Repaso/resumen: Introducción a la Programación orientada a objetos

- Strings
- Archivos
- Clases y objetos
- Herencia
- Ventanas
- Dibujos
- Gráficos

Archivos

```
import java.io.*;
class CopiarArchivos{
static public void main(String[]args)
throws IOException{
  BR I=new BR(new FR(U.readLine("input?")));
  PW O=new PW(new FW(U.readLine("output?"));
  int nc=0,int nl=0;
  String linea;
  while((linea=I.readLine())!= null){
    O.println(linea);
    ++nl; nc+=linea.length();
  }
  O.close(); I.close();
  U.println(nl+" líneas "+nc+" caracteres");
}
}
```

Clase String

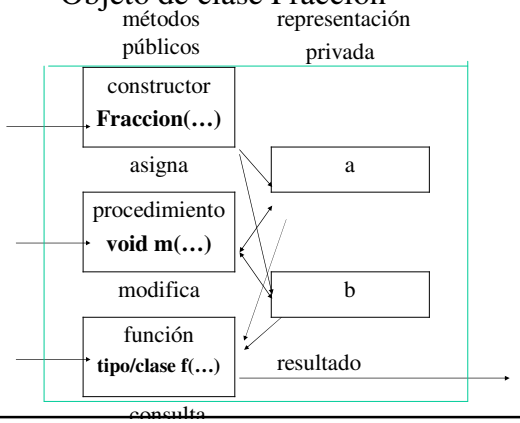
```
final class String{     //no se puede extender
String(String x){...}   //constructor
//funciones: devuelven un valor
int length(){...}       //nº de caracteres
char charAt(int x){...} //caracter en índice x(desde 0)
int indexOf(String x){...} //índice de x(-1 si no está)
int indexOf(String x,int y){...} //índice de x, desde y
boolean equals(String x){...}//true si iguales
int compareTo(String x){...}  //0, <0, >0
//devuelven String (no modifican objeto)
String substring(int x,int y){...}//en índices [x,y[
String substring(int x){...}//en índices [x,length()[
String concat(String x){...}//añadir x
String replace(char x,char y){...}//todos los x por y
String toUpperCase(){...}//a mayúsculas
String toLowerCase(){...}//a minúsculas
String trim(){...}//eliminar espacios de izq y der
...
}
```

Definición (declaración de clase)

```
class NombreClase{
//representación (datos, atributos)
private ...;
//métodos:
//constructores: inicializan objeto
public NombreClase(parámetros){...}
...
//funciones: entregan un resultado
public tipo/clase nombre(parámetros){...}
...
//procedimientos: modifican objeto
public void nombre(parámetros){...}
...
//internos (de servicio)
private ... nombre(parámetros){...}
}
```

```
//obtener una respuesta "si" o "no"
static public String siOno(String x){
  String r=U.readLine(x+" si o no?")
           .trim().toLowerCase();
  return r.equals("si")||r.equals("no")) ?
    r : siOno(x);
}
//invierte string. Ej:inverso("roma")="amor
static public String inverso(String x){
  if(x.length()==0) return "";
  return inverso(x.substring(1))+x.charAt(0);
}
//capicua: capicua("reconocer")=true
static public boolean capicua(String x){
  return x.equals(inverso(x));
}
```

Objeto de clase Fracción



Clase Fraccion(F): resumen de métodos

método	Función	Procedimiento
constructor	F(int x,int y) F() F(F x) F(tipo/Clase x)	canónico default copia conversión
operador unario	F simple() F inverso()	void simplificar() void invertir()
operador binario	F suma(F x) F resta(F x) F producto(F x) F division(F x)	void sumar(F x) void restar(F x) void multiplicar(F x) void dividir(F x)
comparación	boolean equals(F x) int compareTo(F x)	
asignación		void asignar(tipo/Clase x)
conversión	T/C toT/C()	T:tipo, C:Clase
selector/ modificador	int numerador() int denominador()	void numerador(int x) void denominador(int x)

Problema . Escriba la clase base Figura y las clases extendidas Cuadrado y Circulo para el siguiente método que permite calcular y mostrar el área y el perímetro de un rectángulo o un círculo

```
static public void mostrar(Figura f){
  U.println("area="+f.area());
  U.println("perimetro="+f.perimetro());
}
```

Enlace dinámico
El método mostrar invocará a las funciones area y perímetro de la clase del objeto apuntado por f.

