



Guía de Estudio Control 2

Juan Alvarez – Nelson Baloian

Kurt Schwarze – Erich Reimberg – Andrés Muñoz



CC10A - 2003

Tabla de Contenidos

TABLA DE CONTENIDOS	2
1. EL AJEDREZ (PREGUNTA 1 CONTROL 2, 2002)	3
2. MORSE (PREGUNTA 2 CONTROL 2, 2002)	6
3. CENSO (PREGUNTA 3 CONTROL 2, 2002)	8
4. TERNAS DE VALORES (PREGUNTA 1 CONTROL 2, 2001)	10
5. VOTACIONES (PREGUNTA 2 CONTROL 2, 2001)	12
6. DIVISIBLES POR Y (PREGUNTA 3 CONTROL 2, 2001)	14
7. CALENDARIO (PREGUNTA 1 EXAMEN, 1999)	15
8. VOTACIONES II (PREGUNTA 2 EXAMEN, 1999, PROPUESTO)	17
9. POLINOMIOS	18
10. CUADRADOS Y CÍRCULOS	21
11. POLÍGONOS	23
12. CANTIDAD DE HABITANTES	27
13. FRECUENCIAS	28
14. PERÍMETROS DE FIGURAS Y ENLACE DINÁMICO	29
15. ADIVINADOR GRÁFICO DE NÚMEROS	32
16. CONJUNTOS EN JAVA	34

1. El Ajedrez (Pregunta 1 Control 2, 2002)

Para mantener una pieza de ajedrez en el tablero de 8 filas y 8 columnas, se dispone de la clase:

```
class Pieza{
    //private puede cambiarse por protected u omitirse
    private int fila, columna;

    public Pieza(int x,int y){fila=x; columna=y;}
    public boolean mover(int x,int y){fila=x; columna=y; return true;}
    public int obtenerFila(){return fila;}
    public int obtenerColumna(){return columna;}
}
```

(a) Escriba las clases Caballo y Torre que extiendan la clase Pieza con los métodos:

Ejemplo	Significado
C=new Caballo(1,1)	constructor que ubica un caballo en fila 1 y columna 1
C.mover(2,3)	Mueve C a fila 2 y col. 3, y entrega true (movimiento válido)
C.mover(1,2)	Entrega false y no mueve el caballo (movimiento inválido). Nota. Un movimiento de un caballo es válido si avanza 2 filas y 1 columna o 2 columnas y 1 fila dentro del tablero.
T = new Torre(5,6)	constructor que ubica una torre en fila 5 y columna 6
T.mover(5,8)	Mueve T a fila 5 y col 8, y entrega true (movimiento válido).
T.mover(5,9)	Entrega false y no mueve la torre (movimiento inválido). Nota. Es válido si avanza en la misma fila o columna dentro del tablero.

(b) Escriba el método boolean comer(Pieza X,Pieza Y) que devuelva true si la Pieza X "come" en una jugada a la Pieza Y, considerando que no hay más piezas en el tablero. Por ejemplo, si Caballo C=new Caballo(1,2) y Torre T=new Torre(3,1), entonces comer(C,T) devuelve true y comer(T,C) devuelve false.

Solución

Versión 1:

```
// Parte (a)
public class Caballo extends Pieza {
    public Caballo(int x, int y) {
        super(x, y);
    }

    public boolean mover(int nx, int ny) {
        int x = super.obtenerFila();
        int y = super.obtenerColumna();

        // Debe moverse en ambos ejes (x e y <> nx e ny)
        if (nx == x || ny == y) return false;

        // Debe moverse 3 casillas
        if (Math.abs(nx-x) + Math.abs(ny-y) != 3) return false;
    }
}
```

```
// Se mueve el caballo
return super.mover(nx, ny);
}

}

public class Torre extends Pieza {
    public Torre(int x, int y) {
        super(x, y);
    }

    public boolean mover(int nx, int ny) {
        int x = super.obtenerFila();
        int y = super.obtenerColumna();

        // Debe moverse por solo un eje
        if (nx != x && ny != y) return false;

        // Se mueve el caballo
        return super.mover(nx, ny);
    }
}

// Parte (b)
public class MiPrograma {
    public boolean comer(Pieza X, Pieza Y) {
        // Obtenemos la posición de destino de la pieza
        int dx = Y.obtenerFila();
        int dy = Y.obtenerColumna();

        // Verificamos si se puede mover a esa casilla
        boolean come = true;
        if (X instanceof Caballo) {
            come = ((Caballo) X).mover(dx, dy);
        } else {
            come = ((Torre) X).mover(dx, dy);
        }

        // Retornamos lo que ocurrió
        return come;
    }
}
```

Versión 2:

```
// parte(a)
class Caballo extends Pieza {
    public Caballo(int x,int y){
        super(x,y);
    }

    public boolean mover(int x,int y){
        if( (1<=x && x<=8) && (1<=y && y <=8)
            &&(Math.abs(x-obtenerFila())==2 &&
                Math.abs(y-obtenerColumna())==1) ||
            (Math.abs(x-obtenerFila())==1 &&
                Math.abs(y-obtenerColumna())==2)))
            return super.mover(x,y);
        else
            return false;
    }
}

class Torre extends Pieza {
    public Torre(int x,int y){

```

```

        super(x,y);
    }

    public boolean mover(int x,int y){
        if( (1<=x && x<=8) && (1<=y && y <=8)
            && (( x == obtenerFila() && y != obtenerColumna() )
                ||( x != obtenerFila() &&
                    y == obtenerColumna() ) ) )
            return super.mover(x,y);
        else
            return false;
    }

    // parte (b)
    boolean comer(Pieza X, Pieza Y){
        int i = Y.obtenerFila();
        int j = Y.obtenerColumna();
        return X.mover(i,j);
    }

```

2. Morse (Pregunta 2 Control 2, 2002)

La siguiente figura muestra la interfaz de un Applet o Frame que permite ingresar una palabra en código morse en la forma indicada en el siguiente ejemplo en que se ingresa la palabra OLA.

