/* HERENCIA MÚLTIPLE
Interfaz: é un contrarto entre dúas clases. Unha clase comprométese a implementar os métodos que se din na interfaz, e ten que implementar polo mínimo eses métodos
(polo que se poden consensuar os nomes e tipos de argumentos, os tipos devoltos....)
As interfaces :
- Non teñen constructores
- Non teñen atributos
- Os métodos non teñen corpo e non son abstractos
- Os únicos atributos que poden ter son CONSTANTES (final e estática)

ATRIBUTOS DE CLASE (constantes). Son atributos que especifican as clases non as instancias.
ATRIBUTOS DE INSTANCIA. Estes últimos son os que non pode ter unha interfaz, pero si unha clase abstracta.

+ ¿Por qué se di que java ten herencia múltiple? Porque unha clase pode implementar varias interfaces, ou unha interfaz e ter herencia.
+ ¿Cal é a diferencia entre unha clase abstracta que so ten metodos abstractos e unha interfaz?*/

```
//Interfaz
public interface Aritmetica{
    public static final float PI =3.1416;
    public int suma(int x, int y);
    public int resta(int x, int y);
    public int multiplicacion(int x, int y, int z);
    public int divisionEnteira(int x, int y);
}
//Clase que implementa a interfaz e hereda da clase Matematica
public class MinhaMatematica implements Aritmetica extends Matematica{
    public int suma(int x, int y){
        return x+y;
    }
    public int resta(int x, int y){
        return x-y;
    }
    public int multiplicacion(int x, int y, int z){
        return x*y;
    }
    public int divisionEnteira(int x, int y){
        if (y!=0) return x/y;
        else return -1;
    }
}
//Clase que usa a interfaz, os seus métodos, etc.
public class InterfazGrafica{
    public InterfazGrafica (Aritmetica a){...};
}

//Na clase principal.
public void main (String[] arg){
    MinhaAritmetica mm=new MinhaAritmetica();
    mew InterfazGrafica((Aritmetica) mm);
}
```

/* XENERALIZACIÓN (dentro de polimorfismo)

```
Sobrecarga de argumentos variable.
*/
//Para sumar diferentes número de enteiros poderíamos facer:
int suma (int x, int y)
int suma (int x, int y, int z)
int suma (int [] x);

//En lugar de todo iso, a mellor solución é:
int suma (int x, int y, int...z){
    int resultado = x+y;
    //Na implementación trátase como un array
    for (int i=0; i<z.length; i++) //Se non hai terceiro argumento non entra aquí xa que z.length==0
        resultado= resultado + z[i];
    return resultado;
}

// Dende fora vese como se fixésemos "polimorfismo infinito". Con este método podemos facer:
int r;
r=suma (1,2);
r=suma(1,2,3,4,5);

/*XENERALIZACIÓN. Parametrizar
A partir das versión 1.5 e 1.6 de Java*/
class Aritmetica <PAR>{
    public PAR suma(PAR x, PAR y){
        return x+y;
    }
}

main (String[] args){
    Aritmetica <Integer> a= new Aritmetica<float>();
}
```

```
/*EXCEPCIÓNS
Sirven para detectar e tratar erros que suceden durante a execución dun programa (os máis complicados de detectar).
Utilízanse para separar o código de tratamento de erros do código do algoritmo.
Poderíamos separar en tres grupos:
- Clases de modelado
- Clases de interfaz gráfica
- Clases de excepcións

1. Ocurre, ¿onde?
2. Detéctase onde ten lugar
3. Lánzase
4. Captúrase/trátase
*/
class Propietario{
    private String dni;
    public void setDni(String dni) throws DniIncorrecto, DniCorrecto{
        if (dni.length()!=9)
            throw new DniIncorrecto(dni);
        else
            this.dni=dni;
```

```
}

public class DniIncorrecto extends Exception{
    private String valor;
    public DniIncorrecto(String dni){
        valor = dni;
    }
    public String getValor(){
        return valor;
    }
    public void corrixeDni(){
        if(valor.length()==8)
            valor="0" + valor;
    }
}

pubic class Principal{
    public static void main (String [] args){
    Propietario p=new Propietario();
        try{
            p.setDni("11223344A");
        }catch(DniIncorrecto dniIncorrecto){
            System.out.println("O erro é que o dni " + dniIncorrecto.getValor()
            + " é distinto de 9 caracteres");
            dniIncorrecto.corrixeDni();
            p.setDni(dniIncorrecto.getValor());
        }catch(DniCorrecto dni){...}
}
```

