()==salida)
{
System.exit(0);
}
}
public static void main(String arg[]){
Juego1 r=new Juego1();
}
}
```

3.6 Clase Format.

```
import java.text. Decimal Format;
class Formatos{
psvm{
DecimalFormat twoDigits= new DecimalFormat("0.00");
DecimalFormat separado= new DecimalFormat ("$#,##0.00");
DecimalFormat porciento= new DecimalFormat("%");
DecimalFormat cientifico= new DecimalFormat("E");
double d= 13244.56
double p= 0.56;
int a= 43;
SOP(twoDigits.format(a)); //43.00
SOP(separado.format(d)); //$13,244.56
SOP(porciento.format(p)); //%56
SOP(cientifico.format(d)); //E13245
```

```
}
}
import java.text.SimpleDateFormat;
class formatos{
psvm{
Date fecha= new Date(); //información del día (fecha y hora)
SimpleDateFormat sd;
sd=new SimpleDateFormat("ddMMMYYYY hh:mm:ss YYYY);
SOP(sd.format(fecha)); //23 FEB 2004 11:21:55 2004
}
}
```

Clases de uso común.

```
Class comunes{
psvm{
Character c= new Character();
char a[]= {'a', 'b', 's', ' ', 'A'};
if (c.isLetter(a[0]))
SOP(a[0]+"es una letra"); // a es una letra
if (Character.isLowerCase(a[1]))
SOP("es minuscula");
SOP(c.isDigit(a[2])); //regresa un valor booleano true
SOP(c.isUpperCase(a[4])); //true
SOP(c.toUpperCase(a[0])); //A
}
}
```

3.7 Clase String, métodos más comúnes

```
class str{

psvm(s){

char c[]= {'J', 'A', 'V', 'A'};

String s3= new String(c,2,2); //s3=va

String s1= new String(c);

String s2= new String(s1);

SOP(s1.equals(s2)); // regresa un booleano true

SOP(s1==s2); //false a menos de que sean objetos hermanos o de la misma instancia.

String s5= new String("JAVA");

SOP(s1.equalsIgnoreCase(s5)); //true

SOP(s5.indexof("va")); //2

SOP(s5.indexof('v')); //2

SOP(s5.lastIndexof('a')); //3

SOP(s5.subString(3,1)); //A

StringBuffer sb= new StringBiffer("Hello");

SOP(sb.charAt(1)); //e

SOP(sb.append("Hola que)); //HelloHola que

SOP(sb.insert(4, "oooo")); //HellooooHola que

SOP(sb.delete(4,8)); //Hellaque

SOP(sb.reverse()); //euqalleH

SOP(sb.replace(3,3,"was")); //uqawash

SOP(sb.deletecharAt(0)); //uqallell
```

- Elabore un programa que calcule el **rfc** :

```
import java.awt.*;

import java.awt.event.*;

import java.util.StringTokenizer;
```

```
import java.lang.String;
public class Rfc1 extends Frame implements ActionListener
{
TextField a,b;
Button boton, salida;
public Rfc1() {
setTitle("RFC del profe");
a=new TextField (30);
b=new TextField (10);
boton=new Button ("RFC");
salida=new Button ("Salir");
boton.addActionListener(this);
salida.addActionListener(this);
add(a); add(boton); add(b); add(salida);
setLayout(new FlowLayout());
setVisible(true);
setSize(400,400);
}
public void actionPerformed(ActionEvent p){
if (p.getSource()==boton)
{
StringBuffer sb=new StringBuffer();
String r= new String(a.getText());
StringTokenizer st= new StringTokenizer(a.getText(),",");
String ele=new String();
while (st.hasMoreTokens()){
```

```
ele=(String) st.nextToken();
if (st.countTokens()==5)
sb.append(ele.toUpperCase().charAt(0));
if(st.countTokens()==4)
{
sb.insert(0,ele.toUpperCase().charAt(1));
sb.insert(0,ele.toUpperCase().charAt(0));
}
if(st.countTokens()==3)
sb.insert(2,ele.toUpperCase().charAt(0));
if(st.countTokens()==2){
sb.append(ele.charAt(0));
sb.append(ele.charAt(1));
}
if(st.countTokens()==1)
{
if(ele.equalsIgnoreCase("Enero"))
ele="01";
if(ele.equalsIgnoreCase("Febrero"))
ele="02";
if(ele.equalsIgnoreCase("Marzo"))
ele="03";
if(ele.equalsIgnoreCase("Abril"))
ele="04";
if(ele.equalsIgnoreCase("Mayo"))
ele="05";
```