Button	punto	raya	Button
Button	espacio	ingresar otra palabra	Button
Label	Palabra en morse: -. - .- .- .- .-		Label
Label	Palabra en castellano: OLA		Label

Escriba la clase que controle la interfaz anterior de acuerdo a las siguientes reglas:

- un click en el botón "punto" o "raya" debe ingresar un carácter . o un carácter -
- un click en el botón "espacio" indica fin del código de una letra
- un click en el botón "ingresar otra palabra" restablece la situación inicial

Nota: Suponga que existe el método Morse que devuelve la letra correspondiente a un código. Por ejemplo, Morse("...") devuelve "L".

Solución

```

class Traductor extends Frame implements ActionListener {

    private Label
        L1 = new Label("Palabra en morse:"),
        L2 = new Label(""),
        L3 = new Label("Palabra en Castellano:"),
        L4 = new Label("");
    private Button
        punto = new Button("punto"),
        raya = new Button("raya"),
        espacio = new Button("espacio"),
        otra = new Button("ingresar otra palabra");
    private String
        codigo="",
        palabra="";

    static public void main(String[] args){
        Traductor r=new Traductor();
        r.pack();
        r.show();
    }

    public Traductor(){
        this.setLayout(new GridLayout(4,2));

        this.add(punto);
        this.add(raya);
        this.add(espacio);
        this.add(otra);
        this.add(L1);
        this.add(L2);
        this.add(L3);
        this.add(L4);
    }

```

```
punto.addActionListener(this);
raya.addActionListener(this);
espacio.addActionListener(this);
otra.addActionListener(this);
}

public void actionPerformed(ActionEvent x){
    if(x.getSource() == punto)
        codigo += ".";
    else if(x.getSource() == raya)
        codigo += "-";
    else if(x.getSource() == espacio){
        palabra += Morse(codigo);
        codigo = "";
    }
    else {
        codigo="";
        palabra="";
    }
    L2.setText(L2.getText()+codigo);
    L4.setText(palabra);
}
}
```

3. Censo (Pregunta 3 Control 2, 2002)

Los resultados del último censo de población están grabados en el archivo "censo.txt". Cada línea contiene el nombre de una comuna (columnas 1 a 20) y su cantidad de habitantes (columnas 21 a 26).

Escribir un programa que muestre la siguiente tabla de resultados:

Habitantes	Cantidad comunas	% de comunas
0 - 9999	Nº	xx.x
10000 - 19999	Nº	xx.x
...
90000 - 99999	Nº	xx.x
100000 - 119999	Nº	xx.x
120000 - 139999	Nº	xx.x
...
180000 - 199999	Nº	xx.x
200000 y más	Nº	xx.x
Total	Nº	100.0

Solución

Usaremos un arreglo para hacer la tabla que almacenará rango de habitantes y cantidades de comunas en ese rango.

Versión 1:

```
Console c = new Console();

// Inicializamos el arreglo
int rangos[] = new int[21];
for (int i=0; i<21; i++) {
    rangos[i] = 0;
}

// Procesamos el archivo
BufferedReader b = new BufferedReader(new FileReader("censo.txt"));
int nDatos = 0;
int nHab = 0;
while(true) {
    String linea = b.readLine();
    int hab = Integer.parseInt(linea.substring(20));
    if (hab < 200000) {
        rango[hab / 10000]++;
    }
    else {
        rango[20]++;
    }
    nHab += hab;
    nDatos++;
}
b.close();

// Desplegamos tabla
```

```
c.println("Habitantes\tCantidad de Comunas\t% Comunas");
for (int i=0; i<10; i++) {
    c.print((i*10000) + "-" + ((i+1)*10000-1) + "\t");
    c.print(rango[i] + "\t");
    c.println((100.0*rango[i]/nDatos)+"%");
}
for (int i=10; i<20; i+=2) {
    c.print((i*10000) + "-" + ((i+2)*10000-1) + "\t");
    c.print(rango[i]+rango[i+1] + "\t");
    c.println((100.0*(rango[i]+rango[i+1])/nDatos)+"%");
}
c.print("200000 y más\t");
c.print(rango[20]+ "\t");
c.println((100.0*(rango[20])/nDatos)+"%");

// Imprime el total
c.print("Total\t");
c.print(nDatos + "\t");
c.println("100.0%");
```

Versión 2:

```
int[] limite = {10000,20000,30000,40000,50000,60000,70000,
80000,90000,100000,120000,140000,160000,180000,200000,
1000000};

final int N=16;
int[] frecuencia = new int[N];
for(int i=0; i<N; ++i)
    frecuencia[i] = 0;
int total = 0;

BufferedReader A = new BufferedReader(new FileReader("censo.txt"));
while(true){
    String linea = A.readLine();
    if( linea == null ) break;

    int h = Integer.parseInt(linea.substring(20,25));

    for(int i=0, i<N, ++i)
        if(h < limite[i] ){
            ++frecuencia[i];
            break;
        }

    ++total;
}

System.out.println("Habitantes\tcomunidades\t%");
for(int i=0; i<N; ++i){
    System.out.println(
        ((i==0)?0:limite[i-1]) + "-" +
        (i==N-1?"y más":(limite[i]-1) + "\t" +
        frecuencia[i] + "\t" +
        100.0*frecuencia[i]/total);
}
System.out.println("total\t" + total + "\t100.0%"); //0.1
```