/*INTERFACES GRÁFICAS
Ventana: marco (frame) e fondo (pane). Hai que definir primeiro o marco. A venta e JFrame.
Elementos gráficos:
- Menu: JMenu, JMenuItem,
- Botóns: JButton
- Campo de texto: JTextField
- Separadores: JSplitPane
- Paneis: JPanel
- Etiquetas: JLabel
- Caixas de selección: JComboBox
- Botóns de selección: JCheckBox, JRadioButton
Eventos: accións que os usuarios levan a cabo cos elementos do interfaz gráfica. Os unicos elementos que non xeneran eventos son o JPanel eo JSplit. Para o resto de elementos:
1. Definir
2. Configurar
3. Posicionar
4. Xestionar eventos*/

```
//Exemplo
class PanelConsulta extends Jpanel{
private JLabel nome;
private JButton bOK;
private JTextField campoNome;
private JComboBox poboacions;
private JCheckBox horario;
private JLabel etiquetaHorario;
```

```
}

public class Vent extends JFrame{
    JComboBox meses;
    String[] nomeMeses;
    JTextField nome;

    public vent(){
        nomeMeses=new String[]{"Marzo", "Abril", "Xuño"}
    }
    public void initComponents(){
        meses= new JComboBox(nomeMeses);
        nome=new JTextField();
    }
    public void setupComponents(){...}
    public void layoutComponents(){
        Container con=this.getContentPane();
        FlowLayout fl=new FlowLayout();
        con.setLayout(fl);
        con.add(meses);
        con.add(nome);
    }
    public void addEventHandler(){
        meses.addMouseListener(new mesesHandler(this));
        nome.addActionListener(new mesesHandler(this));
    }
}

public class MesesHandler implements MouseListener, ActionListener{
    Vent ventana;
    public void mousePressed(MouseEvent me){}
    public void mouseExited(MouseEvent me){}
    public void mouseReleased(MouseEvent me){}
    public void mouseEntered(MouseEvent me){}
    public void mouseClicked(MouseEvent me){
        JOptionPane.showMessageDialog(ventana, "Seleccionouse o mes " +
        ventana.getMeses().getSelectedItem(), "Aviso de mes");
    }
    public MesesHandler(Vent v){
        ventana=v;
    }
    public void actionPerformed(ActionEvent ae){
        JOptionPane.showMessageDialog(Ventana, "Escribiuse o nome " + ventana.getNome().
getText(), "Nome escrito");
    }
}



//Outro exemplo
public class Manejador implements ActionListener{
    private Ventana ventanapadre;
    public Manejador(Ventana v){
        ventanapadre=v;
    }
    public void actionPerformedX throws EdadCorrecta{
        if(new Integer(ventana.getEdad().getText())>90) throw EdadCorrecta();
```

```
        else{
            new VentanaPrincipal().setVisible(true);
            v.setVisible(false);
        }
    }
    public void actionPerformed (ActionPeformed(ActionEvent ae){
        try{
            actionPerformedX();
        catch(EdadCorrecta ec){}
    }
}

public class VentanaPrincipal extends JFrame{
    private Ventana ventana;
    public void initComponents(){
        ventana=new Ventana();
    }
    public void layoutComponents(){
        Container cont=this.getContentPane();
        cont.add(ventana, BorderLayout.CENTER);
    }
}

public class EdadCorrecta extends Exception{
    public void informaError(Ventana v){
        JOptionFrame.showDiaglog(v, "Error: la dedad debe estar entre 18 y 90", "Error en
edad");
    }
}


/*EVENTOS
Cando ocurre un evento tipo acción aise executar sempre o método actionPerformed. É un método dunha interfaz. Por tanto, temos que implementar
este método, creando unha
clase que implemente a interfaz. Polo que cando chamemos a object.addActionListener()  pasarémoslle como argumento unha dinstancia da clase que
implementa o interfaz. ActionListener().*/


//Exemplo 3
public class Ventana extends JFrame{
    private JButton but1;
    private JButton but2;
    private JButton but3;
    public void initComponents(){
        but1=new JButton("Button 1");
        but2=new JButton("Button 2");
        but3=new JButton("Button 3");
    }
    public void setupComponents(){
        but1.setAlignementx(Component.CENTER_ALIGNMENT);
        but2.setAlignementx(Component.CENTER_ALIGNMENT);
        but3.setAlignementx(Component.CENTER_ALIGNMENT);
    }
    public void layoutComponents(){
        /*Se quixésemos axustalo a man sería:
        this.getContentPane().setLayout(null);
        but1.setBounds(10,10,50,20);
```

```
        but2.setBounds(10,35,50,20);
        but3.setBounds(10,60,50,20);*/
        Container cont=this.getContentPane();
        BoxLayout bl=new BoxLayout(this, BOX_LAYOUT.Y_AXIS);
        cont.setLayout(bl);
        this.add(but1);
        this.add(but2);
        this.add(but3);
    }
}

public void main (String [] args){
    new Ventana().setVisible(true);
}



//Exemplo 4 (non utiliza paneles para que sexa máis curto)
public class ventana extends JFrame{
    private JButton cancel;
    private JButton ok;
    public void initComponents(){
        ok=new JButton("ok");
        cancel=new JButton("cancel");
    }
    public void SetupComponents(){}
    public void LayoutComponents(){
        //Coloco o elemento no centro, na parte superior
        this.getContentPane().setLayout(new FlowLayout());
        this.getContentPane().add(ok);
        this.getContentPane().add(cancel);
    }
    public void addListeners(){
        //ok.addActionListener(new ManexadorOk()); //Cabeceira: addActionListener(ActionListener e)
        //Listener de tipo accion: engadir unha isntancia dunha clase que implementa
        // o método actionPerformed cando ocurre un evento do tipo acción
        ok.addActionListener(new ManexadorBotons(this));
    }
}
public class ManexadorBotons implements ActionListener{
    private ventana v;
    public void actionPerformed(ActionEvent ae){
        if(ae.getSource().equal(v.getOK()){
            v.getCancel().setVisbile(false);
            JOptionFrame.showDiaglog(v, "Hola mundo", "Saúdo");
        }
        else if (ae.getSource().equal(v.getCancel()){...}
    }
    public ManexadorBoton(Ventana v1){
        v=v1;
    }
}
```