```
if(ele.equalsIgnoreCase("Junio"))
ele="06";
if(ele.equalsIgnoreCase("Julio"))
ele="07";
if(ele.equalsIgnoreCase("Agosto"))
ele="08";
if(ele.equalsIgnoreCase("Septiembre"))
ele="09";
if(ele.equalsIgnoreCase("Octubre"))
ele="10";
if(ele.equalsIgnoreCase("Noviembre"))
ele="11";
if(ele.equalsIgnoreCase("Diciembre"))
ele="12";
sb.insert(4,ele);
}
if(st.countTokens()==0)
sb.insert(4,ele.substring(2,4));
}
b.setText(sb.toString());
}
if (p.getSource()==salida)
{
System.exit(0);
}
}
```

```
public static void main(String arg[]){

Rfc1 r=new Rfc1();

}

}
```

UNIDAD 4

AWT Interfaces Gráficas

Jerarquía de Clases.

Label: Función meramente informativa o señalativa.

TextField: Recibe y muestra información, se le puede manejar como variable y puede iniciar eventos.

Button: sirve para informar y/o iniciar eventos (no almacena información).

Canvas: Lienzo de dibujo, no almacena información, solamente despliega.

CheckBox: puede almacenar una o más opciones de un conjunto de opciones.

Choice: solamente se puede seleccionar una opción de un conjunto de opciones, su apariencia es de un Filter.

List: recibe información, despliega, puede ser considerado variable y puede ser un choice.

Menu: Facilita una interfaz pop−up.

Container:

- **Frame:** es una ventana o pantalla completa.
- **File Dialog:** una ventana para que el usuario seleccione un archivo.
- **Panel:** porción rectangular de una ventana pre−existente (generalmente un Frame).
- **Applet:** panel dentro de un Browser web.

Batalla Naval.

Para ejemplo de estas aplicaciones, en este ejercicio se va a presentar la interacción de AWT con Swing; realización de un ActionListener, una clase Adapter y un Layout.

(Fragmento del programa original)

```
import java.awt.*;

import java.awt.event.*;

import java.awt.color.*;

import javax.swing.*;
```

```
import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import java.awt.Label;

import javax.swing.ImageIcon;

import java.awt.FlowLayout;

import java.awt.event.ActionEvent;

import javax.swing.JOptionPane;

ImageIcon boom=new ImageIcon("a:/bum.gif");

ImageIcon nuevo=new ImageIcon("a:/nuevo.gif");

JButton B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11;

Panel MiPanel, MiPanel1, MiPanel3,p4,pn;

List listita;

Label j2, j1;

Batalla()

{

MiPanel=new Panel();

MiPanel1=new Panel();

MiPanel3=new Panel();

pn=new Panel();

B1=new JButton();

B1.setBackground(Color.black);

B1.setIcon(barco);

B2=new JButton();

B2.setBackground(Color.black);

B2.setIcon(barco);
```

```
B1.addActionListener(new B1a());

B2.addActionListener(new B2a());

B3.addActionListener(new B3a());

B4.addActionListener(new B4a());

}

class B1a implements ActionListener

{

public void actionPerformed (ActionEvent e)

{

if(b1==1)

{

cont1++;

B1.setIcon(boom);

listita.add("Barco, "+Integer.toString(cont1)+" destruido");

listita.add("Jugador 2 sigue atacando");

if(cont1==5)

{

JOptionPane.showMessageDialog(MiPanel1.getParent(), "Jugador 2 Ha Ganado, Presiona la carita para un
juego nuevo");

MiPanel1.disable();

MiPanel.disable();

}

}

else

{

B1.setBackground(Color.blue);

listita.add("Tu turno Jugador 1");
```

```
MiPanel.disable();

MiPanel1.enable();

}

}
```

4.7 Clase Adapter.

Es una forma abreviada para la interfaz window Listener.

