Resumen-Formulario Control 2

Clases y Objetos

Una clase es un tipo de dato. Una cosa es construir la clase y la otra es utilizarla.

1. Construir una clase:

Una clase siempre tiene 3 partes:
- **Variables de instancia:** variables que caracterizan a la clase
- **Constructor:** construye un objeto de la clase.
- **Métodos:** funciones de la clase. Sirven para mostrar lo que el objeto puede hacer o lo que se puede hacer con el objeto.

Ejemplo:

```
public class Caja{

  //variables de instancia
  double x,y,z;
  String contenido;

  //constructor
  Public Caja(double x,double y,double z){

    //aquí preocúpense de asignar un valor
    //para todas las variables de instancia
    this.x=x;this.y=y;this.z=z;contenido=null;

    //hay 2 variables x, las de la caja
    //y las que vienen de parámetros al constructor
    //para referirse a las variables de instancia
    //se utiliza "this.x", sino se utiliza this,
    //la variable x se refiere a la de los parámetros.
    //otra forma es cambiarle el nombre a los parámetros

    //public Caja(double u,double v,double w){
    // x=u;y=v;z=w;contenido=null;}
  }

  //métodos
  public String getContenido(){
    String aux=contenido;
    contenido=null;
    return aux;
  }

  public void setContenido(String x){
    contenido=x;
  }

  public boolean estaVacia(){
    return contenido==null;
  }
}
```

2. Utilizar una clase:

Una clase es un tipo de dato por lo que se puede utilizar en programas funciones e incluso en otras clases.
Para utilizar una clase se crea un objeto de la clase con "new".

Ejemplo:

```
public static void main(String[] arg){
  Caja c=new Caja();
  c.setContenido("papel");
  U.println("contenido="+ c.getContenido());

  if (c.estaVacia())
    U.println("la caja esta vacia");
  else
    U.println("la caja no está vacia");
}
```

Herencia

Ejemplo:

```
public class CajaCubica extends Caja{
  public CajaCubica(double a){
    //se llama al constructor de la clase
    //madre Caja con "super"
    super(a,a,a);
  }
}
```

La clase CajaCubica tiene todos las variables de instancia y métodos de la clase Caja. Es necesario escribir un nuevo constructor. Es posible agregar más variables de instancia y métodos a la clase CajaCúbica. Si se vuelve a escribir un método del mismo encabezado que uno de la clase Caja, este reemplaza al anterior. Utilizar extends permite códigos como:

```
Caja c=new CajaCubica(20);
```

Obs: esto es lo básico y más importante de herencia. También están los conceptos de clase abstracta e interface.

Archivos

Para trabajar con archivos, Java proporciona clases y objetos especiales.

Para leer:

```
BufferedReader bf= new BufferedReader(new
FileReader("origen.txt"));

String linea="";
While ((linea=bf.readLine())!=null){
  //se itera para cada línea del archive
  //trabajar con las líneas del archivo
  //con los métodos de String
}

bf.close();
```

Para escribir

```
PrintWriter pw=new PrintWriter(new
FileWriter("destino.txt"));

pw.println("esta línea de imprime en el archivo");
String línea="esta es la segunda línea del archivo";
pw.println(linea);
int n=3;
pw.println("línea n="+n+" y última línea");

pw.close();
```

Transformaciones números <-> Strings

```
double d=Double.parseDouble("9.34");
int n=Integer.parseInt("56");
String s="n="+n+" d="+d; // luego s="n=56 d=9.34"
```