4. Ternas de Valores (Pregunta 1 Control 2, 2001)

La siguiente clase permite asignar y recuperar los valores de una terna de números enteros:

```
class Terna {
    protected int a,b,c;
    public Terna() { a=0; b=0; c=0; }
    public void set1(int x){a=x;}
    public void set2(int x){b=x;}
    public void set3(int x){c=x;}
    public int get1(){return a;}
    public int get2(){return b;}
    public int get3(){return c;}
}
```

- a) Escriba la clase Triangulo que extienda la clase Terna agregando los siguientes métodos:

Ejemplo	Resultado	Significado
Triangulo()	-	constructor que inicializa con ceros los valores de los tres lados
T.esTriangulo()	boolean	true si T corresponde a un triángulo o false si no (3 números positivos forman un triángulo si todas las sumas de 2 de ellos son mayores que el 3º)
T.ladosIguales()	int	cantidad de lados iguales (0, 2 o 3) del triángulo T
T.graficar(x)	void	grafica el triángulo T de color x (String "rojo","azul"o"verde")

Nota. NO debe escribir el método graficar

- b) Escriba un programa que dibuje 100 triángulos (de lados enteros entre 1 y 100 generados al azar) de modo que los equiláteros (tres lados iguales) se dibujen de color rojo, los isósceles (dos lados iguales) de color azul, y los escalenos (todos los lados distintos) de color verde.

Nota. Recuerde que Math.random() devuelve un N° real de tipo double en el rango [0,1[

Solución

```
a)
class Triangulo extends Terna{

    public Triangulo(){super();}

    public boolean esTriangulo(){
        if(a+b>c && a+c>b && c+b>a)
            return true;
        return false;
    }

    public int ladosIguales(){
        if(a==b && b==c)
```

```

        return 3;
    if(a==b || b==c || c==a)
        return 2;
    return 0;
    }
}

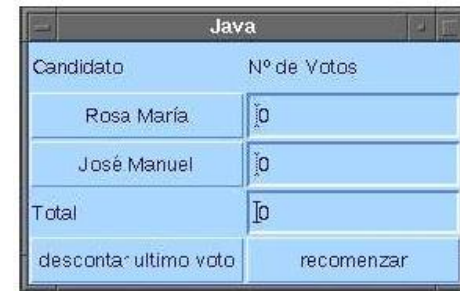
b)
for(int i=0;i<100;i++){
    Triangulo t=new Triangulo();
    t.set1((int)Math.floor(Math.random()*100+1));
    t.set2((int)Math.floor(Math.random()*100+1));
    t.set2((int)Math.floor(Math.random()*100+1));

    if(t.esTriangulo){
        if(t.ladosIguales()==3)
            t.graficar("rojo");
        else if(t.ladosIguales()==2)
            t.graficar("azul");
        else if(t.ladosIguales()==0)
            t.graficar("verde");
        }
    else i--;
}

```

5. Votaciones (Pregunta 2 Control 2, 2001)

La siguiente figura muestra la interfaz de un programa (Frame) que permite apoyar el recuento de votos de una elección con dos candidatos ("Rosa María" y "José Manuel"):



Escriba la clase que controle la interfaz anterior de acuerdo a las siguientes reglas:

- Cada vez que se da un click en el botón de un candidato se debe incrementar en uno el N° de votos del candidato (y el total de votos)
- El botón "recomenzar" debe volver a poner zeros en los tres contadores de votos
- El botón "descontar último voto" sirve para corregir un error al registrar la última preferencia, es decir, debe descontar un voto al candidato que recibió el voto más reciente (y al total de votos)

Solución

```

class Recuento{

    public Frame f;
    private Label L1,L2,L3;
    private TextField votos1,votos2,votos3;
    private Button    cdto1, cdto2, descontar, recomenzar;

    private int v1,v2,ultimoVoto;

    static public void main(String args[]){
        Recuento r=new Recuento();
    }

    public Recuento(){

        f=new Frame();
        f.setLayout(new GridLayout(5,2));

        L1= new Label("Candidato");
        L2= new Label("No de Votos");
        L3= new Label("Total");

        votos1 = new TextField("0");
        votos2 = new TextField("0");
        votos3 = new TextField("0");
    }
}

```

```

        cdto1= new Button("Rosa Maria");
        cdto2= new Button("Jose Manuel");
        descontar= new Button("descontar ultimo voto");
        recomenzar= new Button("recomenzar");

        add(L1); add(L2);
        add(cdto1); add(votos1);
        add(cdto2); add(votos2);
        add(L3); add(votos3);
        add(descontar); add(recomenzar);

        cdto1.addActionListener(new MiEscuchador());
        cdto2.addActionListener(new MiEscuchador());
        descontar.addActionListener(new MiEscuchador());
        recomenzar.addActionListener(new MiEscuchador());

        f.pack(); f.show();

        v1=0; v2=0; ultimoVoto=0;
    }

    class MiEscuchador implements ActionListener{
        public void actionPerformed (ActionEvent x){

            if(x.getSource()==cdto1){
                ++v1;
                ultimoVoto=1;
            }
            else if(x.getSource()==cdto2){
                ++v2;
                ultimoVoto=2;
            }
            else if(x.getSource()==recomenzar)
                v1=v2=0;
            else {
                if(ultimoVoto==1) --v1;
                else --v2;
            }
            votos1.setText(""+v1);
            votos1.setText(""+v2);
            votos1.setText(""+(v1+v2));
        } //actionPerformed
    } //class MiEscuchador
} //class Recuento

```

6. Divisibles por Y (Pregunta 3 Control 2, 2001)

- a) Escriba un método de encabezamiento

int reordenar(int n, int[]x, int y)

que reordene los primeros n elementos del arreglo x en dos grupos: a la izquierda los divisibles por y, y a la derecha los que no son divisibles por y. El método debe entregar adicionalmente como resultado la cantidad de elementos que son divisibles por y. Por ejemplo:

```

int[] a = { 5, 8, 6, 9, 3, 2, 4 }; //arreglo de 7 elementos
int i = reordenar(6, a, 2);        //reordenar los primeros 6
                                    //elementos del arreglo

```

deja el arreglo a con los valores {8,6,2,5,9,3,4} y la variable i con el valor 3

- b) Escribir un programa que use el método anterior para escribir todos los números pares entre 1 y 1000 que son divisibles por 3 y por 5.

