



# Guía de Estudio

## Control 4

Profesor: Andrés Muñoz

Auxiliares: Agustín Almonte F.

Patricio Salles D.

Marcelo Muñoz V.

---

---



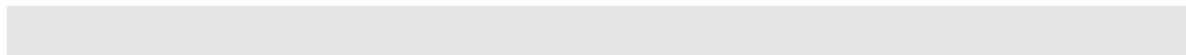
CC10A - 2004

## Tabla de Contenidos

TABLA DE CONTENIDOS	2
1. MILITARES EN BASES DE DATOS	4
2. NOTAS DEL CONTROL 4 EN BASES DE DATOS	6
3. TRAPÉCIOS Y SIMPSON	11
4. CÁLCULO DE PI	13
5. BÚSQUEDA BINARIA Y NEWTON-RAPHSON	14
6. POLINOMIO	15
7. POSTULACIÓN 2005	19
8. CINEMATOGRAFÍA.	20
9. APROXIMACIÓN DE SENOS.	21
10. NÚMEROS AZAROSOS.	23
11. OPERACIONES DE POLINOMIOS.	25
12. EL LUCA BAR	26
13. MAS SQL	28
14. JDBC CON IG.	30
15. VOLUMEN NÚMÉRICO	35
16. APROXIMACIÓN DE FUNCIONES.	37
17. RAÍCES CON MÉTODO DE LA SECANTE.	38
18. AREA ENTRE FUNCIONES.	40
19. AEROLINEAS BEAUCHEFF (P1C4-2001)	42
20. BUSQUEDA EN RAF	47
21. MANEJANDO RAF	49

**Guía de Estudio: Control 4**  
**CC10A 2004**

<u>21. ADMINISTRACIÓN CON RAF.</u>	<u>51</u>
<u>22. MEDIA NARANJA RAF.</u>	<u>53</u>



## 1. Militares en Bases de Datos

[Gentileza del profesor Kurt Schwarze]

Se tienen las siguientes tablas (con datos):

Personal		
id	nombre	id_rango
0	Juan	1
1	Pedro	2
2	Luis	1
3	Mario	0

Rangos		
id	nombre	sueldo
0	Conscripto	50
1	Capitán	100
2	General	200

Asignación	
id_pers	donde
0	Putre
1	Santiago
3	Coyahique
2	Vallenar

(id\_pers tanto en la tabla Asignación corresponden al código del individuo de la tabla Personal)

(a) Responda vía SQL:

1. Cuál es el rango de Juan

```
SELECT r.nombre
FROM rango r, personal p
WHERE r.id = p.id_rango
AND p.nombre = 'Juan'
```

2. Cuál es el sueldo de Pedro

```
SELECT r.sueldo
FROM rango r, personal p
WHERE r.id = p.id_rango
AND p.nombre = 'Pedro'
```

3. El nombre de todos los conscriptos asignados a Putre

```
SELECT p.nombre
FROM personal p, asignacion a
WHERE p.id = a.id_pers
AND a.donde = 'Putre'
```

4. El nombre del militar de mayor Rango

```
SELECT nombre
FROM personal
WHERE id_rango = ( SELECT max(id)
                  FROM rango )
```

(b) Indique qué datos lanzarían la siguiente consulta SQL

```
SELECT r.nombre, p.nombre
FROM personal p, rangos r
WHERE p.id_rango = r.id AND r.sueldo > 99
```

## Guía de Estudio: Control 4

### CC10A 2004

Capitán	Juan
Capitán	Luis
General	Pedro

(c) Considere que existe el método **ResultSet ejecuta(String SQL)** que permite obtener un **ResultSet** con los valores de una consulta SQL. Escriba un programa que escriba en la pantalla los resultados de la consulta de la parte (b) de la siguiente forma:

[RANGO] - [NOMBRE]

```
// Primero obtenemos el ResultSet
ResultSet rs = ejecuta("SELECT r.nombre, p.nombre "
                      + " FROM personal p, rangos r "
                      + " WHERE p.id_rango = r.id "
                      + " AND r.sueldo > 99");

// Ahora hacemos el ciclo de despliegue
while (rs.next()) {
    String rango = rs.getString("r.nombre");
    String nombre = rs.getString("p.nombre");

    System.out.println(rango + " - " + nombre);
}
```

## 2. Notas del Control 4 en Bases de Datos

Para el control 4 se requiere que realices un programa que permita a los auxiliares guardar las notas de la sección en una Base de Datos access. Considere el siguiente modelo:



Para ello se te pide:

- (a) Escribe un método que escriba en la base de datos `jdbc:odbc:control` los datos ingresados por parámetro en la Tabla **Alumno**, y que retorne verdadero si lo hizo y falso si no:

**public boolean insertarAlumno (String nombre, String seccion, int codigo)**

```
public boolean insertarAlumno (String nombre, String seccion,
                               int codigo) {
    try {
        // Cargar Driver
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    }
    catch (ClassNotFoundException e) {
        return false;
    }

    try {
        // Abrir conexión
        Connection c =
            DriverManager.getConnection(
                "jdbc:odbc:control", "", "");

        // Crear el paquete
        Statement stmt = c.createStatement();

        // Crea la consulta
        String sql = "INSERT INTO alumno "
            + "VALUES (" + codigo + ", "
            + nombre + ", "
            + seccion + ")";

        // Ejecuta la consulta
        stmt.executeUpdate(sql);

        // Cierra conexión
        c.close();
    }
    catch (SQLException e) {
        return false;
    }

    return true
}
```

- (b) Escribe un método que escriba en la misma base de datos, en la tabla **Nota**, los datos de la nota que el alumno obtiene en cada pregunta:

**public boolean insertarNota (int codigo, int pregunta, double nota)**

```
public boolean insertarNota (int codigo, int pregunta,
                             double nota) {
    try {
        // Cargar Driver
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    }
    catch (ClassNotFoundException e) {
        return false;
    }

    try {
        // Abrir conexión
        Connection c =
            DriverManager.getConnection(
                "jdbc:odbc:control", "", "");

        // Crear el paquete
        Statement stmt = c.createStatement();

        // Primero se verifica de que exista el alumno
        // en la tabla alumno
        ResultSet rs = stmt.executeQuery(
            "SELECT * "
            + " FROM alumno "
            + " WHERE codigo = " + codigo);
        if (!rs.next()) // El ResultSet no trae datos
            return false;

        // Crea la consulta
        String sql = "INSERT INTO nota "
            + " VALUES (" + codigo + ", "
            + "                + pregunta + ", "
            + "                + nota + ")";

        // Ejecuta la consulta
        stmt.executeUpdate(sql);

        // Cierra conexión
        c.close();
    }
    catch (SQLException e) {
        return false;
    }

    return true
}
```

- (c) Se necesita que escribas en SQL la consulta que le permitiría a los ayudantes separar las pruebas de los alumnos de la sección 03. Recuerda que las pruebas solo tienen el código del alumno y los ayudantes deben conocer la pregunta y la nota que posee la prueba.

```
SELECT cod_alumno, pregunta, nota
FROM nota, alumno
WHERE codigo = cod_alumno
AND seccion = 3
```

**Nota:** Ojo que en este caso no estamos utilizando sinónimos porque los nombres de las columnas son todos distintos entre si. No son necesarios.

(d) Escribe el SQL que promediará las 3 preguntas para cada alumno.

```
SELECT a.nombre, a.seccion,
       (p1.nota + p2.nota + p3.nota) / 3
FROM alumno a, nota p1, nota p2, nota p3
WHERE a.codigo = p1.cod_alumno
      AND p1.pregunta = 1
      AND a.codigo = p2.cod_alumno
      AND p2.pregunta = 2
      AND a.codigo = p3.cod_alumno
      AND p3.pregunta = 3
```

**Nota:** Acá si utilizamos sinónimos para obtener las 3 notas por separado.

(e) Escribe una interfaz de texto que permita el ingreso de las notas utilizando el método **insertarNota**. Esto ayudaría a los ayudantes a ingresar las notas de las pruebas a una Base de Datos para publicar sus resultados.

```
// Será realizado en un método
public void llenarNotas () {
    BufferedReader IN = new BufferedReader (
        new InputStreamReader (System.in));

    System.out.println("Ingrese las notas");
    System.out.println("Termine ingresando el código 0");
    while (true) {
        System.out.print(" Código del Alumno?");
        int codigo = Integer.parseInt(IN.readLine());
        if (codigo == 0) break;

        System.out.print(" Pregunta?");
        int pregunta = Integer.parseInt(IN.readLine());

        System.out.print(" Nota?");
        double nota = new Double (IN.readLine()).doubleValue();

        // Insertamos la nota
        if (!insertarNota(codigo, pregunta, nota))
            System.out.println("No se pudo insertar");
    }

    System.out.println("Fin del Ingreso de Notas");
    IN.close();
}
```

(f) Escribe un programa que permita desplegar una tabla con los resultados del control. Puede suponer que existe el método:

**public ResultSet consultar(String SQL)**



que permite al programa ejecutar en la base de datos **control** una consulta pasada por parámetro y retornar sus resultados en formato de **ResultSet**.