Ventanas

Para trabajar con ventanas, Java proporciona clases y objetos especiales.
Una foma fácil de utilizar ventanas es:

```
public class Ventana extends Frame implements
ActionListener{
  //las componentes de la ventana se usan
  //como variables de instancia
  private Label lab=new Label("texto fijo");
  private TextField tex=new TextField("texto usuario");
  private Button but=new Button("texto del boton");

  //en el contructor se construlle la ventana
  public Ventana(){
    super("Nombre de la ventana");

    this.setLayout(new BorderLayout());
    this.setSize(300,300);
    this.add("North",tex); tex.addActionListener(this);
    this.add("Center",but); but.addActionListener(this);
    this.add("South",lab);
  }
  public void actionPerformed(ActionEvent x){
    //este método se ejecuta cada vez que ocurre
    //una acción en la ventana
    //las acciones ocurren cuando se presiona un
    //botón o 'enter' dento de un campo de texto.

    if (x.getSource()==bot || x.getSource()==tex){
      tex.setText(lab.getText()+tex.getText());
      //en general aqui van las instrucciones a
      //realizar cuando se presiona bot o se
      //presiona 'enter' dentro de tex.
    }
    //si hay varias acciones posibles poner un if
    //para cada una de ellas.
  }

  //se agrega el programa principal
  public static void main(String[] arg){
    //se crea el objeto Ventana y se muestra
    //su contenido con show()
    (new Ventana()).show();
  }
}
```

Dibujos

Para dibujar se utiliza la componente Canvas. Este elemento es un campo o una tela de dibujo.
Ejemplo: se dibuja un circulo de 100x100 pixeles al presionar dibujar.

```
public class Dibujo extends Frame implements ActionListener
{
  private Canvas cv=new Canvas();
  private Button dib=new Button("dibujar");

  public Dibujo(){
    super("Ventana de dibujo");

    cv.setSize(200,200);

    this.setLayout(new BorderLayout());
    this.setSize(230,230);
    this.add("Center",dib); dib.addActionListener(this);
    this.add("South",cv);
  }
  public void actionPerformed(ActionEvent x){
    if (x.getSource()==dib){
      //el método getGraphics retorna el
      //pincel de la tela. Con él se puede dibujar.
      this.dibujar(cv.getGraphics());
    }
  }
  public void dibujar(Graphics g){
    //se dibuja un cuadro blanco de fondo
    g.setColor(Color.white);
    g.fillRect(0,0,200,200);

    //se dibuja la silueta de un circulo color azul
    //que empiza en el punto (50,50) de
    //ancho 100 y alto 100
    g.setColor(Color.blue);
    g.drawOval(50,50,100,100);
  }

  public static void main(String[] arg){
    (new Dibujo()).show();
  }
}
```

Observación: la coordenada x se mide hacia la derecha y la coordenada y se mide hacia abajo. Es decir, el punto (0,0) del Canvas está en la esquina superior izquierda.

Métodos de Graphics para dibujar:

setColor(Color.red)	Asigna el color rojo al pincel. Se puede acceder a los colores básicos con Color.(nombre en ingles). Ej: Color.green, Color.orange, etc…
fillRect(x0,y0,ancho,alto)	Dibuja un rectángulo pintado empezando en la posición (x0,y0) del ancho y alto especificado.
drawRect(x0,y0,ancho,alto)	Dibuja la silueta de un rectángulo. Especificaciones análogas a fillRect.
fillOval(x0,y0,ancho,alto)	Dibuja una elipse que empieza en el punto (x0,y0) y del ancho y alto especificados.
drawOval(x0,y0,ancho,alto)	Dibuja solo la silueta de la elipse. Mismas indicaciones.
drawLine(x0,y0,x1,y1)	Dibuja una línea entre los puntos (x0,y0) y (x1,y1).
drawString(palabra,x0,y0)	Escribe el String palabra en la posición (x0,y0).

Ejemplo2: también se puede dibujar implementando el método paint() de la clase que extiende a Frame. Pero en la ventana solo estará el dibujo. Es decir, no se podrán añadir otras componentes.

```
public class Dibujo2 extends Frame {
  public Dibujo2(){
    super("Ventana de dibujo");
    this.setSize(200,200);
  }
  public void paint(Graphics g){
    g.setColor(Color.black);
    g.fillRect(0,0,100,200);

    g.setColor(Color.yellow);
    g.fillRect(100,0,100,200);
  }
  public static void main(String[] arg){
    (new Dibujo2()).show();
  }
}
```