Solución

```

a)
int reordenar(int n, int []x, int y){
    int i=0;
    int j=n-1;
    while(i<j){
        if(x[i]%y!=0 && x[j]%y==0){
            int temp=x[j];
            x[j]=x[i];
            x[i]=temp;
        }
        if(x[i]%y==0) i++;
        if(x[j]%y!=0) j--;
    }
    return i;
}

b)
Console c=new Console();
for(int i=1;i<=1000;i++){
    a[i-1]=i;

    int largo=a.length;

    largo=reordenar(largo;a;5);
    largo=reordenar(largo;a;3);

    for(int i=0;i<largo;i++){
        c.println(a[i]);
    }
}

```

7. Calendario (Pregunta 1 Examen, 1999)

Programa el siguiente método:

```
void cal(int[] diasXMes, String[] nombreMes, int diaSemana)
```

Este método debe escribir en el archivo calen.txt un calendario como este:

```
Enero
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

```
Febrero
Mo Tu We Th Fr Sa Su
      1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28
```

cal recibe en diasXmes[i] el numero de días que posee el i-ésimo mes (28, 30 o 31) y en nombreMes[i] el nombre de ese mes (Enero, Febrero, etc.). Ambos arreglos poseen 12 elementos. El parámetro diaSemana indica en que día de la semana comienza el año (1 corresponde 1 a un Lunes, 2 a un Martes, ... y 7 a un Domingo).

El calendario se obtiene invocando cal de manera que diasXmes[1] sea 28 (año bisiesto) y diaSemana sea 3 (el primero de Enero es Miércoles). Preocúpese de cada cifra quede exactamente debajo del día que le corresponde (en la forma indicada en el ejemplo):

Solución

```
void cal(int[] diasXMes, String[] nombreMes, int diaSemana)
    throws IOException {
    PrintWriter pw = new PrintWriter (
        new FileWriter("calen.txt"));

    // Para comenzar los meses
    int ultimoMes = diaSemana;

    // Ciclo por mes
    for (int mes=0; mes<12; mes++) {
        // Nombre del mes
        pw.println(nombreMes[mes]);

        // Dias de la semana
        pw.println("Mo Tu We Th Fr Sa Su");

        // Se salta los primeros días
        for (int i=1; i<ultimoMes; i++)
            pw.print("   "); // 3 espacios
        pw.println();
    }
}
```

```
// Se ponen los días dependiendo de la semana
int semana = ultimoMes;
for (int dia=1; dia<=diasXMes[mes]; dia++) {
    if (dia < 10)
        pw.print(" "); // 1 espacio
    pw.print(dia);
    semana++;
    if (semana > 7) {
        pw.println();
        semana = semana - 7;
    }
}

// Para el siguiente mes
ultimoMes = semana;
pw.println();
pw.println();
}

// Fin de la escritura
pw.close();
}
```


8. Votaciones II (Pregunta 2 Examen, 1999, Propuesto)

El servicio electoral le pide a Ud. que escriba un programa que agilice el recuento de votos para una posible segunda vuelta en las próximas elecciones. El programa debe desplegar una ventana (Frame) con dos botones (uno para cada candidato de la segunda vuelta) y 4 campos de texto que muestren la cantidad de votos que ha percibido cada candidato y el porcentaje relativo de votos. La siguiente figura muestra la apariencia que debe tener la ventana:

```

-----
| cand.A | 6 | 60% | cand.B | 4 | 40% |
-----

```

Cada vez que el usuario presione algunos de los botones, su programa debe incrementar el número de votos del candidato respectivo y recalcular los porcentajes relativos. En este recuento no se consideran los votos nulos y blancos.

9. Polinomios

La siguiente tabla define las operaciones del Tipo de Dato Abstracto (TDA) en una clase llamada **Polinomio**:

Método	Descripción	Tipo Retorno
Polinomio()	Inicializa polinomio vacío	N/A
Polinomio(int n, double[] a)	Inicializa polinomio de grado n con elementos del arreglo a[]	N/A
P.valor(double x)	Evalúa P en x	double
P.suma(Polinomio Q)	P+Q	Polinomio
P.coeficiente(int i, double x)	Asigna x al i-ésimo coeficiente	void
P.grado()	Grado del polinomio	int
P.toString()	Ej: $12.1x^3 + 5x + 4.9$	String

- (a) Escribe un programa que use el TDA **Polinomio** para calcular el seno de un ángulo en radianes a través de la fórmula:

$$\text{Seno}(x) = x/1! - x^3/3! + x^5/5! - x^7/7! + x^9/9! - x^{11}/11!$$

- (b) Escribe el método **suma**, suponiendo la representación (variables de instancia) de **Polinomio**:

```

protected final int N = 10 ;           // grado máximo
protected double[] A = new double[N]   // coeficientes
protected int n = 0                     // grado

```

- (c) Define la clase **Parabola** (heredada a partir de **Polinomio**), para la cual deben implementarse los siguientes métodos:

Método	Descripción
Parabola(double a, double b, double c)	Construye una parábola con coeficientes de ax^2+bx+c
boolean imaginaria()	Retorna verdadero si la parábola no corta el eje x

Solución

```

// (a) Método del seno
public class AlgunPrograma {
    // Método para calcular el factorial
    public int fact(int x) {
        if (x == 0) return 1;
        return x * fact(x-1);
    }

    public double seno(double x) {
        // Declaramos el arreglo de coeficientes
        double a[] = new double[12];