La salida debe ser:

Nombre	P1	P2	P3	Promedio
Juan Pérez	3.5	4.5	7.0	5.0
María Conejo	1.6	1.3	3.3	3.1
...				

**Nota:** Considere que el largo del nombre es máximo 50 caracteres. Para grabar la información, utilice un archivo de texto como salida y su nombre debe ser **notas\_xx** en donde **xx** corresponde a la sección.

```
// Será realizado en un método
public void publicarNotas () {
    BufferedReader IN = new BufferedReader (
        new InputStreamReader (System.in));

    System.out.println("Notas del Control 4");
    System.out.println("=====");
    for (int seccion=1; seccion<=6; seccion++) {
        System.out.print("Sección 0" + seccion);

        BufferedWriter OUT = new BufferedWriter (
            new PrintWriter("notas_0" + seccion));

        ResultSet rs = ejecutar(
            "SELECT a.nombre, "
            + "      p1.nota, p2.nota, p3.nota, "
            + "      (p1.nota + p2.nota + p3.nota)/3 prom"
            + " FROM alumno a, nota p1, nota p2, nota p3"
            + " WHERE a.codigo = p1.cod_alumno"
            + " AND p1.pregunta = 1"
            + " AND a.codigo = p2.cod_alumno"
            + " AND p2.pregunta = 2"
            + " AND a.codigo = p3.cod_alumno"
            + " AND p3.pregunta = 3"
            + " AND a.seccion = " + seccion
            + " ORDER BY a.nombre";

        OUT.println("Nombre      P1      P2      P3      Promedio");

        while (rs.next()) {
            String nombre = rs.getString("a.nombre");
            alargarString(nombre, 50);
            double p1 = rs.getDouble("p1.nota");
            double p2 = rs.getDouble("p2.nota");
            double p3 = rs.getDouble("p3.nota");
            double prom = rs.getDouble("prom");

            OUT.println(nombre + " " + p1 + " " + p2 + " "
                + p3 + " " + prom);
        }

        OUT.close();

        System.out.println("... Ok!");
    }
}
```

```
        System.out.println("Las notas han sido publicada");
        IN.close();
    }

    // El que agranda un String
    public void alargarString(String s, int n) {
        if (s.length() > n)
            s = s.substring(0, n);
        for (int i=s.length(); i<n; i++)
            s = s + " ";
    }
}
```

En la consulta SQL dice:

```
...
+ "          (p1.nota + p2.nota + p3.nota)/3 prom"
...
```

En el caso de este nombre **prom** que le ponemos a la columna es para que después podamos utilizar eso como nombre de la columna directamente para obtenerlos desde el ResultSet. Funciona igual como funcionan los sinónimos para las tablas.

### 3. Trapecios y Simpson

Se desea comparar el método de los trapecios y el método de Simpson para evaluar áreas abajo de una curva. La idea es evaluar el valor de `Math.sin()` en el intervalo  $[0, \pi]$ , utilizando los 2 algoritmos vistos en clases, e implementados en clase auxiliar, para las siguientes mallas:  $h = 0.1, 0.5$  y  $1.0$ .

Para esto usted debe crear el siguiente método:

```
static public void area_sin(double inicio, double fin, double h)
{ ... }
```

Que imprime en la salida estándar el área del método seno calculada usando:

- (a) El método de Simpson en los intervalos  $[\text{inicio}, \text{fin}]$
- (b) El método de los trapecios en el intervalo  $[\text{inicio}, \text{fin}]$
- (c) El resultado usando calculo tradicional. Donde  $\text{integral}(\text{seno}) = -\text{coseno}$

```
class Area {

    static public double trapecios(double y[], double h){

        int n=y.length;
        double suma=(y[0]+y[n-1])/2;

        for(int i=0;i<n-1;i++){
            suma += y[i];
        }

        return suma*h;
    }

    static public double Simpson(double y[], double h){

        int n=y.length;
        double pares=0, impares=0;

        for(int i=1; i<n-1; i+=2){
            impares += y[i];
            pares += y[i+1];
        }

        return h*(y[0]+4*pares+2*impares+y[n-1])/3;
    }

    static public void area_sin(double inicio, double fin,
                                double h){

        double[] a;
        double n= (fin-inicio)/h;
        a=new double[(int)n];

        // lo primero es llenar el arreglo
        for(int i=0;i<(int)n;i++)
            a[i]=Math.sin(inicio+(i*h));
    }
}
```

```
        //ahora imprimimos los resultados

        System.out.println("*****");
        System.out.println("Inicio:" + inicio + "    Fin:" +
            fin + "    h:" + h + "\n");
        System.out.println("Simpson:"+Simpson(a,h));
        System.out.println("Trapezio:"+trapecios(a,h));
        System.out.println("Integrando"+
            (Math.cos(inicio)-Math.cos(fin)));

    }

    static public void main(String args[]){

        area_sin(0, Math.PI, 0.01);
        area_sin(0, Math.PI, 0.1);
        area_sin(0, Math.PI, 0.5);
        area_sin(0, Math.PI, 1);

    }

}
```

## 4. Cálculo de PI

Se le pide crear un método que permita calcular a Pi con tantos decimales como se desee. Para esto usted debe usar la serie de expansión de Pi y programar el siguiente método:

```
static public double Pi(int n) { ... }
```

Que retorne el valor de Pi calculado con n decimales.

Para resolver el problema se recomienda crear un  $\epsilon = 10^{-n}$  y cuando  $Pi(i) - P(i-1) < \epsilon$  entonces la función retorne el valor de  $Pi(i)$ , donde (i) es Pi evaluado con i términos de la serie.

```
class Pi{

    static public void main(String args[]){

        System.out.println(Pi(5));

    }

    /* Usaremos la serie de liebnitz
    PI/4 = 1-1/3+1/5-1/7+1/9-1/11 ....
    En todo caso para los interesados en el tema esta serie
    es muy lenta. Revisen:
    http://webs.adam.es/rlllorens/PIvaldes.htm
    para ver otras. */

    static double PI(int n){
        double PI=0; //algun valor muy grande
        double PI_anterior=100; // valor chico comparado con PI
        double epsilon=Math.pow(10,-n);
        double signo=-1;

        for(int i=1; Math.abs(PI-PI_anterior)>=epsilon;i+=2){
            PI_anterior=PI;
            PI+= (signo/i);
            signo*=-1;
            System.out.println(i+": "+PI_anterior+": "+PI);
        }

        return PI*4;
    }

}
```

## 5. Búsqueda Binaria y Newton-Raphson

Se desea comparar la velocidad del método de búsqueda binaria y el método de Newton-Raphson. Para esto usted debe definir un epsilon lo suficientemente pequeño tal que para  $X_n - X_{(n-1)} < \epsilon$  => El algoritmo ha convergido a la solución. La idea es contar cuantas iteraciones (veces que se aplica el ciclo del algoritmo) toma cada uno de los algoritmos en converger a la solución. Para esto, usted debe implementar la función:

```
static public void comparacion(double X0, double Xder) { ... }
```

Y encontrar numericamente para que valor de  $x$ ,  $\text{Math.cos}(x)=0$  (Suponga que  $X_0 < X_{der}$ ). El método debe imprimir lo siguiente en `System.out()`:

- Cantidad de iteraciones del algoritmo de Newton-Raphson
- Cantidad de iteraciones del algoritmo de búsqueda binaria
- El valor encontrado para el cual  $\text{Math.cos}()$  es cero.

