

Interacción de objetos

Creación de objetos que cooperan

UN RELOJ DIGITAL

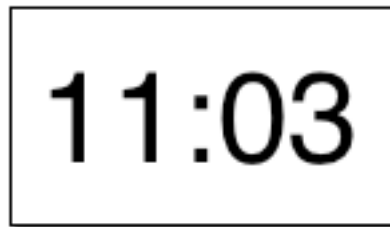
A digital clock display consisting of a white rectangular box with a thin black border. Inside the box, the time "11:03" is displayed in a large, bold, black sans-serif font. The box has a subtle drop shadow, giving it a three-dimensional appearance.

11:03

Abstracción y Modularización

- **La Abstracción** es la habilidad de ignorar los detalles de las partes para enfocar la atención en un nivel más elevado de un problema.
- **La Modularización** es el proceso de dividir un todo en partes bien definidas que interactúan en formas bien definidas. Estas partes pueden construirse y examinarse en forma separada.

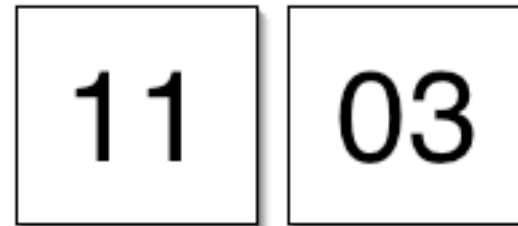
Modularización del visor del reloj



11:03

¿Un visor de uno o cuatro dígitos?

¿O dos visores de dos dígitos?



11 03

Implementación del VisorDeNumeros

```
public class VisorDeNumeros  
  
{  
    private int limite;  
    private int valor;  
  
    El Constructor y los  
    métodos se omiten.  
}
```

Implementación del VisorDeReloj

```
public class VisorDeReloj
{
    private VisorDeNumeros horas;
    private VisorDeNumeros minutos;

    El Constructor y los  
métodos se omiten.
}
```