        // Llenamos los coeficientes del polinomio
        boolean positivo = true;
        for (int i=0; i<12; i++) {

```

```
        if (i % 2 == 0) {
            // Caso par
            a[i] = 0;
        }
        else {
            // Caso impar
            a[i] = 1.0 / fact(i);

            // le ponemos el signo
            if (!positivo) {
                a[i] = a[i] * -1;
            }

            // siguiente es de signo cambiado
            positivo = !positivo;
        }
    }

    // Creamos el polinomio
    Polinomio p = new Polinomio(11, a);

    // Retornamos la evaluación en x
    return p.valor(x);
}

// (b) Método Suma
public class Polinomio {
    protected final int N = 10 ;           // grado máximo
    protected double[] A = new double[N]   // coeficientes
    protected int n = 0                    // grado

    ...

    public Polinomio suma(Polinomio Q) {
        // Como se tienen disponibles los coeficientes de
        // cada polinomio, basta sumarlos en un nuevo arreglo
        double[] B = new double[N];
        for (int i=0; i<N; i++) {
            B[i] = this.A[i] + Q.A[i];
        }

        // Ahora retornamos un nuevo polinomio con esos valores
        return new Polinomio(N, B);
    }
}

// (c) Clase Parábola
public class Parabola extends Polinomio {
    public Parabola(double a, double b, double c) {
        // Asignamos los coeficientes al arreglo
        super.A[2] = a;
        super.A[1] = b;
        super.A[0] = c;

        // Le asignamos el grado
        super.n = 2;

        // inicializamos con 0 el resto de los coeficientes
        for (int i=3; i<N; i++) {
            super.A[i] = 0;
        }
    }
}
```

```
public boolean imaginaria() {
    // Lo único que hay que analizar es si
    //  $b^2 - 4ac$  es  $< 0$ , ya que eso determina si
    // los valores de x son imaginarios
    return ((Math.pow(super.A[1], 2) - 4 * super.A[2] *
        super.A[0]) < 0);
}
```

10. Cuadrados y Círculos

(a) Escribe la clase `Circulo` de manera que acepte las siguientes instrucciones

Métodos	Descripción
<code>void asignar(int r, int h, int v)</code>	Asigna radio y coordenadas vertical y horizontal
<code>void dibujar(Graphics G)</code>	Dibuja el círculo en un objeto <code>Graphics</code>
<code>double area()</code>	Retorna el área del círculo

(a) Escribe la clase `CuadradoCircunscrito` que se derive de la clase `Circulo` y que incluya las operaciones

Métodos	Descripción
<code>void dibujar(Graphics G)</code>	Dibujar el cuadrado (y el círculo inscrito)
<code>double area()</code>	Retorna el área del cuadrado

(b) Escribe un programa que reciba un arreglo de objetos, los dibuje y entregue la suma de sus áreas

- `double sumaDibujando(int n, Circulo[] x)`

Solución

```
import java.awt.*;

public class Circulo {
    protected int centroX;
    protected int centroY;
    protected int radio;

    public Circulo(){
        asignar(0,0,0);
    }

    public void asignar(int r, int h, int v){
        radio=r;
        centroX=h;
        centroY=v;
    }

    public void dibujar(Graphics g){
        g.setColor(Color.RED);
        g.drawOval(centroX-radio,centroY-radio, 2*radio,2*radio);
    }

    public double area(){
        return Math.PI*radio*radio;
    }
}
```

```
public static double sumaDibujando(int n, Circulo[] x,Graphics g){
    double sumaAreas=0;
    for (int i = 0; i < n; i++) {
        Circulo circulo = x[i];
        circulo.dibujar(g);
        sumaAreas+=circulo.area();
    }
    return sumaAreas;
}

import java.awt.*;

public class CuadradoCircunscrito extends Circulo{

    public CuadradoCircunscrito(){
        asignar(0,0,0);
    }

    public void dibujar(Graphics g) {
        super.dibujar(g);
        g.drawRect(centroX-radio,centroY-radio,2*radio,2*radio);
    }

    public double area() {
        return 4*radio*radio;
    }
}
```

11. Polígonos

La clase Poligono usa la clase Vertice definida de la siguiente forma:

```
public class Vertice {
    public int x, y ;

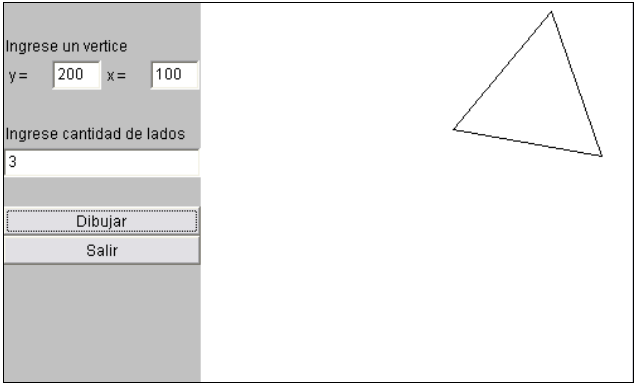
    public Vertice (int x, int y) {
        this.x = x ;
        this.y = y ;
    }
}
```

Los polígonos tienen una representación interna como un arreglo de vértices, y el largo máximo de vértices que puede tener un polígono es de 100. Además, para que el polígono sea cerrado, debe repetirse el último vértice (igual al primero).

Los métodos de la clase Poligono son:

Método	Descripción
Poligono()	Crea un polígono sin vertices
void agregar(int x, int y)	Agrega el vertice (x, y) al Poligono
void eliminar(int n)	Elimina el n-ésimo vértice el polígono, dejando el resto contiguo
void dibujar(Graphics G)	Dibuja el polígono

- (a) Implementa la clase Polígono
- (b) Deriva de Poligono una clase Regular, que dado un entero $n < 100$ y un vértice, cree un polígono regular de n lados
- (c) Escribe un applet que permita al usuario ingresar un vértice y la cantidad de lados , de modo que cada vez que se presione el botón Dibujar, un polígono regular se dibuje a partir de dicho vértice. La pantalla debe lucir como la siguiente:



Solución