## 6. Polinomio

*[Gentileza del Profesor José Piquer]*

Veamos un ejemplo de un programa (que funciona) que juega con polinomios.

```
import java.io.*;
import java.lang.*;

public class Polinomio {

    /**
     * La clase polinomio representa a un Polinomio colocando sus
     * coeficientes en un arreglo donde el indice da el grado:
     *
     * Ej:  $2X^3 + x - 5$  ; quedaría representado por un arreglo a[] con
     * los siguientes valores: a[0] = -5, a[1] = 1, a[2] = 0; a[3] = 2
     */

    double[] cons;
    double delta = 0.00001;

    public Polinomio(double[] cons) {
        this.cons = cons;
    }

    /**
     * Esta función evalúa un polinomio para un X dado
     */

    public double eval(double x) {
        int i;
        double result=0;

        for(i=0;i<cons.length;i++)
            if (cons[i] != 0)
                result = result + cons[i]*Math.pow(x,i);

        return(result);
    }

    /**
     * Esta función calcula la derivada en un punto X
     */

    public double derivada(double x) {
        return((eval(x+delta)-eval(x))/delta);
    }

    /**
     * Esta función calcula un punto maximo o mínimo, o en otras
     * palabras, cuando la derivada se va a cero en un intervalo entre
     * x1 y x2. Realiza una búsqueda binaria.
     *
     * Para que funcione bien, debe asegurarse que tal punto existe en
     * el intervalo que se dan de parametros
     *
     * Esta es una version recursiva
     */
}
```

```
public double inflexion_rec(double x1, double x2) {
    double der1 = derivada(x1);
    double der2 = derivada(x2);
    double aux = x1;

    if (Math.abs(der1) < delta) return(x1);
    if (Math.abs(der2) < delta) return(x2);

    if ((der1<0 && der2<0) || (der1>0 && der2>0)) {
        System.err.println("error");
        java.lang.System.exit(-1);
    }

    if (x1 > x2) {
        x1 = x2;
        x2=aux;
    }

    aux = (x2+x1)/2;
    double der_aux = derivada(aux);
    if (der_aux < delta) return(aux);

    if ((der_aux<0 && der2<0) || (der_aux>0 && der2>0))
        return(inflexion(x1,aux));
    else
        return(inflexion(aux,x2));

}

/**
 * Esta función calcula un punto maximo o mínimo, o en otras
 * palabras, cuando la derivada se va a cero en un intervalo entre
 * x1 y x2. Realiza una busqueda binaria
 *
 * Para que funcione bien, debe asegurarse que tal punto existe en
 * el intervalo que se dan de parametros
 *
 * Solucion no recursiva
 */

public double inflexion(double x1, double x2) {
    double der1 = derivada(x1);
    double der2 = derivada(x2);
    double aux = x1;

    if (Math.abs(der1) < delta) return(x1);
    if (Math.abs(der2) < delta) return(x2);

    if ((der1<0 && der2<0) || (der1>0 && der2>0)) {
        System.err.println("error");
        java.lang.System.exit(-1);
    }

    if (x1 > x2) {
        x1 = x2;
        x2=aux;
    }

    aux = (x2+x1)/2;
    double der_aux = derivada(aux);
```



```
while (Math.abs(der_aux) > delta) {
    if ((der_aux<0 && der2<0) || (der_aux>0 && der2>0))
        x2 = aux;
    else
        x1 = aux;

    aux = (x2+x1)/2;
    der_aux = derivada(aux);
}
return(aux);
}

/**
 * Esta funcion clacula la integral en un bloque, utilizxando
 * triangulos en las puntas
 *
 * va avanzando desde x1 hasta x2 en pedazos delta
 */

public double integral(double x1, double x2) {
    double result=0;
    double aux = x1;

    if (x1 > x2) {
        x1 = x2;
        x2=aux;
    }

    while (aux < x2) {
        result = result +
            delta*(this.eval(aux)+this.eval(aux+delta))/2;
        aux += delta;
    }

    result = result - delta*(this.eval(x2)+this.eval(aux+delta))/2;
    return(result);
}

/**
 * Esta funcion calcula la integral en un bloque, utilizxando
 * triangulos en las puntas
 *
 * es recursiva, va dividiendo el bloque hasta que el ancho es
 * menor que delta
 */

public double integral_rec(double x1, double x2) {

    if (Math.abs(x1-x2) < delta)
        return ((x2-x1)*((this.eval(x1)+this.eval(x2))/2));
    else {
        double middle = (x1+x2)/2;
        return(integral_rec(x1,middle)+integral_rec(middle,x2));
    }
}

public static void main(String[] args) {
    // x^2+6x+14
    double[] cons = new double[3];
```

```
cons[0]=14;
cons[1]=6;
cons[2]=1;
Polinomio pol = new Polinomio(cons);
System.out.println("Evaluacion en 3: "+pol.eval(3));
System.out.println("Derivada en 3: "+pol.derivada(3));
System.out.println("Integral entre 2 y 5 Recursiva:"+
    pol.integral_rec(2,5));
System.out.println("Integral entre 2 y 5:"+pol.integral(2,5));
System.out.println("Inflexion entre -9 y 5:"+
    pol.inflexion(-9,5));
System.out.println("Inflexion entre -9 y 5 Recursivo:"+
    pol.inflexion_rec(-9,5));
}
}
```

## 7. Postulación 2005

Para el proceso de selección de alumnos a las universidades se dispone de una base de datos compuesta por las siguientes tablas:

Carreras		Alumnos		Postulaciones	
Columna	Tipo	Columna	Tipo	Columna	Tipo
codigo	String	cedula	String	cedulaAlumno	String
nombre	String	nombre	String	codigoCarrera	String
ponderador1	String	puntaje1	String		
ponderador2	String	puntaje2	String		
...	...	...	...		
ponderador7	String	puntaje7	String		

Escriba un programa en Java que muestre el nombre del postulante con el mejor puntaje ponderado para ingresar al Plan Común de Ingeniería (código 1520).

### Notas

- Las tres tablas se encuentran ordenadas por sus primeras columnas
- El puntaje ponderado se calcula como:  
$$\text{Puntaje1} * \text{Ponderador1} + \dots + \text{Puntaje7} * \text{Ponderador7}$$
- El programa puede escribirse utilizando JDBC (Java y SQL) o archivos (de texto o acceso directo) según la convención utilizada por su profesor

## 8. Cinematografía.

Una base de datos cinematográfica contiene las siguientes cuatro tablas (todas ordenadas por el identificador ubicado en su primera columna):

Películas		Artistas		Dirige		Actua	
Columna	Tipo	Columna	Tipo	Columna	Tipo	Columna	Tipo
id	String	id	String	idArtista	String	idArtista	String
titulo	String	nombre	String	idPelicula	String	idPelicula	String
pais	String	pais	String				

- Escriba un programa que, utilizando JDBC (Java+SQL), muestre el nombre del director chileno que ha dirigido más películas.
- Escriba un programa que, utilizando las clases y convenciones utilizadas por el profesor de su sección, muestre los nombres de las películas en que actúan conjuntamente Penélope Cruz y Antonio Banderas.

## 9. Aproximación de seno.

La función trigonométrica seno para un ángulo  $x$  (en radianes) se calcula con la siguiente serie:  
 $x - x^3 / 3! + x^5 / 5! - x^7 / 7! + \dots$

- escriba un método de encabezamiento `double seno(double x, int n)` que calcule el seno de un ángulo  $x$ , usando los primeros  $n$  términos de la serie anterior.
- Utilice el método anterior en un programa que muestre la tabla siguiente:

Angulo	tangente
0	0
$\pi/8$	...
$\pi/4$	...
...	...
$2\pi$	...

Notas.

- Recuerde que  $\text{tangente}(x)$  es  $\text{seno}(x)/\text{coseno}(x)$  y  $\text{coseno}(x)^2 + \text{seno}(x)^2 = 1$
- En caso que  $\text{coseno}(x)$  sea cero, se debe escribir tangente "infinito"
- Calcule la función seno con 10 términos ( $n=10$ )

## Solución:

```
public class Problema1 {
    public static double seno(double x, int n) {
        double senx = x;
        boolean sumar = false;
        for (int i = 1; i < n; ++i) {
            double aux = Math.pow(x, 2 * i + 1) / factorial(2 * i + 1);
            if (sumar) {
                senx = senx + aux;
            }
            else {
                senx = senx - aux;
            }
            sumar = !sumar;
        }
        return senx;
    }
    private static double factorial(int n) {
        double f = 1;
        for (int i = 1; i <= n; ++i) {
            f = f * i;
        }
        return f;
    }
    public static void main(String[] args) {
```

```
System.out.println("Angulo\t\t\t\t\ttangente");
double angulo = 0;
for (int i = 0; angulo <= 2 * Math.PI; ++i) {
    double sen = seno(angulo, 10);
    double cos = Math.sqrt(1 - sen * sen);
    String tangente;
    if (cos == 0) {
        tangente = "Infinito";
    }
    else {
        tangente = sen / cos + "";
    }
    System.out.println(angulo + "\t\t\t\t\t" + tangente);
    angulo = angulo + Math.PI / 8;
} } }
```

## 10. Números azarosos.

La clase Tabla permite mantener una lista de números reales a través de los siguientes métodos:

ejemplo	resultado	significado
T = new Tabla()	-	constructor que inicializa tabla con cero elementos
T.agregar(x)	void	agregar el N° real x a la Tabla T
T.estaVacia()	boolean	Devuelve true si T está vacía (y false si no)
T.extraerMayor()	double	Devuelve (y elimina) el mayor N° de la Tabla T
T.extraerMenor()	double	Devuelve (y elimina) el menor N° de la Tabla T

a) Escriba el método extraerMenor suponiendo la siguiente declaración de la clase Tabla:

```
class Tabla{
    private double[] numeros;
    private int n; //cantidad efectiva de elementos
    public Tabla(){ numeros=new double[100]; n=0; }
    ...
}
```

b) Utilice la clase anterior en un programa que genere un número entero n al azar entre 10 y 100, y a continuación genere n números reales al azar. Finalmente, escriba los 10 mayores y los 10 menores.