Ej de uso:

```
mostrar(U.azar(0,1)==1 ?
  new Cuadrado(U.azar(1,100)):
  new Circulo(U.azar(1,100)));
```

Uso de objetos de una clase

```
//Fracción más lejana (del promedio)
F f=new F(U.readLine("n°/n°?")), s=new F(0),
  min=new F(f), max=new F(f), cero=new F();
int n=0;
while(!f.equals(cero)){
  s.sumar(f); ++n;
  if(f.compareTo(max)>0) max.asignar(f);
  if(f.compareTo(min)<0) min.asignar(f);
  f.asignar(U.readLine("n°/n°?"));
}
s.dividir(new F(n)); //promedio
U.println("más lejana="+
  (max.resta(s).compareTo(s.resta(min))>0 ?
  max : min).toString() );
```

Clase abstracta: con métodos que **deben** definirse en extensiones

```
abstract class Figura{
protected double x;
public Figura(double x){
  this.x=x; if(x<=0) U.abortar("<=0");}
abstract public double area();
abstract double perimetro();
}
class Cuadrado extends Figura{
public Cuadrado(double x){super(x);}
public double area(){return x*x;}
public double perimetro(){return 4*x;}
}
class Circulo extends Figura{
public Circulo(double x){super(x);}
public double area(){return Math.PI*x*x;}
public double perimetro(){return 2*Math.PI*x;}
}
```

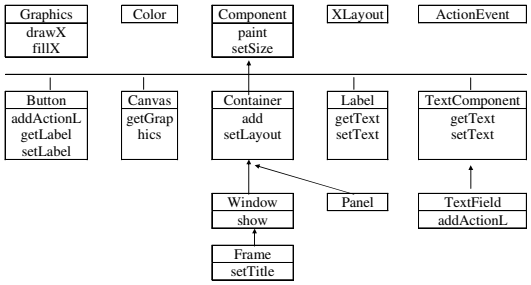
Herencia

```
class Circunferencia{ //clase base
protected double r;//visible en clase y extensiones
public Circunferencia(double x){r=x;}
public double perimetro(){return 2*Math.PI*r;}
}
class Circulo extends Circunferencia{//extensión
public double area(){return Math.PI*r*r;}
public Circulo(double x){super(x);}
}
class Programa{
static public void main(String[]args)throwsIOException{
  Circunferencia c=new Circunferencia(1);
  U.println("perímetro="+c.perimetro());
  Circulo circulo=new Circulo(2);
  U.println("area="+circulo.area());
  U.println("perímetro="+circulo.perimetro());
}
}
```

Interface: clase abstracta sólo con métodos abstractos

```
interface Figura{
public double area();
public double perimetro();
}
class Cuadrado implements Figura{
protected double a;
public Cudrado(double x){a=x;}
public double area(){return a*a;}
public double perimetro(){return 4*a;}
}
class Circulo implements Figura{
protected double r;
public Circulo(double x){r=x;}
public double area(){return Math.PI*r*r;}
public double perimetro(){return 2*Math.PI*r;}
}
```

Interfaces con Usuario (Ventanas): Jerarquía de Herencia




```
public void actionPerformed(ActionEvent x){
  if(x.getSource()==quit) System.exit(0);
  if(x.getSource()==rendir)
    resp.setText("X="+X);

  else if(x.getSource()==ayuda)
    resp.setText(min+"<X<"+max);

  else if(x.getSource()==jugar){
    X=U.azar(1,100); min=0; max=101;
    resp.setText(""); numero.setText("");

  }else if(x.getSource()==numero){
    int n=Integer.parseInt(numero.getText());
    if(X==n) resp.setText("X="+n);
    if(X<n){resp.setText("X<"+n);if(n<max)max=n;}
    if(X>n){resp.setText("X>"+n);if(n>min)min=n;}
  }
}}
```

Programar juego de adivinanza de un nº con la sgte interfaz

	Label	TextField	
p1	X(1-100)?	23	Panel
resp	X<23		Label
p2	jugar	quit	ayuda	rendir	Panel
	Button	Button	Button	Button	

```
import java.awt.*; import java.awt.event.*;
class Juego extends Frame implements ActionListener
{
static public void
main(String[]x){newJuego().show();}
private Label resp=new Label("");
private TextField numero=new TextField();
private Button
  jugar=new Button("jugar"),
  quit=new Button("quit"),
  ayuda=new Button("ayuda"),
  rendir=new Button("rendir");
private int X=U.azar(1,100),min=0,max=101;
```