```
addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent we)

{

dispose();

System.exit(0);

}

});
```

Conclusion: es una forma resumida de las interfaces.

Ejemplo:

```
class MiBoton extends JButton implements ActionListener {

int jug=0;

public MiBoton()

{

addActionListener(this);

setBackground(Color.gray);

setVisible(true);

}

public void actionPerformed(ActionEvent e){

if(Gato.Turno==1){

setIcon(cruz);
```

```
jug=1;
Turno=2;
enable(false);
ganador();
}
else{
setIcon(cir);
jug=2;
Turno=1;
enable(false);
ganador();
}
}
}
```

Elaborar un programa que capture palabras y muestre si es palíndroma o no:

```
import java.awt.*;
import java.awt.event.*;
import java.util.StringTokenizer;
import java.lang.String;
public class Palindromas extends Frame implements ActionListener
{
String el=new String();
TextField a,b;
Button boton, salida;
public Palindromas() {
```

```
setTitle("RFC del profe");
a=new TextField (30);
b=new TextField (10);
boton=new Button ("Palindromas");
salida=new Button ("Salir");
boton.addActionListener(this);
salida.addActionListener(this);
add(a); add(boton); add(b); add(salida);
setLayout(new FlowLayout());
setVisible(true);
setSize(400,400);
}
public void actionPerformed(ActionEvent p){
if (p.getSource()==boton)
{
StringBuffer sb=new StringBuffer();
StringTokenizer st= new StringTokenizer(a.getText(),",");
String ele=new String();
while (st.hasMoreTokens()){
ele=(String) st.nextToken();
el=new String(ele);
if (st.countTokens()==5)
{
el.reverse();
if(ele.equalsIgnoreCase(el))
ele="si";
```

```
else
ele="NO";
}
sb.insert(0,ele);
}
b.setText(sb.toString());
}
if (p.getSource()==salida)
{
System.exit(0);
}
}
public static void main(String arg[]){
Palindromas r=new Palindromas();
}
}
```

UNIDAD V: UTILERÍAS DE JAVA

- **Arrays**

Declaración:

Un array se decrara de la siguiente manera:

Int a[]; // Los arrays declarados de esta manera no se especifica el número de valores que contendra el vector.

Int b[] ={0,0,0,0}; // Se inicializan los valores del array con ceros.

- **Matrices**

Declaracion:

Una matriz se declara de la siguiente manera:

int vs[ ][ ];

- **Collecciones (Collections Framework)**

Provee un conjunto de clases de intefaces para almacenamiento y manipulacion de grupos de datos(conocidas como colecciones)

Inteface	Implementaciones					Historia
Set	HashSet		TreeSet			
List		ArrayList		LinkedList	Vector	Stack
Map	HashMap		TreeMap		HashTable	properties

Diferencias:

- Set.− Colecciones que no permiten dupicados
- List.− Colecciones que permiten duplicados y tienen indice
- Map.− Es un objeto que mapea (Relaciona una llave a un valor (key−value)), no permite llaves duplicadas.

Ejemplo 1: .../LUISF/Colección.java

```
import java.util.*;

class Coleccion{

public static void main (String g[]){

LinkedList n=new LinkedList();

n.add(new Integer('5')); //almacena unicamente objetos al final de la lista

n.addFirst(new Float("3.1")); // agrega un objeto al inicio

n.addLast(new Float("2.1")); // agrega un objeto al final

//n.remove(2); // elimina el obj. que se encuentre en el lugar 2

//n.removeFirst(); //elimina el obj. de la primera posición

Iterator itr=n.iterator(); // instancia una interface

while (itr.hasNext()){

System.out.print(itr.next());

} //imprime 3.1 2 2.1

}

}
```

En el ejemplo anterior solo se asignaron varios objetos de tipo Float a un LinkedList y se desplegaron en pantalla.

Ejemplo 2: /LUISF /Alumno.java

El siguiente programa crea una colección que guarda objetos que almacenan el nombre, edad y el id

```
import java.util.*;

class Alumno{

LinkedList r=new LinkedList();