Nota. Recuerde que Math.random() genera un número al azar de tipo double en el intervalo [0,1[

## Solución:

Parte a:

```
public class Tabla {
    private double[] numeros;
    private int n;
    public Tabla() {
        numeros = new double[100];
        n = 0;
    }
    public void agregar(double x) {
        numeros[n] = x;
        ++n;
    }
    public boolean estaVacia() {
        return n == 0;
    }
    public double extraerMayor() {
        int indice_mayor = 0;
        for (int i = 0; i < n; i++) {
```

```
if (numeros[indice_mayor] < numeros[i]) {
    indice_mayor = i;
}
double mayor = numeros[indice_mayor];
--n;
numeros[indice_mayor] = numeros[n];
return mayor;
}

public double extraerMenor() {
    int indice_menor = 0;
    for (int i = 0; i < n; i++) {
        if (numeros[indice_menor] > numeros[i]) {
            indice_menor = i;
        }
    }
    double menor = numeros[indice_menor];
    --n;
    numeros[indice_menor] = numeros[n];
    return menor;
}
```

Parte b:

```
public static void main(String[] args) {
    int n = azarozo(10, 100);
    Tabla t1 = new Tabla();
    Tabla t2 = new Tabla();
    for (int i = 1; i <= n; ++i) {
        double azar = Math.random();
        t1.agregar(azar);
        t2.agregar(azar);
    }
    System.out.println("10 Mayores :");
    for (int i = 1; i <= 10; ++i) {
        System.out.println(t1.extraerMayor());
    }
    System.out.println("");
    System.out.println("10 Menores :");
    for (int i = 1; i <= 10; ++i) {
        System.out.println(t2.extraerMenor());
    }
}

/**
 * Metodo auxiliar que genera un numero al azar entre [n, m[
 */
public static int azarozo(int n, int m) {
    return (int) Math.floor(Math.random() * (m - n) + n);
}
}
```



## 11. Operaciones de Polinomios.

La siguiente clase permite realizar operaciones con Polinomios:

```
class P{
public P(double[]coef,int grado){...} //inicializar polinomio con
coef[0],...,coef[grado]
public double evaluar(double x){...} //entregar evaluación del
polinomio en argumento x
public int obtenerGrado(){...} //entregar grado de polinomio
public double obtenerRaiz(){...} //entregar una raíz del polinomio
public P dividir(double r){...} //entregar polinomio que resulta de
dividir el polinomio original
//por el polinomio (x-r), en que r debe ser una raíz
private int n; //grado del polinomio
private double[] a; //arreglo para n+1 coeficientes: a[0], ...,
a[n]
}
```

a)Escribir el método dividir.

Indicación. Recuerde que si  $r$  es una raíz entonces  $(a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1} + a_nx^n) / (x-r)$  es un polinomio de la forma  $b_0 + b_1x + \dots + b_{n-1}x^{n-1}$

Desarrollando, se obtienen las igualdades:  $a_0 = -r b_0$ ;  $a_1 = b_0 - r b_1$ ; ... ;  $a_{n-1} = b_{n-2} - r b_{n-1}$

Nótese que si  $r=0$ , las igualdades anteriores se reducen a  $a_1 = b_0$ , ... ,  $a_n = b_{n-1}$

b)Escribir un método que, utilizando los métodos de la clase anterior, construya y devuelva un arreglo con todas las raíces de un polinomio.

· Encabezamiento: double[] raices(P p)

· Indicación. Suponga que  $a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1} + a_nx^n$  se puede expresar como  $(x-r_1)(x-r_2)\dots(x-r_n)$

en que  $r_1, \dots, r_n$  son raíces reales del polinomio

c)Escribir un programa que, utilizando el método anterior, determine y escriba las 3 raíces de  $x^3 - 9x^2 + 26x - 24$

## 12. El Luca Bar

El "Luca Bar" mantiene una base de datos para la preparación de sus tragos con las siguientes tablas:

Tabla:	Tragos	Tabla:	Ingredientes		Tabla :	Recetas	
columna	ejemplo	columna	ejemplo1	ejemplo2	columna	ejemplo1	ejemplo2
codigo	T340	codigo	I765	I368	codigoTrago	T340	T340
nombre	Blue Monday	nombre	Vodka	Cointreau	codigoIngrediente	I765	I368
		alcohol	50°	35°	porcentaje	75%	25%

Escriba un programa en Java que encuentre el (o los) trago(s) con menor grado alcohólico. Por cada trago se debe mostrar el nombre y la receta para su preparación (por cada ingrediente una línea con el nombre y el porcentaje).

Notas.

- el grado alcohólico de un trago se calcula como el promedio ponderado de los grados alcohólicos de todos sus ingredientes. Por ejemplo el Blue Monday tiene el grado  $50 \times 75 + 35 \times 25$ .
- todas las columnas son Strings.
- En caso de no utilizar JDBC debe suponer que están disponibles las clases que permiten el acceso a las tablas de acuerdo a la convención utilizada por su profesor (suponiendo que las tablas están grabadas en archivos de acceso random y se encuentran ordenadas por su primera columna)

### Solución:

```
//declarar arreglos para codigos y nombres de tragos mas suaves: 0.5
int N=1000, n=0; //cantidad maxima y efectiva de tragos mas suaves
String []codigos=new String[N], nombres=new String[N];
//recorrer todos los tragos: 0.5
int menor=999; //0.1
ResultSet R=S.executeQuery("select * from Tragos"); //0.2
while (R.next()) //0.2
{
    //obtener ingredientes del trago: 1.0
    ResultSet R1=S.executeQuery( //0.25
        "select * from Recetas, Ingredientes "+ //0.25
        "where Recetas.codigoTrago='"+R.getString("Tragos.codigo") +//0.25
        "' and Recetas.codigoIngrediente = Ingredientes.codigo"); //0.25
    //calcular grado del trago: 1.0
    int grado=0; //0.1
    while (R1.next()) //0.2
        grado+= //0.2
            Integer.parseInt(R1.getString("Recetas.porcentaje"))* //0.25
            Integer.parseInt(R1.getString("Ingredientes.alcohol")); //0.25
    //mantener tragos mas suaves: 1.0
    if (grado<menor) { //0.1
        menor=grado; n=0; //0.2
    }
```

```
}
if (grado==menor) { //0.2
codigos[n]=R.getString("Tragos.codigo"); //0.2
nombres[n]=R.getString("Tragos.nombre"); //0.2
++n;
}
}
//mostrar recetas de tragos mas suaves : 2.0
for (int i=0; i<n; ++i) //0.25
{
System.out.println(nombres[i]); //0.25
ResultSet R1=S.executeQuery("select * from Recetas, Ingredientes"+
//0.25
"where Recetas.codigoTrago='"+codigos[i]+ //0.25
" and Recetas.codigoIngrediente = Ingredientes.codigo"); //0.25
while (R1.next()) //0.25
System.out.println(R1.getString("Ingredientes.nombre") + //0.25
R1.getString("Recetas.porcentaje")); //0.25
}
```

### 13. Mas SQL

Considerando la tabla EMPLEADO:

#### "Tablas de Datos y sus atributos"

##### EMPLEADO

NPILA	APPAT	APMAT	RUT	FNAC	DIRECCION	SEXO	SUELDO	RUTSUPERV	NDEPTC
-------	-------	-------	-----	------	-----------	------	--------	-----------	--------

##### DEPARTAMENTO

DNOMBRE	DNUMERO	RUTGERENTE	GERFECHAINIC
---------	---------	------------	--------------

##### UBICACIONES DEPTO

DNUMERO	DUBICACION
---------	------------

##### PROYECTO

PNOMBRE	PNUMERO	PUBICACION	DNUM
---------	---------	------------	------

##### TRABAJA EN

ERUT	PNO	HORAS
------	-----	-------

##### CARGA

ERUT	NOMBRE_CARGA	SEXO	FNAC	PARENTESCO
------	--------------	------	------	------------

- a) ¿Cómo devolvería una tabla con la fecha de nacimiento y dirección del empleado Jorge Burgos Monsalve?
- b) retorne el nombre y apellido paterno del empleado, y nombre y apellido del supervisor (=SUP).
- c)¿Cómo construiría la consulta, lista de todos los números de proyecto que involucran a un empleado de apellido "Perez", ya sea como trabajador, o gerente del departamento ? .
- d)Analizando las tablas, ¿como devolvería el número de sueldos distintos de la base de datos?.
- e)Entregue una tabla que muestre el número del proyecto, nombre del proyecto, y el número de empleados que trabajan en el proyecto, siempre y cuando en el trabajen más de dos empleados.
- f)Regrese una lista de empleados y los proyectos en los que trabajan, ordenada por departamento, y dentro de cada departamento, ordenada alfabéticamente por nombre y apellido.

g) Algunos tipos de consulta requieren que la tabla obtenida luego de la operación sea utilizada para una futura comparación, este tipo de operación se llama consulta anidada. Para ejemplificar lo anterior se le pide obtener una tabla que muestre el número del proyecto sólo si, en este trabaja Antonio Alcantara como gerente, ó si en el proyecto esta trabajando como empleado alguien de apellido Alcantara.

h) Las bases de datos, como elemento flexible de almacenamiento y consulta de datos, permiten insertar, borrar y actualizar datos. Para ejemplificar dichas operaciones, se le pide:

- a).- Insertar en la Tabla empleados sus propios datos (Imagine los que no se apliquen a su caso).
- b).- Ahora piense en un cambio que podría sufrir esta información (digamos que le cambiaron el sueldo), y realice esta actualización de los datos.
- c).- Como sabemos realmente no trabaja en esta empresa, le solicitamos ahora ejecutar la instrucción que borre sus datos de la BD.

## 14. JDBC con IG.

JDBC es básicamente, el protocolo usado para manejar bases de datos dentro de un contexto Java. Con JDBC es posible ejecutar sentencias SQL, tal como se ha aprendido anteriormente en el curso.

A continuación vamos a crear una pequeña aplicación grafica donde se verán los aspectos más centrales de una aplicación JDBC.