Diagrama de objetos

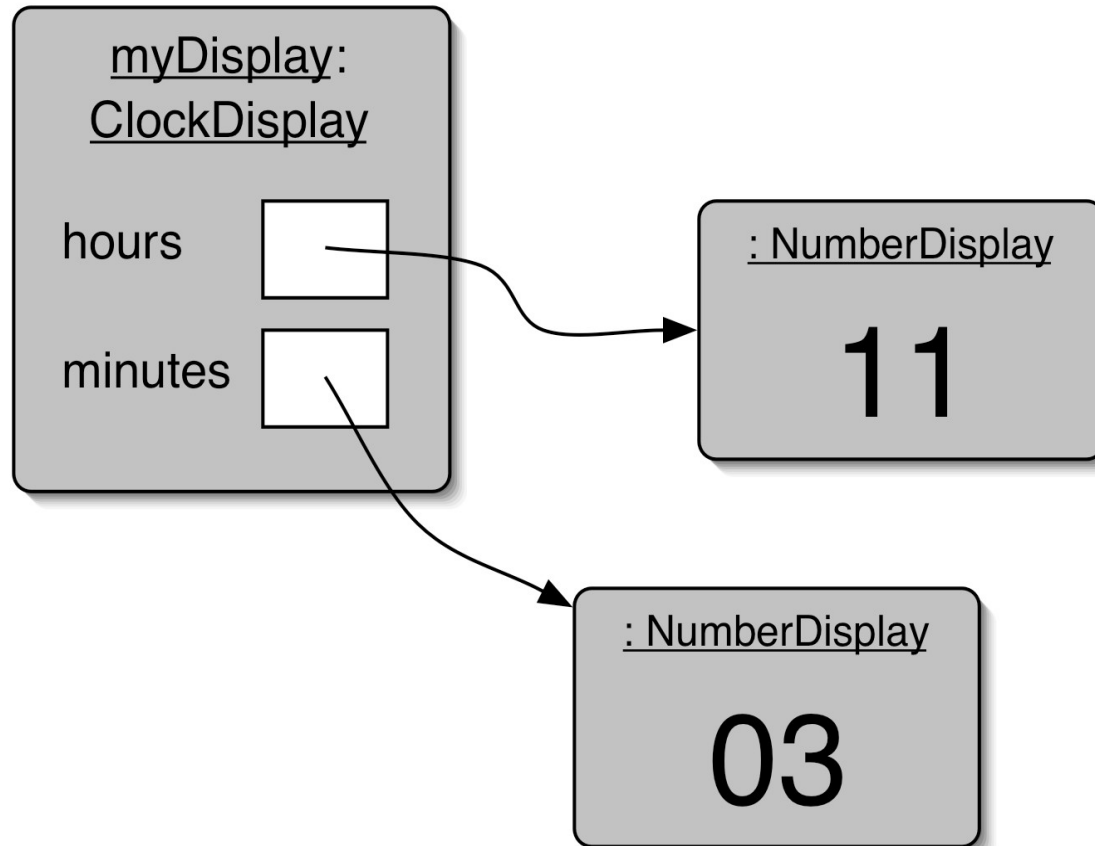
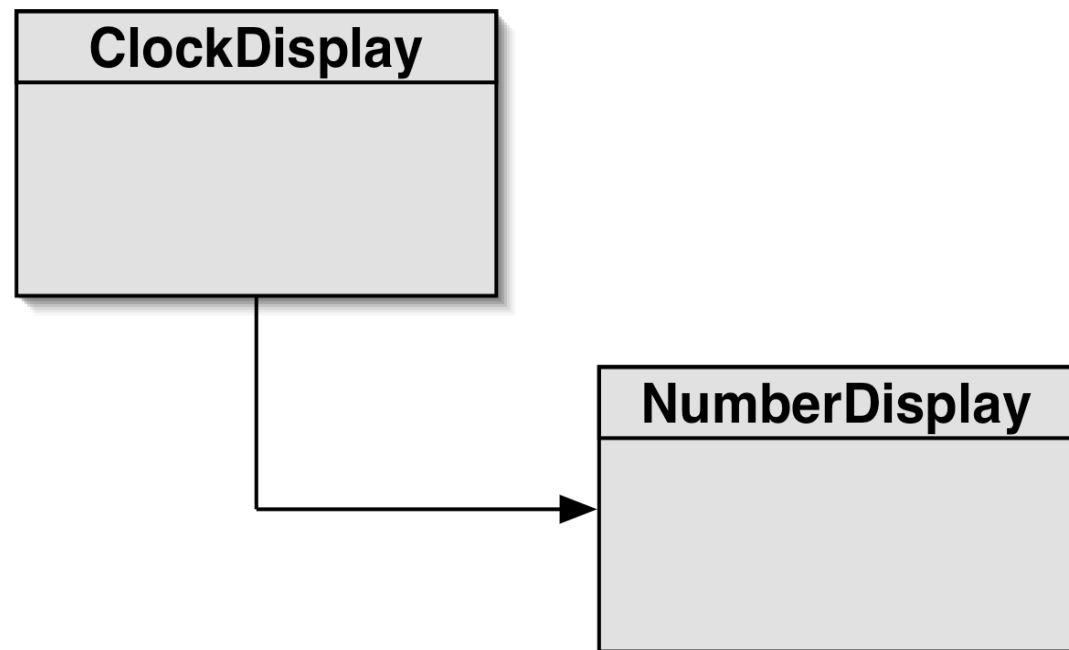