```
// (a) Clase Poligono
import java.awt.*;

public class Poligono {
    private Vertice[] vertices;
    private static int MAX_CANTIDAD=100;
    private int cantidad;

    public Poligono(){
        vertices=new Vertice[MAX_CANTIDAD];
        cantidad=0;
    }

    public void agregar(int x, int y){
        vertices[cantidad++]=new Vertice(x,y);
    }

    public void eliminar(int n){
        for (int i = n-1; i < cantidad-1; i++) {
            vertices[i]=vertices[i+1];
        }
        cantidad--;
    }

    public void dibujar(Graphics G){
        for (int i = 1; i < cantidad; i++) {
            Vertice verticeAnterior = vertices[i-1];
            Vertice verticeActual=vertices[i];
            G.drawLine(verticeAnterior.x,
            verticeAnterior.y,verticeActual.x,verticeActual.y);
        }
    }
}

// (b) Clase Regular
public class Regular extends Poligono{

    public Regular(int n,Vertice vertice){
        //calculamos un angulo inicial al centro del poligono al azar
        double angulo=Math.random()*2*Math.PI;
        //calculamos una distancia inicial al centro del poligono
        double largo=Math.random()*100;
        //calculamos el angulo del poligono regular
        double deltaAngulo=2*Math.PI/n;
        //coordenadas del centro del poligono
        double centroX=vertice.x-largo*Math.cos(angulo);
        double centroY=vertice.y-largo*Math.sin(angulo);

        agregar(vertice.x,vertice.y);
        for (int i = 0; i < n; i++) {
            angulo+=deltaAngulo;
            double x=centroX +Math.cos(angulo)*largo;
            double y=centroY+Math.sin(angulo)*largo;
            agregar((int) x,(int) y);
        }
        agregar(vertice.x,vertice.y);
    }
}
```

```
// (c) Applet
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;

public class MyApplet extends Applet {
    private Canvas c;
    private Button dibujar;
    private Button salir;
    private TextField vertX;
    private TextField vertY;
    private TextField lados;

    public void init() {
        c = new Canvas();
        setLayout(new BorderLayout());
        setSize(500, 300);

        vertX = new TextField();
        vertY = new TextField();

        Panel numeros = new Panel();
        numeros.setLayout(new GridLayout(1, 4));
        numeros.add(new Label(" y = "));
        numeros.add(vertX);
        numeros.add(new Label(" x = "));
        numeros.add(vertY);

        dibujar = new Button("Dibujar");
        lados = new TextField();
        salir = new Button("Salir");

        Panel west = new Panel();
        west.setBackground(Color.lightGray);
        west.setLayout(new GridLayout(13, 1));
        west.add(new Label(""));
        west.add(new Label("Ingresa un vertice"));
        west.add(numeros);
        west.add(new Label(""));
        west.add(new Label("Ingresa cantidad de lados"));
        west.add(lados);
        west.add(new Label(""));
        west.add(dibujar);
        west.add(salir);
        //agregamos algunos labels para que se vea mejor
        west.add(new Label(""));
        west.add(new Label(""));
        west.add(new Label(""));

        add(BorderLayout.WEST, west);
        add(BorderLayout.CENTER, c);

        //agregamos Listeners
        salir.addActionListener(new QuitListener());
        dibujar.addActionListener(new DrawListener());
    }

    class QuitListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            System.exit(0);
        }
    }
}
```

```
class DrawListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String textX = vertX.getText();
        String textY = vertY.getText();
        String textL = lados.getText();
        if(textX.equals("") || textY.equals("") ||
        textL.equals("")) {
            return;
        }
        Vertice v = new Vertice(Integer.parseInt(textX),
        Integer.parseInt(textY));
        Poligono p = new Regular(Integer.parseInt(textL), v);
        p.dibujar(c.getGraphics());
    }
}
```

12. Cantidad de Habitantes

Escribe un programa que solicite la cantidad de habitantes de cada una de las 13 regiones del país y escriba la siguiente tabla:

Region	Población	%
1		
2		
...		
13		
Total		100.0

Solución

```
Console c = new Console();
int[] pobl = new int[13];
int total = 0;

for (int i=0; i<13; i++) {
    c.println("Ingrese la población de la región " + i);
    pobl[i] = c.readInt();
    total += pobl[i];
}

c.println("Region  Población          %");
c.println("-----");
for (int i=0; i<13; i++) {
    c.println(i + "          " + pobl[i] +
              "          " + (pobl[i]*100/total));
}
c.println("Total  " + total + "          100.0");
```

13. Frecuencias

Escribe un programa que determine la frecuencia de aparición de cada una de las letras del alfabeto (sin importar mayúsculas / minúsculas) de un archivo de texto:

letra	frecuencia
a	
b	
...	
z	

Solución

Definiremos un método que realice y que lo devuelva en un arreglo de enteros, en donde cada una de las 61 letras (a -> z sin contar ñ, ll ni ch) corresponde a un espacio del arreglo. Para ello definiremos una constante String con las letras del abecedario.