La aplicación consistirá en una subclase de Frame, cuyas variables de instancia serán las componentes graficas (botones, areas de texto, etc). En el constructor de la clase definiremos el LayoutManager, haremos los add's de los componentes, y registraremos los Listeners de la aplicación.

El comportamiento de la aplicación lo programaremos en el método actionPerformed.

Contamos con una base de datos llamada "example", la cual contiene las siguientes tablas:

**Tabla: DISTANCIAS**

NOMBRE	KM
Rancagua	84
...	...

**Tabla: COMBUSTIBLE**

TIPO	COSTO
Diesel	300
...	...

**Tabla: BUSES**

TRAYECTO	VALOR
200	80
...	...

**Tabla: AUTO**

TIPO	REND	VEL	TIPO_BENC
Familiar	7	90	97
...	...	...	...

## Guía de Estudio: Control 4

### CC10A 2004

---

Se le pide que realice la(s) consulta(s) apropiadas para obtener la siguiente información:

1. Distancia máxima que pueden llegar cada tipo de vehículo con 5.000 pesos de combustible.
2. Costo (por conceptos de combustible) de viajar en un camión hasta chillan.
3. Coeficiente entre el costo de viajar para cada vehículo hacia concepción y el costo en bus para el mismo trayecto.

### Solución:

1.

---

```
Select COSTO, REND
From  AUTO, COMBUSTIBLE
Where AUTO.TIPO_BENC = COMBUSTIBLE.TIPO
```

Resultado = (5000 / costo) \* rend

2.

---

```
Select COSTO, REND
From  AUTO, COMBUSTIBLE
Where AUTO.TIPO = 'camion'
And   AUTO.TIPO_BENC = COMBUSTIBLE.TIPO
```

```
Select KM
From  DISTANCIAS
Where NOMBRE = 'chillan'
```

Resultado = costo \* (km / rend)

3.

---

```
Select KM
From  DISTANCIAS
Where NOMBRE = 'concepcion'
```

```
Select COSTO, REND
From  AUTO, COMBUSTIBLE
Where AUTO.TIPO_BENC = COMBUSTIBLE.TIPO
```

```
Select VALOR
From  BUSES
```

Costo en auto = costo \* (km / rend)

Costo en bus = "costo mas cercano a km"

Resultado = Costo en auto / Costo en Bus

**Interfaz gráfica:**

```
import java.sql.*;
import java.awt.*;
import java.awt.event.*;

class JDBCExample extends Frame implements ActionListener{

    //Variables del Frame

    TextArea textBox=new TextArea(15,30);
    TextField command=new TextField("SELECT * from DISTANCIAS",25);
    Button quit=new Button("Quit");
    Button ok=new Button("Ejecutar \n Comando");
    BorderLayout b=new BorderLayout();

    //Variables necesarias para la consulta a la Base de datos

    Connection cx;
    Statement stmt;
    ResultSet rs;

    //esta ultima es opcional
    ResultSetMetaData rsmd;

    String driverOdbc="sun.jdbc.odbc.JdbcOdbcDriver";
    String tipoOdbc="odbc";

    public JDBCExample(){

        setLayout(b);
        setSize(400,400);
        if( ! cargaDriver(driverHsql) || !
hacerConexion(tipoHsql,"example","sa","")
            System.exit(0);

        //si pasa el if se cargo el driver y se abrio la conexion
        System.out.println("Driver Cargado, Base de datos Abierta");

        add("North",textBox);
        add("West",command);
        add("East",ok);
        add("South",quit);

        quit.addActionListener(this);
        ok.addActionListener(this);
        hacerConsulta("SELECT * from DISTANCIAS");

    }

    public void actionPerformed(ActionEvent e){

        if(e.getSource()==quit){
            terminaConexion();
            System.exit(0);
        }
    }
}
```



```
        }else if(e.getSource()==ok){
            hacerConsulta(command.getText());
        }
    }

    public void hacerConsulta(String sqlCommand){

        try{
            rs = stmt.executeQuery(sqlCommand);

            textBox.append("---\n");

            //loop que recorre las filas
            while (rs.next()) {

                rsmd = rs.getMetaData();
                int i = rsmd.getColumnCount();

                //loop que recorre las columnas de la actual fila
                for( int ndx = 1; ndx <= i; ndx++ ){
                    textBox.append(rs.getString(rsmd.getColumnName(
ndx ))+"\t");
                }
                textBox.append("\n");
            }
            textBox.append("---\n");

        }catch (java.sql.SQLException ex){
            System.out.println("Error en la consulta, error: "+ex);
        }
    }

    boolean cargaDriver(String name){

        try{
            Class.forName( name );
            return true;
        }catch (Exception ex) {
            System.out.println("Imposible Cargar el Driver "+name+"
error: "+ex);
            return false;
        }
    }

    boolean hacerConexion(String tipo, String database, String user,
String password){

        try{
            cx
DriverManager.getConnection("jdbc:"+tipo+": "+database, user,
password);
            stmt = cx.createStatement();
            return true;
        }catch (java.sql.SQLException ex){
            System.out.println("Problemas con la BD, error: "+ex);
            return false;
        }
    }
}
```

```
boolean terminaConexion(){  
    try{  
        cx.close();  
        System.out.println("Conexion a la BD Terminada");  
        return true;  
    }catch (Exception ex) {  
        System.out.println("Imposible Cerrar la Conexion, error:  
"+ex);  
        return false;  
    }  
}  
public static void main(String args[]){  
    JDBCExample k=new JDBCExample();  
    k.show();  
}
```

## 15. Volumen numérico

El cálculo del volumen de una superficie irregular, es un problema que difícilmente puede ser resuelto en forma exacta. Se le pide crear un método que permita dada una función de dos variables  $(x,y)$ , del tipo  $f(x,y)=z$ , conocida, siempre mayor que cero, conocer aproximadamente el volumen engendrado por dicha superficie, en una zona de coordenadas,  $[(a,b),(c,d)]$ , esto es, un rectángulo del cual se conocen sus puntas opuestas. Ver figura 1.

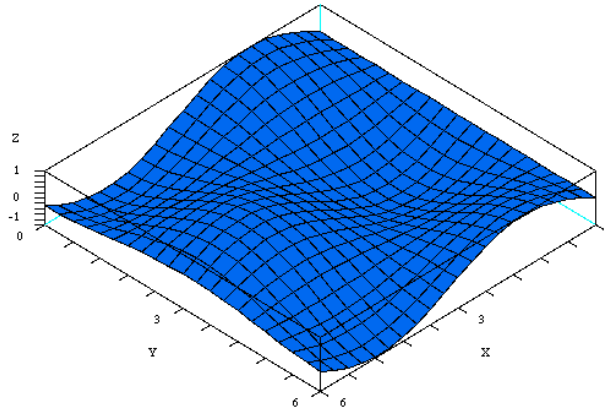


Figura 1.

### Solución:

```
import java.math.*;
class volumen {
    static double eval(double x, double y){
        //System.out.println("evaluacion:"+Math.sqrt(25-Math.pow(x,2)
        Math.pow(y,2)));
        return (Math.sin(x)*Math.cos(y));
    }

    static double area(double a,double b,int n,double y){
        double delta=(b-a)/n;
        double suma=(eval(a,y)+eval(b,y))/2;
        //System.out.println(suma);
        double x=a;
        for (int i=0;i<n-1;i++){
            x+=delta;
            suma+=eval(x,y);
        }
        return suma*delta;
    }

    static double volumen(double xa,double xb,double ya,double
    yb,int n){
        double delta=(yb-ya)/n;
        double suma=(area(xa,xb,n,ya)+area(xa,xb,n,yb))/2;
        //System.out.println(suma);
    }
}
```

```
        double y=ya;
        for (int i=0;i<n-1;i++){
            y+=delta;
            suma+=area(xa,xb,n,y);
        }
        return suma*delta;
    }
    public static void main(String arg[]){
        System.out.println(volumen(0,5,0,5,1000000));
    }
}
```

## 16. Aproximación de funciones.

En muchas ocasiones, no se conoce el valor de una función en un punto específico "a", pero si en algún otro, cercano a dicho valor. Al respecto y considerando que se tienen N pares de valores  $(x_i, y_i)$ , con  $y_i = f(x_i)$ ;  $x_1 < x_2 < \dots < x_n$ , y  $x_1 < a < x_n$ , es decir se tienen los valores  $x_i$  ordenados, y "a" pertenece al intervalo  $[x_1, x_n]$ . Con dicha información se le pide elaborar un método que tomando los dos puntos conocidos más cercanos y trazando la recta entre ellos, entregue un valor aproximado de  $f(a)$ .