Dibujos con paint

```
import java.io.*; import java.awt.*;
class Ojo extends Frame{
private static final int W;
static public void main(String[]x)
throws IOException{
  W=U.readInt("tamaño en pixeles?");
  new Ojo().show();//invoca a paint
}
public Ojo(){
  setSize(W,W);
}
public void paint(Graphics x){
  x.drawOval(0,0,W,W);
  x.fillOval(W/4,W/4,W/2,W/2);
}
}
```

```
public Juego(){
  //diagramar paneles
  Panel p1=new Panel(); p1.setLayout(new GL(1,2));
  p1.add(new Label("X(1-100)?")); p1.add(numero);
  Panel p2=new Panel(); p2.setLayout(new GL(1,4));
  p2.add(jugar); p2.add(quit);
  p2.add(ayuda); p2.add(rendir);

  //diagramar ventana
  setLayout(new GL(3,1));
  setSize(200,50*3);
  add(p1); add(resp); add(p2);

  //activar escuchador
  numero.addActionListener(this);
  jugar.addActionListener(this);
  quit.addActionListener(this);
  ayuda.addActionListener(this);
  rendir.addActionListener(this);
}
```

Dibujos con canvas (circunferencias al azar)

```
import java.awt.*; import java.awt.event.*;
class C extends Frame implements ActionListener{
static public void main(String[]x){new C().show();}
private Canvas cv=new Canvas();
private Button b=new Button("click");
public C(){
 cv.setSize(256,256);
 setLayout(new BorderLayout());
 add("center",cv); add("south",b);
 b.addActionListener(this);
}
public void actionPerformed(ActionEvent x){
 int i=U.azar(0,255);
 Graphics g=cv.getGraphics();
 g.setColor(new Color(i,i,i));
 g.drawOval(i,i,i,i);
}
}
```

```
static public void graficar(Funcion f,
int n,double a,double b,Graphics g,int W,int H)
{
  double delta=(b-a)/(n-1),
    min=min(f,n,a,b),max=max(f,n,a,b),//propuestos
  int h=0, v=H-pixel(f.valor(a),min,max,H);
  for(double x=a+delta; x<=b; x+=delta)
    g.drawLine(h,v,h=pixel(x,a,b,W),
                  v=H-pixel(f.valor(x),min,max,H));
}
static public int pixel(d x,d y,d z,int w){//y<=x<=z
  return (int)Math.round(w*(x-y)/(z-y));//[0,w]
}
```

Uso:

```
graficar(new Seno(),50,0,Math.PI,g,W,H);//Graphics g

class Seno implements Funcion{
public double valor(double x){return Math.sin(x);}
}
```

```
case '+' : case '-': case '*': case '/':
  //guardar primer operando y operacion
  op1=Double.parseDouble(visor.getText());
  op=c; s=“”; break;
case '=':
  //realizar operación entre operandos
  op2=Double.parseDouble(visor.getText());
  switch(op){
  case '+': op1+=op2; break;
  case '-': op1-=op2; break;
  case '*': op1*=op2; break;
  case '/': op1/=op2; break;
  }
  s=""+op1; break;
}
//actualizar visor
visor.setText(s);
}}
```

7	8	9	/
4	5	6	*
1	2	3	-
0	.	=	+
S	C	Q	

```
private Label visor=new Label("");
public Calculadora(){
  //diagramar panel y activar escuchador botones
  Panel p=new Panel();p.setLayout(new GL(5,4));
  String s="789/456*123-0.=+SCQ“;//carac botones
  for(int i=0; i<s.length(); ++i){
    Button b=new Button(s.substring(i,i+1));
    p.add(b); b.addActionListener(this);
  }
  //diagramar ventana
  setSize(4*50,6*25); setLayout(new BL());
  add("North",visor); add("Center",p);
}
```

```
//operandos y operación
private double op1,op2; private char op;
public void actionPerformed(ActionEvent x){
  //obtener botón
  Button b=(Button)x.getSource();
  //obtener primer caracter de texto del botón
  char c=b.getLabel().charAt(0);
  //atender boton (y actualizar visor)
  String s;//nuevo contenido de visor
  switch(c){
  case 'Q': System.exit(0);
  case 'C': s=""; break;
  case 'S': s="-"; break;
  case '.': s=visor.getText()+"."; break;
  default:  s=visor.getText()+c; break;//dígitos
```