```
final String abecedario = "abcdefghijklmnopqrstuvwxyz";
int[] cuentaLetras(String nombreArchivo) {
    BufferedReader f = new BufferedReader(
        new FileReader(nombreArchivo));
    // Se crea e inicializa el arreglo de resultados
    int[] letras = new int[abecedario.length()];
    for (int i=0; i<letras.length; i++)
        letras[i] = 0;
    // Mientras se lee el archivo contaremos las letras
    String linea;
    while( (linea = f.readLine()) !=null) {
        for (int c=0; c<linea.length(); c++) {
            int p = abecedario.indexOf(
                linea.charAt(c));
            // Solo consideramos letras
            if ( p>=0 )
                letras[p]++;
        }
    }
    f.close();
    return letras;
}
```

Y el programa principal para que imprima esto:

```
Console C = new Console();
String nombre = c.readLine();
int[] n = cuentaLetras(nombre);
C.println("letra  frecuencia");
C.println("-----");
for (int j=0; j<n.length; j++) {
    C.print(abecedario.charAt(j));
    C.println(n[j]);
}
```

14. Perímetros de Figuras y Enlace Dinámico

Dada la siguiente clase Figura:

```
class Figura {
    public Figura() { }
    public double perimetro() { return 0; }
}
```

(a) Implemente una clase Circulo que, usando la siguiente representación interna:

```
class Circulo extends Figura {
    double cx, cy; // Centro
    double r; // Radio
    public Circulo(double x, double y, double r) {...}
    public double perimetro() {...}
}
```

permita usar la clase Círculo.

(b) Implemente una clase Cuadrado que, usando la siguiente representación interna:

```
class Cuadrado extends Figura {
    double cx, cy; // Centro
    double l; // Lado
    public Cuadrado(double x, double y, double l) {...}
    public double perimetro() {...}
}
```

también permita usar la clase Cuadrado.

(c) Haciendo un paralelo con las clases anteriores, implemente la clase Poligono definida como:

```
class Poligono extends Figura {
    double x[], y[]; // Puntos
    public Poligono(double[] x, double[] y) {...}
    public double perimetro() {...}
}
```

poniendo énfasis en que el perímetro es la suma de las distancias entre los puntos contiguos de un polígono.

(d) Haga un método:

```
public static double sumaPerimetros(Figura[] fig) {...}
```

que recibe un arreglo de objetos de tipo Figura y que retorna la suma de los perímetros de todas las figuras independientes del Tipo Dinámico.

Solución

```
// parte (a)
class Circulo extends Figura {
    double cx, cy; // Centro
    double r; // Radio
    public Circulo(double x, double y, double r) {
        super("Circulo");
        this.cx = x;
        this.cy = y;
        this.r = r;
    }
    public double perimetro() {
        return 2 * Math.PI * this.r;
    }
}

// parte (b)
class Cuadrado extends Figura {
    double cx, cy; // Centro
    double l; // Lado
    public Cuadrado(double x, double y, double l) {
        super("Cuadrado");
        this.cx = x;
        this.cy = y;
        this.l = l;
    }
    public double perimetro() {
        return 4 * this.l;
    }
}

// parte (c)
class Poligono extends Figura {
    double x[], y[]; // Puntos
    public Poligono(double[] x, double[] y) {
        super("Poligono");
        this.x = new double[x.length];
        this.y = new double[y.length];
        for (int i=0; i<x.length; i++) {
            this.x[i] = x[i];
            this.y[i] = y[i];
        }
    }
    public double perimetro() {
        double suma = 0;
        for (int i=0; i<x.length-1; i++) {
            suma += Math.sqrt(
                Math.pow(x[i+1] - x[i], 2) +
                Math.pow(y[i+1] - y[i], 2));
        }
        suma += Math.sqrt(
            Math.pow(x[0] - x[x.length-1], 2) +
            Math.pow(y[0] - y[y.length-1], 2));
        return suma;
    }
}

public class MiPrograma {
    // parte (d)
    public static double sumaPerimetros(Figura[] fig) {
        double suma = 0;
        for (int i=0; i<fig.length; i++) {
```

```

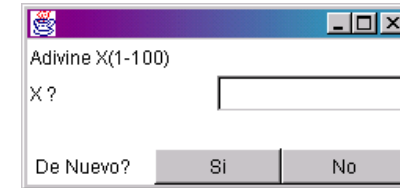
        suma += fig[i].perimetro();
    }
    return suma;
}

public static void main(String[] args) {
    Console c = new Console("Figuras");
    c.println("FIGURAS");
    c.print("# de Figuras?");
    int n = c.readInt();
    Figura[] fig = new Figura[n];
    for (int i=0; i<n; i++) {
        c.print("(C)írculo o C(u)adrado?");
        String a = c.readLine();
        c.println("Ingresa CENTRO");
        c.print("X?");
        double cx = c.readDouble();
        c.print("Y?");
        double cy = c.readDouble();
        c.print("Radio/Lado?");
        double x = c.readDouble();
        if (a.toUpperCase().equals("C")) {
            fig[i] = new Circulo(cx, cy, x);
        }
        else if (a.toUpperCase().equals("U")) {
            fig[i] = new Cuadrado(cx, cy, x);
        }
    }
    c.println("La suma de los perímetros es: " +
        sumaPerimetros(fig));
}
}

```

15. Adivinador Gráfico de Números

Escriba una I.G que permita adivinar un N° de acuerdo al siguiente diálogo:



Solución

```

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

public class Adivinanza extends Frame implements ActionListener {
    Label titulo = new Label("Adivine X(1-100)"),
        pregunta = new Label("X ?"),
        respuesta = new Label(),
        denuevo = new Label(" De Nuevo? ");
    TextField numero = new TextField();
    Button si = new Button("Si"),
        no = new Button("No");
    int x; // NUMERO SECRETO!!!!!!

    public Adivinanza(){
        inicializar();

        Panel p1 = new Panel();
        p1.setLayout(new GridLayout(1,2));
        p1.add(pregunta);
        p1.add(numero);

        Panel p2 = new Panel();
        p2.setLayout(new GridLayout(1,3));
        p2.add(denuevo);
        p2.add(si);
        p2.add(no);

        setLayout(new GridLayout(4,1));
        add(titulo);
        add(p1);
        add(respuesta);
        add(p2);

        numero.addActionListener(this);
        si.addActionListener(this);
        no.addActionListener(this);

        this.pack();
        this.show();
    }

    public void actionPerformed(ActionEvent e){
        if(e.getSource() == si)
            inicializar();
    }
}

```