*Encabezado del método.*

```
public double puntoDesconocido(double a, double[] x, double[] y){...}
```

### Solución:

```
public double puntoDesconocido(double a, double[] x, double[] y){  
  
    double resultado=-1;  
  
    /*  
    a es siempre > que el primero  
    y menor que el ultimo, se considera  
    el caso pudiera estar dentro de x[]  
    */  
    for(int i=1; i<x.length; i++){  
        if( x[i] >= a){  
            double m=(y[i]-y[i-1])/(x[i]-x[i-1]);  
            resultado=m*(a-x[i])+y[i];  
            break;  
        }  
    }  
  
    return resultado;    }
```

## 17. Raíces con método de la secante.

Para determinar una raíz de una función continua se puede utilizar el método de la secante. El algoritmo comprende las siguientes etapas:

- Determinar dos valores  $x_1$  y  $x_2$  tales que  $y_1 = f(x_1)$  e  $y_2 = f(x_2)$  tengan distinto signo.
- Calcular como aproximación de la raíz el punto  $x$  en que la recta entre los puntos  $(x_1, y_1)$  y  $(x_2, y_2)$  corta el eje horizontal (ver figura 2).

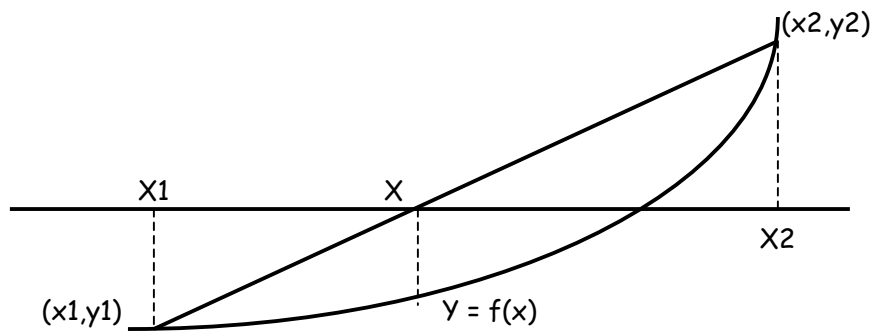


Figura 2

- Si  $y = f(x)$  tiene el mismo signo que  $y_1$ , se descarta  $y_1$ , si no se descarta  $y_2$ .
- Se repite el proceso hasta que el intervalo se acorte tanto como se quiera.

Al respecto, escriba una función que calcule la raíz de una función  $f$  y que tenga el encabezamiento

*public double raiz(double x1, double x2, double epsilon){....}*

- Escríbala en forma iterativa.
- Escríbala en forma recursiva.

## Solución

```
public class Raices {  
  
    static public double f(double x){  
        return x*x - 2*x - 2;  
    }  
  
    static public int signo(double x){  
        if(x<0)
```

```
        return -1;
    else if(x>0)
        return 1;
    else
        return 0;
}

//método de búsqueda binaria iterativo
static public double raiz(double a, double b, double epsilon){
    while(true){
        double x = (a+b)/2;
        if(b-a <= epsilon)
            return x;
        if(signo(f(x)) == signo(f(a)))
            a = x;
        else
            b = x;
    }
}

//método de búsqueda binario recursivo y genérico
static public double raiz(double a, double b, double epsilon,
Funcion f){
    double x = (a+b)/2;
    if(b-a <= epsilon)
        return x;
    else if(signo(f.valor(x)) == signo(f.valor(a)))
        return raiz(x, b, epsilon, f);
    else
        return raiz(a, x, epsilon, f);
}
}
```

## 18. Area entre funciones.

Dadas dos funciones  $f(x)$  y  $g(x)$  que se intersectan una sola vez en intervalo  $[a,b]$ , escriba un método que entregue una buena aproximación del área indicada en el siguiente ejemplo:

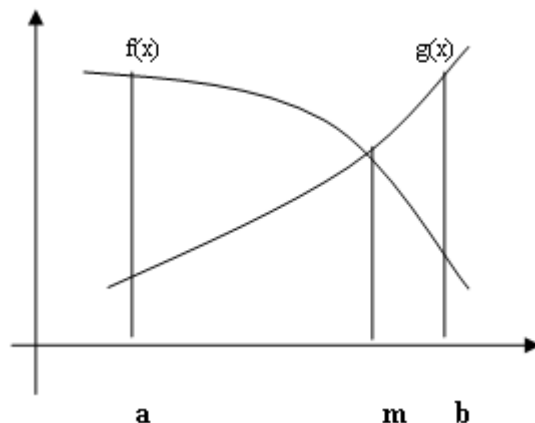


Figura 1.

### Notas:

- Invocación del método: `area(f, g, a, b, n)` en que  $n$  es el  $N^\circ$  total de puntos en que se dividirá el intervalo  $[a,b]$
- Nótese que en el punto  $m$  se cumple que  $f(x)-g(x)=0$
- Por supuesto, en otros ejemplos,  $g(x)$  podría estar por encima de  $f(x)$  antes del punto de intersección.

## Solución

```
//Control 4 2002: pregunta 3

/*
  La integral se calculó de varias maneras
  Puder ser, el máximo, punto medio, trapecio, también alguna
  Que se les ocurra, pero cuidado, si le piden la mayor
  exactitud usen
  Trapecio.
*/
//declarar tipo de funciones : 0.25
interface F { public double valor(double x); }

double area(F f, F g, double a, double b, int n) {
  //calcular valor de m (raiz de f-g) : 0.5

  double m=raiz(f,g,a,b,eps);
  //asegurar que f>g en [a,m]: 1.0
  if (f.valor(a)<g.valor(a)) { //0.5
    F aux=f; f=g; g=aux; //0.5
  }
  //calcular area en [a,m]: 1.0
  double factor=(m-a)/(b-a); //0.25
  int n1=(int)(n*factor); //0.25
```



```
double areal=integral(f,a,m,n1);//0.5

//calcular area en [m,b]: 0.75
n2=n-n1; //0.25
double area2=integral(g,m,b,n2);//0.5
//devolver area total: 0.25
return areal+area2;
}

//método para calcular raíz de f-g: 1.0
double raiz(F f, F g, double a, double b, double eps) { //0.1
    double x=(a+b)/2; //o calcular por secante //0.1
    if (b-a<=eps) return x; //0.1
    if (signo(f.valor(a)-g.valor(a)) == signo(f.valor(x)-
g.valor(x))) //0.5
        return raiz(f,g,x,b,eps); //0.1
    else
        return raiz(f,g,a,x,eps); //0.1
}

int signo(double x) { return x<0:-1:1;}

//método para calcular integral: 1.0
double integral(F f, double a, double b, int n) { //0.1
    double s=0, x=a, ancho=(b-a)/n; //0.3
    for (int i=1; i<=n; ++i) { //0.2
        s+=f.valor(x); x+=ancho; //0.2
    }
    return s*ancho; //0.2
}

//Solución 2: integral de max(f(x), g(x))
interface F { public double valor(double x); } //0.25

double area(F f, F g, double a, double b, int n) { //0.25
    double s=0, x=a, ancho=(b-a)/n; //0.3
    for (int i=1; i<=n; ++i) { //0.3
        s+=Math.max(f.valor(x), g.valor(x)); //4.5
        x+=ancho; //0.2
    }
    return s*ancho; //0.2
}
```

## 19. Aerolíneas Beaucheff (P1C4-2001)

Una base de datos que contiene la información de vuelos está compuesta por las siguientes tablas (suponga funcionan en un sistema que cumple con SQL) :

Tabla: Ciudades		Tabla: Aerolíneas		Tabla: Vuelos			
columna	ejemplo	columna	ejemplo	columna	ejemplo	columna	ejemplo
Codigo	SCL	Codigo	LA	Codigo	LA171	DiaSalida	Jueves
Nombre	Santiago	Nombre	LanChile	Origen	SCL	HoraSalida	23:30
				Destino	NYC	DiaLlegada	Viernes
						HoraLlegada	07:15

### Notas

- Todas las columnas son Strings
- Los dos primeros caracteres del código del vuelo corresponden al código de la aerolínea
- Las tres tablas están ordenadas por código

Usando JDBC+SQL cree un programa gráfico que:

a).- Encuentre todos los vuelos que llegan a Santiago el día lunes. Por cada vuelo se debe mostrar el nombre de la ciudad de origen del vuelo, el código del vuelo, y la hora de llegada.

b).- Muestre todos los vuelos directos desde Buenos Aires a Madrid. Cada vuelo debe mostrarse en la forma: nombre de aerolínea, día y hora de salida, duración del vuelo (en horas y minutos). Adicionalmente, muestre el código del vuelo más rápido.

**Nota:** Suponga que todas las horas están expresadas usando una convención horaria universal y que ningún vuelo dura más de 24 horas.