Diagrama de Clases

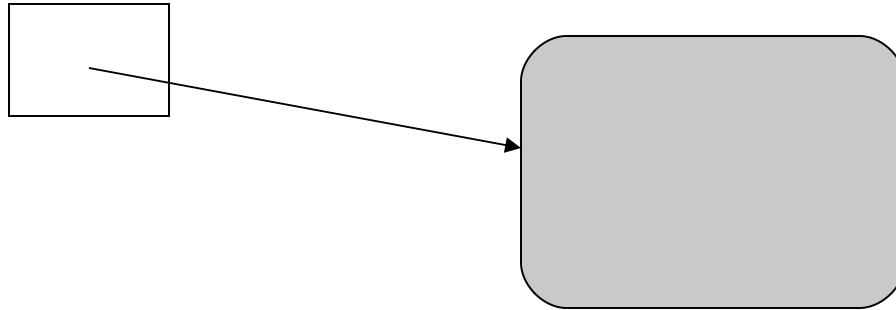


Ej.: 3.1, 3.2, 3.3, 3.4

Tipos primitivos y tipos objeto

AlgunObjeto obj;

tipo de objeto



int i;

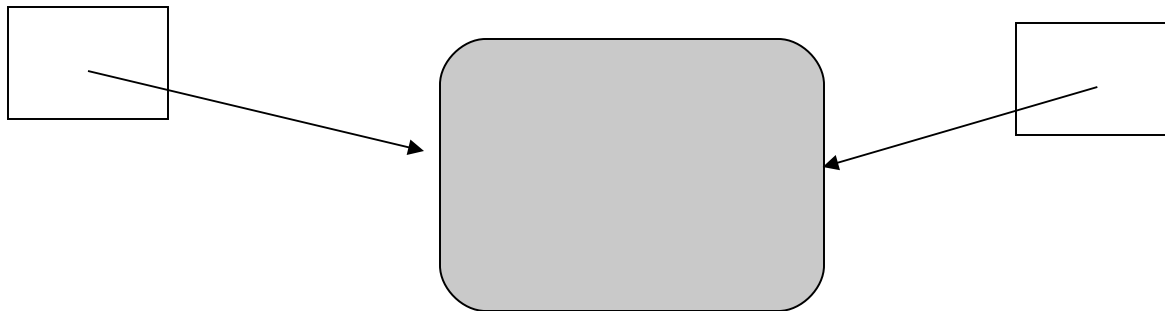
tipo primitivo



Tipos primitivos y tipos objeto

TipoDeObjeto a;

TipoDeObjeto b;



b = a;

int a;

32

int b;

32

Código fuente: VisorDeNumeros

```
Public VisorDeNumeros(int limiteMaximo)
{
    limite = limiteMaximo;
    valor  = 0;
}

public void incrementar()
{
    valor = (valor + 1) % limite;
}
```

Ej.: 3.6, 3.7, 3.8, 3.9, 3.10, 3.11, 3.12

Código Fuente: VisorDeNumeros

```
public String getValorDelVisor()  
{  
    if(valor < 10) {  
        return "0" + valor;  
    }  
    else {  
        return "" + valor;  
    }  
}
```

Ej.: 3.13, 3.14, 3.15, 3.16, 3.17, 3.18

Objetos que crean objetos

```
public class VisorDeReloj
{
    private VisorDeNumeros horas;
    private VisorDeNumeros minutos;
    private String cadVisor;

    public VisorDeReloj()
    {
        horas    = new VisorDeNumeros(24) ;
        minutos  = new VisorDeNumeros(60) ;
        updateDisplay() ;
    }
}
```