Alumno(){

r.add(new Persona("Luis",19,3000));

r.add(new Persona("Fer",19,3003));

Iterator itr=r.iterator();

while (itr.hasNext()){

Persona a=new Persona("",0,0);

a=(Persona)itr.next();

System.out.println(a.NSS+" "+a.ed+" "+a.nom);

} // imprime los valores del LinkedList

}

public static void main (String g[]){

new Alumno();

}

}

class Persona{

int NSS=3000;

int ed=19;

String nom="Luis";

Persona(String nom1, int ed1, int NSS1){

ed=ed1;
```

NSS=NSS1;

nom=nom1;

}

}

UNIDAD VI: IO, BASE DE DATOS

SQL (Structured Query Language)

SQL Es un lenguaje estandar para acceder y manipular bases de datos.

Ejemplo 1:

Create table alumno(ncon int, nom varchar(30), ape_pat varchar(15), ape_mat varchar(15), edad int, carrera varchar(15));

Select * from alumno;

Select ncon, nom from alumno;

Select ncon, nom from alumo where edad>21;

Insert into alumo (ncon,nom,...,carrera)values(200,'erik','Garcia',...,'ISC');

//en java en la parte de values es: values('+t1.gettext()+'**,**+ t2.gettext()+

NOTA : todo lo que es de lenguaje SQL va entre comillas dobles (como cadena).

Delete from alumno where carrera=LI;

update calificaciones set calificación=10 where nom_materia=Economia and ncon=89;

// realiza un cambio a la calificacion asignandole 10 en donde el nombre de la materia sea //Economia y el numero de control sea 89

select * from alumnos order by nom desc; //ordena por nombre descendientemente

select a.ncon **,** a.nom **,** b.nom_matera**,** b.calificacion from alumnos a**,** calificacion b where a.ncon=b.ncon;

// Asigna un alias a las tablas **a = alumnos b = calificación** y de esta manera es más fácil //hacer asignaciones y selecciones

Reglas para acceder

- Define un URL(Unit Resourse Locations) guardandolo en un String
- Utilizas los drivers registrados en windows con la clase class
- damos de alta una coneccion con la clase driver Manager y de paramentro el URL
- Creamos un objeto statement con una instancia de la conexión
- Creamos objeto resourseSet con una instancia del objetos statement

- Imprimimos el objeto resultSet con una especie de Iterator
- Cerramos las conexiones.

Ejemplo1: ../LUISF/Dtbase.java

En el siguiente ejemplo veremos como acceder a una base de datos llamada bd1 que también se encuentra el la ruta indicada. El programa abre a la tabla y toma valores como el nombre, numero de control y el apellido paterno y lo imprime el una ventana.

NOTA: debe agregar a la tabla en el ODBC (En las herramientas Administrativas del panel de control)

```
import java.awt.*;

import java.awt.event.*;

import java.sql.*;

public class Dtbase extends Frame {

TextArea datos=new TextArea(3,12);

public Dtbase(){

datos=new TextArea(3,12);

setLayout(null);

setSize(500,500);

datos.setBounds(100,100,200,200);

datos.setColumns(3);

datos.setRows(3);

add(datos);

setVisible(true);

}

public static void main(String g[]){

Dtbase b=new Dtbase();

String url="jdbc:odbc:bd1";

try{Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

}catch(ClassNotFoundException e){

System.out.print("clase no encontrada");
```

```
System.err.println(e.getMessage());};

try{Connection con=DriverManager.getConnection(url,"","");

Statement stmt=con.createStatement();

ResultSet results=stmt.executeQuery("select * from alumnos;");

while(results.next())

{String n=results.getString("ncon");

String no=results.getString("nom");

String ap=results.getString("ape_pat");

System.out.print("reg\t"+n+" "+no+" "+ap+"\n");

b.datos.append("reg\t"+n+" "+no+" "+ap+"\n");

}

stmt.close();

con.close();

} catch(SQLException ex) { System.err.print(ex.getMessage());}

}

}
```

Ejemplo 2:

- **Dar de alta**

```
stmt.executeUpdate("insert into
calificaciones(ncon,Nom_materia,Calificacion)values("+nd+",'"+m+"',"+clf+");");
```

- **Dar de baja**