## Solución

Implementación Gráfica:

```
/*
El problema son esencialmente dos consultas SQL, y
un poco de uso de JDBC. El frame tiene un botón de consulta
y tienen que activar uno u otro modo de consulta para cada una de las
partes de la pregunta.
Lo central esta en el método hacerConsulta(..) más abajo.
*/

/*
JDBCExample 0.3a
By BusMan
Consultas, Sugerencias, mcerda@dcc.uchile.cl
*/
```

## Guía de Estudio: Control 4

### CC10A 2004

---

```
*/

import java.sql.*;
import java.awt.*;
import java.awt.event.*;

class JDBCExample extends Frame implements ActionListener{

    /*
    Variables del Frame
    */
    TextArea textBox=new TextArea("",15,30,1);
    Button quit=new Button("Quit");
    Button ok=new Button("Ejecutar");
    BorderLayout b=new BorderLayout();

    /*
    Variables necesarias para la consulta a la Base de datos
    */
    Connection cx;
    Statement stmt;
    ResultSet rs;
    ResultSet rsAux;
    /*
    Variables para cambiar a ODBC o a HSQLDB
    */
    String driverOdbc="sun.jdbc.odbc.JdbcOdbcDriver";
    String tipoOdbc="odbc";

    public JDBCExample(){

        setLayout(b);
        setSize(400,400);
        if( ! cargaDriver(driverHsql) || ! hacerConexion(tipoHsql,"example","sa",""))
            System.exit(0);

        //si pasa el if se cargo el driver y se abrio la conexion
        System.out.println("Driver Cargado, Base de datos Abierta");

        add("North",textBox);
        add("West",command);
        add("East",ok);
        add("South",quit);

        quit.addActionListener(this);
        ok.addActionListener(this);

    }

    public void actionPerformed(ActionEvent e){

        if(e.getSource()==quit){
            terminaConexion();
            System.exit(0);
        }else if(e.getSource()==ok){
            hacerConsulta();
        }

    }

    public void hacerConsulta(){

        /*
        Implementacion para Parte a
        */
        try{
            rs = stmt.executeQuery("SELECT Destino,Codigo.Vuelos, HoraLlegada
                                   FROM Ciudades, Vuelos
                                   WHERE Destino.Vuelos IN (      SELECT Codigo
                                   FROM Ciudades WHERE Nombre='Santiago'");
        }
    }
}
```

```
        )
        AND DiaSalida='Lunes'
    ");

    /*
    Se tiene lista la tabla pero falta reemplazar el codigo de origen por el nombre
    de la ciudad, para no complicar más la consulta es mejor hacerlo usando una sub
    consulta, e insertando con JDBC, así no sale tan enredada la sentencia SQL
    */
    while (rs.next()) {
    /*
    Ahora en rsAux se realiza para cada fila de rs una consulta aparte para sacar el
    nombre de la ciudad
    */
        rsAux=stmt.executeQuery("SELECT Nombre FROM Ciudades
                                WHERE Codigo='"+rs.getString("Destino")+"'");

    /*
    El siguiente next es porque la consulta anterior tendrá 1 fila, y hay que
    ubicarse en ella sino lanza error (es como si estuviera en el índice -1)
    */
        rsAux.next();

    /*
    Ahora si se está en condiciones de mostrar en el frame el resultado requerido
    Notar que se utiliza tanto rsAux como rs
    */
        textBox.append(rsAux.getString("Nombre")+"\t"+rs.getString("Codigo")+
                      "\t"+rs.getString("HoraLlegada"));
        textBox.append("\n");
    }

    } catch (java.sql.SQLException ex){
        System.out.println("Error en la consulta, error: "+ex);
    }

    /*
    Implementacion para Parte b
    */

    try{
        /*
        Ahora, se hace la consulta tal cual, el problema es que, no se podrá calcular el
        tiempo de vuelo, esto se calcula con una función aparte, pero tomamos en la consulta la hora
        de salida y llegada, con esa info se podrá calcular el tiempo de vuelo (no es directo
        restando). El otro problema es que se pide el nombre de la aerolínea y no el código así que
        igual que en la parte a, se debe hacer una búsqueda por cada fila de la columna del
        siguiente resultado
        */

        rs = stmt.executeQuery("SELECT Codigo, DiaLlegada, DiaSalida,
                                HoraSalida, HoraLlegada
                                FROM Vuelos
                                WHERE Origen IN          (SELECT Codigo FROM Ciudades
                                                         WHERE Nombre='Buenos Aires')
                                AND
                                Destino IN              (SELECT Codigo FROM Ciudades
                                                         WHERE Nombre='Madrid')");

    /*
    Pero ahora falta calcular las horas de vuelo y mostrar el nombre de la aerolínea
    */
    }
```

```
y no su código
*/
while (rs.next()) {
    /*
    Para la actual fila, se busca el nombre de la aerolinea
    */
    rsAux=stmt.executeQuery("SELECT Nombre FROM Aerolineas
                              WHERE Codigo='"+rs.getString("Codigo")+"'");

    /*
    El siguiente next es porque la consulta anterior tendra 1 fila, y hay que
    ubicarse en ella sino lanza error (es como si estuviera en el índice -1)
    */
    rsAux.next();

    textBox.append(rsAux.getString("Nombre")+"\t"+rs.getString("DiaSalida")
                  +"\t"+rs.getString("HoraSalida")+"\t"
                  +calculaHoras(rs.getString("HoraSalida"), rs.getString("HoraSalida")) );

    /*Falta crear la funcion calculaHoras, eso se lo dejamos propuesto, es
    sencillo, crear una función que dadas dos horas, con las condiciones dadas en el enunciado,
    devuelve la diferencia en horas entre ellas.*/

    textBox.append("\n");

}

} catch (java.sql.SQLException ex){
    System.out.println("Error en la consulta, error: "+ex);
}

/*FIN METODO*/
}
```

```
boolean cargaDriver(String name){

    try{
        Class.forName( name );
        return true;
    } catch (Exception ex) {
        System.out.println("Imposible Cargar el Driver "+name+" error: "+ex);
        return false;
    }

}

boolean hacerConexion(String tipo, String database, String user, String password){

    try{
        cx = DriverManager.getConnection("jdbc:"+tipo+": "+database, user, password);
        stmt = cx.createStatement();
        return true;
    } catch (java.sql.SQLException ex){
        System.out.println("Problemas con la BD, error: "+ex);
        return false;
    }

}

boolean terminaConexion(){

    try{
        cx.close();
    }
```

## Guía de Estudio: Control 4

### CC10A 2004

---

```
        System.out.println("Conexion a la BD Terminada");
        return true;
    } catch (Exception ex) {
        System.out.println("Imposible Cerrar la Conexion, error: "+ex);
        return false;
    }
}

public static void main(String args[]){
    JDBCExample k=new JDBCExample();
    k.show();
}
}
```

## 20. Búsqueda en raf

Escriba el método *String buscar(String tipo, int id)* que devuelva el nombre del lector (en caso que el tipo sea "lector") o el título del libro (en caso de que el tipo sea "libro"), conocido su ID, las tablas están ordenadas por ID, así que construya un método binario de búsqueda, este método le será útil en la segunda parte.

Note que los archivos lectores.dat y libros.dat poseen la misma estructura. Todo String se encuentra truncado a 32 caracteres y los archivos se encuentran ordenados ascendentemente por id. Los métodos se encuentran dentro de una clase Biblioteca.

Escriba el método *void getMorosos()* que imprima en pantalla ordenadamente la lista de lectores que deben libros a la fecha de hoy (24-09-2003) (nombre del lector - título del libro), recuerde que no es directo comparar dos fechas, primero se debe comparar, año, luego mes y finalmente día.

### Estructura de las Tablas RAF

Lectores: id\_lector(int) - Nombre(string) - email(string)

Libros: id\_libro(int) - título(string) - autor(string)

Prestamos: id\_libro(int) - id\_lector(int) - fecha\_pedido(string)

## Solución

```
import ...

public class Biblioteca {
    /*
    Respecto a la clase auxiliar de hoy, hay dos pequeñas correcciones, no todas las tablas
    tienen el mismo largo de registro, prestamo es diferente, y un char al menos lo que probamos
    en Unix, utiliza 2 bytes, luego es 32*2 en los largos.
    Al resolver un problema escriban al comienzo "considere que un char usa 2 bytes", y así se
    evitan cualquier problema
    */
    RandomAccessFile lectores = null;
    RandomAccessFile libros = null;
    RandomAccessFile prestamos = null;
    int largo_registro = -1;

    public Biblioteca() throws FileNotFoundException {
        lectores = new RandomAccessFile("lectores.dat", "rw");
        libros = new RandomAccessFile("libros.dat", "rw");
        prestamos = new RandomAccessFile("prestamos.dat", "rw");
        largo_registro = 4+32*2+2+32*2+2;
        largo_registro_prestamo = 4 + 4 + 2 + 32*2 ;
    }
    /*
    El siguiente método realiza una búsqueda binaria en RAF, vean que la recursividad, no se hizo
    como tal, sino que en forma de loop-while, pero es la misma idea.
    */
    public String buscar(String tipo, int id) throws IOException {

        RandomAccessFile aux = null;
        if(tipo.equals("lector"))
            aux = lectores;
        else
```

```
        aux = libros;

    int n_registros = (int)(aux.length()/largo_registro);
    int ip=0, iu=n_registros-1;

    while(ip<=iu){
        int im = (ip+iu)/2;
        aux.seek(im*largo_registro);
        int identificador = aux.readInt();
        if(identificador == id){
            return aux.readUTF();
        }
        else if(identificador > id)
            iu = im-1;
        else
            ip = im+1;
    }
    return "No existe";
}

/*
El siguiente método hace la búsqueda de lectores morosos en el sistema, notar que utiliza la
función de búsqueda, para poder mostrar el nombre del lector, y el nombre del libro (eso fue
comentado en clase auxiliar, pero no se construyo dicho método)
*/
public void getMorosos() throws IOException {
    int n_registros = (int)(prestamos.length()/largo_registro);

    for (int i = 0; i < n_registros; i++) {
        prestamos.seek(i*largo_registro_prestamo);
        int id_libro = prestamos.readInt();
        int id_lector = prestamos.readInt();
        String fecha_pedido = prestamos.readUTF();

        if(esFechaMenor(fecha_pedido, "29-09-2003"))
            System.out.println(buscar("lector", id_lector)+" - "+
                buscar("libro", id_libro));
    }
}

/*
Método que es capaz de comparar dos fechas y decir cual es mayor que la otra, las fechas
están en el formato día/mes/año así que hay que comparar primero el año, luego mes, luego
día
*/
public boolean esFechaMenor(String fecha, String fecha_actual){
    int anno_a=Integer.parseInt(fecha.substring(6,10));
    int mes_a=Integer.parseInt(fecha.substring(3,5));
    int dia_a=Integer.parseInt(fecha.substring(0,2));

    int anno_b=Integer.parseInt(fecha_actual.substring(6,10));
    int mes_b=Integer.parseInt(fecha_actual.substring(3,5));
    int dia_b=Integer.parseInt(fecha_actual.substring(0,2));

    if(anno_b<anno_a) return true;
    if(mes_b<mes_a) return true;
    if(dia_b<dia_a) return true;

    return false;
}
}
```