Llamadas a Métodos

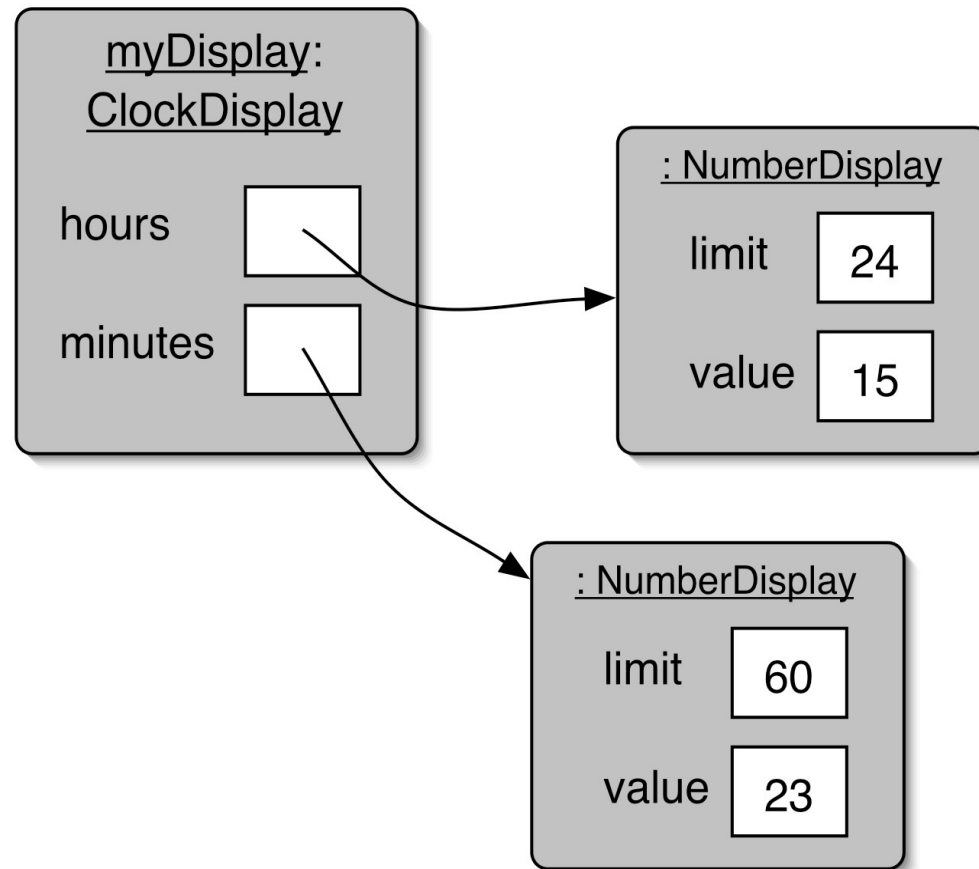
```
public void ticTac()  
{  
    minutos.incrementar();  
    if(minutos.getValor() == 0) {  
        // alcanzó el límite!  
        horas.incrementar();  
    }  
    actualizarVisor();  
}
```

Ej.: 3.28, 3.29, 3.30

Método Interno

```
/**
 * Actualiza la cadena interna que
 * representa el visor.
 */
private void actualizarVisor()
{
    cadVisor =
        horas.getValorDelVisor() + ":" +
        minutos.getValorDelVisor();
}
```


Diagrama de objetos del VisorDeReloj



Objetos que crean objetos

En la clase *VisorDeNumeros*:

```
public VisorDeNumeros(int limiteMaximo);
```

Parámetro formal

En la clase *VisorDeReloj*:

```
horas = new VisorDeNumeros(24);
```

Parámetro real

Llamadas a métodos (1)

- Llamadas a métodos internos

```
actualizarVisor();
```

```
...
```

```
private void actualizarVisor()
```

- Llamadas a métodos externos

```
minutos.incrementar();
```

Ej.: 3.22, 3.23, 3.24, 3.25, 3.26, 3.27

Llamadas a métodos (2)

objeto.nombreMetodo (lista-parámetros)

Ej.: 3.28, 3.29, 3.30, 3.31, 3.32, 3.33,
3.34, 3.35, 3.36, 3.37, 3.38, 3.39,
3.40, 3.41, 3.42, 3.43, 3.44

Conceptos

- Abstracción
- Modularización
- Las clases definen tipos
- Diagrama de clases
- Diagrama de objetos
- Referencias a objetos
- Tipos primitivos
- Tipos objeto
- Creación de objetos
- Sobrecarga
- Llamada a métodos internos/externos
- Depurador