```
stmt.executeUpdate("Delete from alumnos where ncon ="+nct1.toString()+";");
```

- **Realizar cambios**

```
stmt.executeUpdate("update calificaciones set Nom_materia='"+m+"',Calificacion="+clf+" where
ncon="+nd+" and Nom_materia='"+nm1+"';");
```

- **Mostrar datos**

```
ResultSet results1=stmt1.executeQuery("select * from alumnos order by nom desc;");
```

```
While( results1.next()){

nc=results1.getString("ncon");

System.out.print("numero de control : \t"+nc+ "\n");

}
```

NOTA: un ejemplo completo sobre base de datos se encuentra en la siguiente ruta **../LUISF/Nopple.java** el password es **a w q s**; recuerde que debe agregar a la tabla en el ODBC (En las herramientas Administrativas del panel de control)

GRAFICAR:

```
//**********************************GRAFICAR CON LÍNEAS**************************



class Graph extends Frame {

Label n=new Label();

int val=10;;

Integer n1;

int x1,y1,x2,y2;

String econ=new String();

String ca=new String();

public void paint(Graphics g) {

int x,y,z,w;

x=45; y=50;

g.setColor(Color.red);

z=0;

val=10;

while (z<10){

n1=new Integer(val);

g.drawLine(x,y,x+10,y);

g.drawString(n1.toString(),x−10,y);
```

```
y=y+40;
val=val−1;
z=z+1;
}
g.drawLine(50,50,50,450);
g.drawLine(50,450,600,450);
String url="jdbc:odbc:bd1";
try{Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
}catch(ClassNotFoundException e){
System.out.print("clase no encontrada");
System.err.println(e.getMessage());};
try{Connection con=DriverManager.getConnection(url,"","");
Statement stmt=con.createStatement();
Statement stmt1=con.createStatement();
ResultSet results=stmt.executeQuery("select * from calificaciones where Nom_materia='"+econ+"';");
x=90;
y=445;
x1=50; x2=50; y1=450; y2=450;
while(results.next())
{String n2=results.getString("ncon");
String mat=results.getString("Nom_materia");
String cal=results.getString("Calificacion");
g.drawString(n2,x,y+20);
g.drawLine(x,y,x,y+10);
x=x+40;
Integer v=new Integer(cal);
```

```
switch(v.parseInt(cal)){
case 1:x1=x2; y1=y2; x2=x2+40; y2=450−(1*40); ca="1";break;
case 2:x1=x2; y1=y2; x2=x2+40; y2=450−(2*40); ca="2";break;
case 3:x1=x2; y1=y2; x2=x2+40; y2=450−(3*40); ca="3";break;
case 4:x1=x2; y1=y2; x2=x2+40; y2=450−(4*40); ca="4";break;
case 5:x1=x2; y1=y2; x2=x2+40; y2=450−(5*40); ca="5";break;
case 6:x1=x2; y1=y2; x2=x2+40; y2=450−(6*40); ca="6";break;
case 7:x1=x2; y1=y2; x2=x2+40; y2=450−(7*40); ca="7";break;
case 8:x1=x2; y1=y2; x2=x2+40; y2=450−(8*40); ca="8";break;
case 9:x1=x2; y1=y2; x2=x2+40; y2=450−(9*40); ca="9";break;
case 10:x1=x2; y1=y2; x2=x2+40; y2=450−(10*40);ca="10";break;
}
g.setColor(Color.white);
g.fillOval(x1,y1,10,10);
g.fillOval(x2,y2,10,10);
g.drawString(ca,x2,y2−10);
g.setColor(Color.blue);
g.drawLine(x1,y1,x2,y2);
g.setColor(Color.red);
}
stmt.close();
con.close();
} catch(SQLException ex) { System.err.print(ex.getMessage());}
}
Graph(String nom){
addWindowListener(new WindowAdapter(){ public void windowClosing(WindowEvent w)
```

```
{dispose(); }});

econ=nom;

setBackground(Color.black);

setLayout(null);

setSize(700,500);

setVisible(true);