## 21. Manejando Raf

Para manejar Bases de Datos en RAF usualmente se utiliza una clase de representación de estructura de tablas, para simplificar su uso. Se le pide desarrollar dicho modelo siguiendo esta secuencia.

a)- Elabore el objeto linea para poder asociar a una fila de una tabla de tres registros (String, int, double) sus datos correspondientes.

b).- Usando el objeto anterior elabore la clase BDRaf.java (sin main) que tenga la siguiente estructura. (para tres campos, String de largo variable, int, double).

### Class BDRaf

<i>Variables de Instancia</i>	RandomAccessFile, int, int
BDRaf(int x, String archivo)	Constructor, x es el tamaño máximo de caracteres permitidos en el campo nombre.
Public linea leer(int i)	Retorna el i-ésimo registro de la tabla
Public void escribir(linea x, int i)	Escribe la linea x, en el i-ésimo registro.

## Solución:

### Parte a)

```
class linea {  
    String nom; int tel; double prom;  
    linea (String x,int y,double z){  
        nom=x; tel=y; prom=z;  
    }  
}
```

### Parte b)

```
import java.io.*;  
class BDRaf {  
    RandomAccessFile raf;  
    int reg;  
    int len;  
    BDRaf(int x,String arch){  
        try{  
            len=x;  
            raf=new RandomAccessFile(arch,"rw");  
            reg=2+x*2+4+8;  
        }  
        catch(IOException e){  
            System.out.println("error de constructor" );  
        }  
    }  
    public linea leer(int x){  
        try{  
            raf.seek(reg*x);  
            return new linea(raf.readUTF(),raf.readInt(),raf.readDouble());  
        }  
    }  
}
```

```
catch(IOException e){
System.out.println("error");
return null;
}
}
public void escribir(linea l,int x){
l.nom=alargar(l.nom);
try {
raf.seek(reg*x);
raf.writeUTF(l.nom);
raf.writeInt(l.tel);
raf.writeDouble(l.prom);
}
catch(IOException e){
System.out.println("error");
}
}
public String alargar(String s){
if (s.length()>len){
return s.substring(0,len);
}
while (s.length()<len)
s+=" ";
return s;
}
}
```

## 21. Administración con raf.

Siempre preocupado por el rendimiento de sus estudiantes, el director de un colegio decide tomar medidas extremas. Para cada alumno del establecimiento se tienen los siguientes datos guardados, nombre (String), número de telefono (int), y promedio actual (double), con los cual se decide implementar un sistema de alarma que para cualquier estudiante bajo cierta nota, genere una tabla de alumnos con los que debe hablar. Usando el objeto de manejo de RAF en la pregunta anterior, construya un Frame que solicite un promedio mínimo, y que para cualquier estudiante bajo ese promedio, despliegue los tres atributos de la tabla (nombre, teléfono y promedio).

### Solución:

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
public class RAFExample extends Frame implements ActionListener{
    /*
    Variables del Frame
    */
    TextArea textBox=new TextArea(15,30);
    TextField command=new TextField("Ingrese Nota Minima",25);
    Button quit=new Button("Quit");
    Button ok=new Button(" Filtrar ");
    BorderLayout b=new BorderLayout();
    /*
    Parametros de Lectura del Archivo RAF
    Son 3 campos Nombre (String), Telefono (int), Promedio (double)
    */
    String nombreArchivoRAF="alumnos.dat";
    int largoNombre=8;
    /*
    Representacion del archivo RAF como
    una tabla de base de datos
    */
    BDRaf tabla;
    public RAFExample(){
        setLayout(b);
        setSize(400,400);
        setTitle("Acusete 1.0" );
        add("North",textBox);
        add("West",command);
        add("East",ok);
        add("South",quit);
        quit.addActionListener(this);
        ok.addActionListener(this);
        if(!inicializarTabla()){
            System.out.println("Error al abrir el archivo raf como Tabla");
            System.exit(0);
        }
        System.out.println("Archivo Cargado");
        hacerConsulta(8.0);
        //lo muestra todos los registros pues nadie tiene notas mayores que 7.0
    }
    public void actionPerformed(ActionEvent e){
        if(e.getSource()==quit){
            System.exit(0);
        }else if(e.getSource()==ok){
            hacerConsulta(Integer.parseInt(command.getText()));
        }
    }
    public void hacerConsulta( double x){
        int size=tabla.nroDeRegistros();
        linea lineaActual;
```

```
double notaActual;
textBox.append("Nombre\t Telefono \t Promedio\n");
for(int i=0;i<size;i++){
lineaActual=tabla.leer(i);
notaActual=lineaActual.prom;
if(notaActual < x)
textBox.append(lineaActual.nom+"\t"+lineaActual.tel+"\t"+lineaActual.prom+"\n");
}
textBox.append("-----\n");
}
public boolean inicializarTabla(){
try{
tabla=new BDRAF(largoNombre,nombreArchivoRAF);
return true;
}catch(Exception e){
System.out.println("Error al crear el Objeto BDRAF: " +e);
return false;
}
}
public static void main(String args[]){
RAFExample k=new RAFExample();
k.show();
}
}
```

## 22. Media Naranja raf.

Los últimos estudios psiquiátricos realizados en la república de Bochez han detectado extraños comportamientos en la conducta amorosa de sus habitantes. Estos estudios arrojaron el siguiente teorema: en primer año, los hombres que cursan CC10A tienden a hacer pareja con sus compañeras de curso que poseen idéntico promedio en dicho ramo. Por casualidad en la escuela existen archivos que contienen el nombre y el promedio de cada alumno de CC10A, los cuales se encuentran modelados de la siguiente manera.

hombres.dat		mujeres.dat	
Nombre	Promedio	Promedio	Nombre
Cristian	70	50	Marcela
José	55	50	Daniela
Carlos	60	55	Carolina
Daniel	65	55	Carla
Roberto	50	60	Mónica
Andrés	65	60	Jimena
Rodolfo	55	65	Loreto
Ángel	60	65	Ana

Como pueden ver, el archivo de hombres se encuentra desordenado, y el de mujeres se encuentra ordenado por promedio. Se le pide a usted desarrollar un programa que por cada nombre del archivo *hombres.dat* le asocie una pareja del archivo *mujeres.dat*. En caso de que no exista una mujer con el promedio del hombre que se esta buscando, debe indicar que tal hombre no posee pareja.

Ej:

Cristian no tiene pareja.

José hace pareja con Carolina.

Solución:

```
import java.io.RandomAccessFile;
import java.io.IOException;
public class Parejas {
    static int N = 8;
    static int largomujer = 8;
    static int largonota = 2;
    public static void main(String[] args) throws IOException {
        RandomAccessFile Hombres = new RandomAccessFile( "hombres.dat",
        "r");
        RandomAccessFile Mujeres = new RandomAccessFile( "mujeres.dat",
        "r");
        for (int i = 0; i<N; i++) {
            String macho = Hombres.readUTF();
            int nota = Integer.parseInt(Hombres.readUTF());
```

```
String hembra = buscaNota(nota, Mujeres);
if(hembra == null)
    System.out.println(macho+" no tiene pareja.");
else
    System.out.println(macho+ " hace pareja con "+hembra);
}
Hombres.close();
Mujeres.close();
} //main
public static String buscaNota(int notahombre, RandomAccessFile raf)
throws IOException {
    //busca el nombre de la persona cuya nora es nota
    //en un archivo RAF ordenado por nota
    //si no esta devuelve null
    final int L = largonota + 2 + largomujer + 2;
    final int n = (int) (raf.length()/L);
    int ip=0, iu=n-1;
    while(ip <= iu){
        int im = (ip+iu)/2;
        raf.seek(im*L);
        int notamujer = Integer.parseInt(raf.readUTF());
        if(notamujer == notahombre)
            break;
        else if(notahombre < notamujer)
            iu = im-1;
        else
            ip = im+1;
    }
    if(ip <= iu)
        return raf.readUTF();
    else
        return null;
    } //buscaNota
} //class
```