```
        else
            if(e.getSource() == no)
                System.exit(0);
            else{
                int n = Integer.parseInt(
                    numero.getText());
                if(x == n)
                    respuesta.setText("X = "+n);
                if(x < n)
                    respuesta.setText("X < "+n);
                if(x > n)
                    respuesta.setText("X > "+n);
            }
    }

    public void inicializar(){
        x = 1+ (int)(Math.random()*100);
        respuesta.setText("");
        numero.setText("");
    }

    static public void main(String[] args) {
        Adivinanza n = new Adivinanza();
    }
}
```

16. Conjuntos en Java

Los conjuntos son reales repositorios donde se almacenan distintos tipos de datos. Por ejemplo, existend conjuntos de Números, Letras, Palabras y hasta Objetos (mesa, silla, casa, etc).

Se desea implementar conjuntos en Java, y se conoce que las funcionalidades de los conjuntos son:

- Unión: Un conjunto A se une con un conjunto B dando como resultado un conjunto C que contiene los elementos de A y B sin repetirlos.
- Intersección: Un conjunto A se intersecta con un conjunto B dando como resultado un conjunto C que contiene solo los elementos repetidos entre A y B.
- Subconjunto: Un conjunto A es subconjunto de B si los elementos de A están también en B, dando como resultado Verdadero o Falso.
- Igualdad: Un conjunto A es igual al conjunto B si A es subconjunto de B y B es subconjunto de A, dando como resultado Verdadero o Falso.
- Cardinal: Retorna un número que indica la cantidad de elementos que tiene el conjunto A.

(a) Implemente una Interface Conjunto que represente estas cuatro funcionalidades como genéricas de los conjuntos.

(b) Se desea implementar una Clase ConjuntoEnteros que represente un Conjunto (implemente la interface) de número enteros. Para ello puede utilizar la siguiente representación interna:

```
public class ConjuntoEntero implements Conjunto {
    protected int[] bolsa;
    public ConjuntoEntero() {...} // Lo crea vacío
    public ConjuntoEntero(int x) {...} // Lo crea con elto x
    public ConjuntoEntero(int[] a) {...} // Lo crea con arreglo a
    ...
}
```

Nota: Se le entregan 3 constructores para que los implemente.

Solución

```
public interface Conjunto {
    public int cardinal(); public Conjunto union (Conjunto B);
    public Conjunto interseccion (Conjunto B);
    public boolean subconjunto (Conjunto B);
    public boolean igualdad (Conjunto B);
}

public class ConjuntoEntero implements Conjunto {
    protected int[] bolsa;

    public ConjuntoEntero() {
        // No es necesario hacer nada
    }
}
```

```
public ConjuntoEntero(int x) {
    // Creamos un arreglo de 1 elemento
    this.bolsa = new int[1];
    this.bolsa[0] = x;
}

public ConjuntoEntero(int[] a) {
    // Creamos el arreglo con el largo de a
    this.bolsa = new int[a.length];
    for (int i=0; i<a.length; i++)
        this.bolsa[i] = a[i];
}

public boolean cardinal () {
    // Retornamos el largo de bolsa
    return this.bolsa.length;
}

public Conjunto union (Conjunto B) {
    // Contamos si hay repetidos
    int rep = 0;
    for (int i=0; i<this.bolsa.length; i++) {
        int[] a = new int[1];
        a[0] = this.bolsa[i];
        Conjunto X = new Conjunto(A);
        if (X.subconjunto(B))
            rep++;
    }

    // Creamos un arreglo que contenga todo
    int[] r = new int[this.cardinal()+B.cardinal()-rep];
    int n = 0;

    // Ponemos los elementos que estan en A y
    // no estan en B
    for (int i=0; i<this.bolsa.length; i++) {
        int[] a = new int[1];
        a[0] = this.bolsa[i];
        Conjunto X = new Conjunto(A);
        if (!X.subconjunto(B)) {
            r[n] = this.bolsa[i];
            n++;
        }
    }

    // Ponemos los elementos de B
    for (int i=0; i<B.length; i++) {
        r[n] = B.bolsa[i];
        n++;
    }

    // Retornamos el conjunto creado con r
    return new Conjunto(r);
}

public Conjunto interseccion (Conjunto B) {
    // Contamos si hay repetidos
    int rep = 0;
    for (int i=0; i<this.bolsa.length; i++) {
        int[] a = new int[1];
        a[0] = this.bolsa[i];
        Conjunto X = new Conjunto(A);
```

```
        if (X.subconjunto(B))
            rep++;
    }

    // Creamos un arreglo para repetidos
    int[] r = new int[rep];
    int n = 0;

    // Ponemos los elementos que estan en A y
    // que también estan en B
    for (int i=0; i<this.bolsa.length; i++) {
        int[] a = new int[1];
        a[0] = this.bolsa[i];
        Conjunto X = new Conjunto(A);
        if (X.subconjunto(B)) {
            r[n] = this.bolsa[i];
            n++;
        }
    }

    // Retornamos un conjunto creado con r
    return new Conjunto(r);
}

public boolean subconjunto (Conjunto B) {
    // Vemos si la intersección es igual a A
    Conjunto C = this.interseccion(B);
    return this.igualdad(C);
}

public boolean igualdad (Conjunto B) {
    // Suponemos iguales, y probamos lo contrario
    boolean igual = true;
    for (int i=0; i<this.bolsa.length; i++) {
        boolean esta = false;
        for (int j=0; j<B.cardinal(); j++)
            if (this.bolsa[i] == B[j])
                esta = true;
        igual = igual && esta;
    }
    return igual;
}
```