}

}
```

El metodo paint(Graphic g) {} es el que realiza que las imágenes se muestren en la pantalla.

Este ejemplo se encuentra en el programa Nopple en la direccion **../LUISF/Nopple.java**

Nota: Otro ejemplo de Bases de Datos es el examen, se encarga de registrar las preguntas y a los alumnos para que hagan un examen, registra su promedio y lo muestra en un reporte. El password es **a w q s.** recuerda dar de alta la tabla **exm.mdb** la ruta del programa es **../LUISF/Exxam.java**

Importante: Si desea abrir todos los ejemplos del semestre abra el workspace *luis.*

UNIDAD VII APPLETS

Ejemplos:

```
import java.applet.Applet;

import java.awt.Graphics;

public class hola extends Applet{

public void paint(Graphics g){

g.drawstring(hola,50,25);

}

}

<HTML><HEAD><TITLE> hola </TITLE></HEAD>

<BODY>

<APPLET CODE = hola.class WIDTH = 150 HEIGHT = 25>
```

```
</APPLET>
</BODY>
</HTML>
import java.applet.Applet;
import java.awt.Graphics;
class Simple extends Applet{
StringBuffer buffer;
public void init(){
buffer=new StringBuffer();
addItem(inicio);
}
public void start(){
addItem(Arrancando);
}
public void destroy(){
addItem(Preparando descarga);
}
public void stop(){
addItem(Deteniendo);
}
void AddItem (String newWord){
buffer.append(newWord);
repaint();
}
public void paint(Grapchis g){
g.drawRect(0,0,size().width−1,size().height−1;
```

g.drawstring(buffer.toString(),5,15);

}

}

UNIDAD VIII SERVLETS

MÉTODOS DEL SERVLET

SERVICE()

Ejecuta respuestas y acepta objetos como parámetros.

Parámetros:

ServletRequest: Una petición del cliente.

ServletResponse: Devuelve información al cliente.

INIT()

Es el lugar donde se inicializa el servlet y acepta como parámetro el ServletConfig y garantiza que solamente se llamará una vez durante toda la vida del servlet.

DESTROY()

getServletConfig

getServletContext Informativos

LOG()

Envía mensajes a la bitácora del motor Servlet

HTTP()

Protocolo orientado a petición −respuesta.

Donde el método service lanza peticiones a otros métodos como: Get, Head, Put, Post, Delete, Trace, Option.

Ejemplo:

import java.io.*

import javaxservlet.*;

import javax.servlet.http.*;

class hola extends HttpServlet{

public void doGet(HttpServletRequest request, HttpServletRespones response)

throws ServletException, IOException{

printwriter out= response.getWriter();

out.println(hola);

}

}

DESPLIEGUE DE APLICACIONES WEB

La implementación de referencia Java utiliza un formato XML personalizable que se guarda en el archivo web.xml (XML= extensible markup language).

APLICACIÓN WEB

Es una conexión de recursos que trabajan en colaboración, dirigidos a un área común del espacio de nombres de servidor web (servlets, jsp, html, imágenes, archivos de clase, datos de configuración).

El área común del espacio de servidor web es el disco duro.

ESTRUCTURA DE DIRECTORIOS DE UNA APLICACIÓN

(ESPACIO DE NOMBRES)

%Catalina_home%\webapps\app1

application root − contenido aplicación

otras carpetas de contenido, ejemplo: imágenes

web.inf − va a ser invisible a los usuarios de la aplicación

classes − clases compiladas de los servlets o package

lib − es similar a classes pero contiene archivos .jar que se ponen automáticamente a disposición del cargador de clases.

Tlds − son destructores de bibliotecas de etiquetas.

Web.xml − es el descriptor de despliegue, es independiente de las marcas comerciales y se usa para configurar servlet.

Web.xml se utiliza para alias de servlets, asignaciones (mapping) para definir límites de tiempo de espera en sesiones, define parámetros globales, define configuración de seguridad.

Applet

Container

Window

Panel

COMPONENT

Text Component

TextField

TextArea

Label

Button

CheckBox

Scrall Bar

Choise

List

Canvas

Menu

Frame

Dialog

FileDialog