

## Capítulo 3

# Interação entre objetos



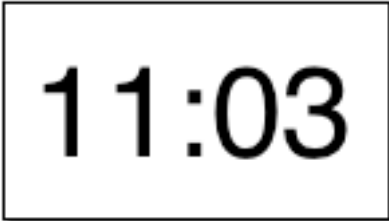
# Um relógio digital

11:03

# Abstração e modularização

- **Abstração** é a capacidade de ignorar detalhes de partes para focalizar a atenção em um nível mais elevado de um problema.
- **Modularização** é o processo de dividir um todo em partes bem definidas, que podem ser construídas e examinadas separadamente, e que interagem de maneiras também bem definidas.


# Modularizando o mostrador do relógio



11:03

Um mostrador  
de quatro dígitos?

Ou mostradores de  
dois dígitos?



11



03

# Implementação – NumberDisplay

```
public class NumberDisplay
{
    private int limit;
    private int value;

    Construtores e
    métodos omitidos.
}
```

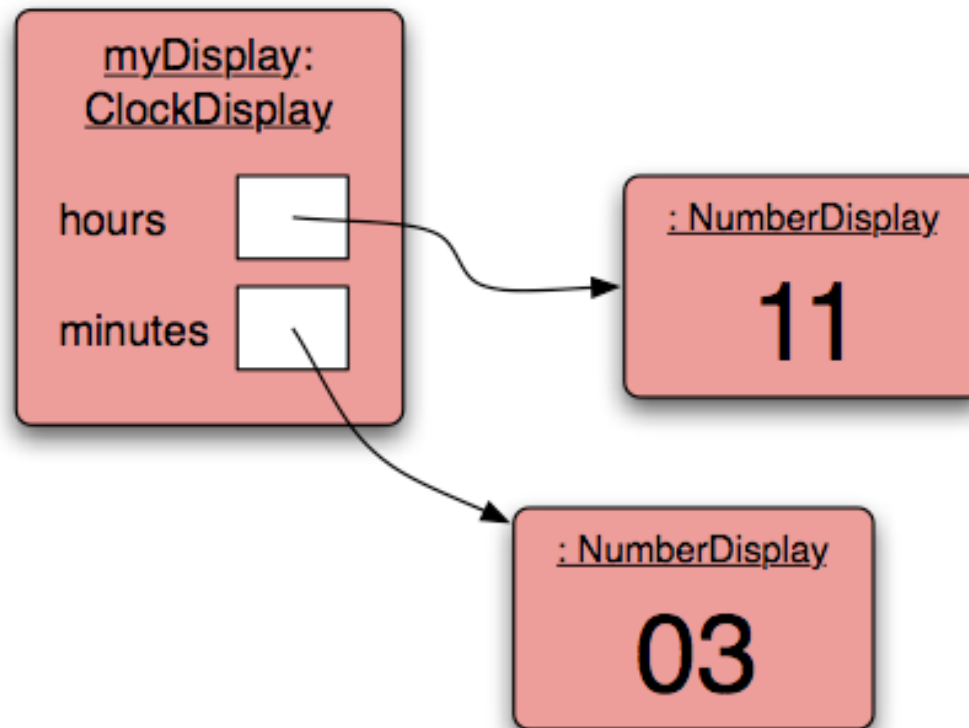
# Implementação – ClockDisplay

```
public class NumberDisplay
{
    private int limit;
    private int value;

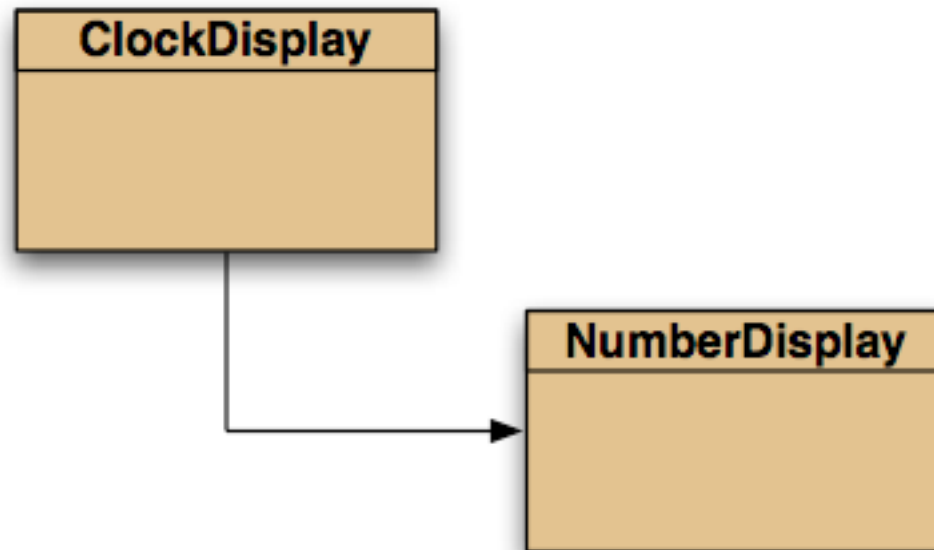
    Construtores e
    métodos omitidos.
}
```



# Diagrama de objetos



# Diagrama de classes

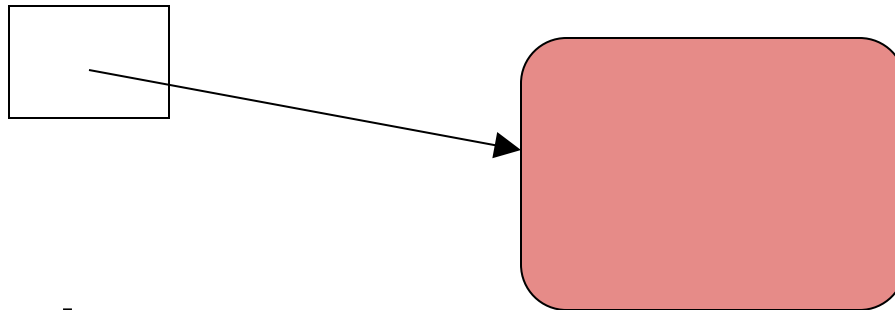




# Tipos primitivos *versus* tipos de objeto

`SomeObject obj;`

tipo de objeto



`int i;`

32

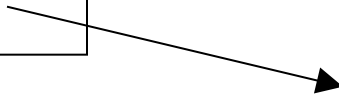
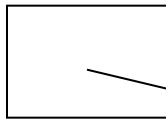
tipo primitivo

# Questionário: qual é o resultado?

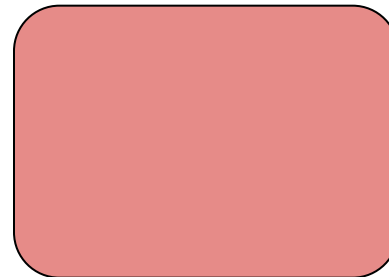
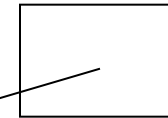
- ```
int a;  
int b;  
a = 32;  
b = a;  
a = a + 1;  
System.out.println(b);
```
- ```
Person a;  
Person b;  
a = new Person("Everett");  
b = a;  
a.changeName("Delmar");  
System.out.println(b.getName());
```

## Tipos primitivos *versus* tipos de objeto

`ObjectType a;`



`ObjectType b;`



---

`b = a;`

`int a;`



`int b;`



# Código-fonte: NumberDisplay

```
public NumberDisplay(int rollOverLimit)
{
    limit = rollOverLimit;
    value = 0;
}

public void increment()
{
    value = (value + 1) % limit;
}
```

# O operador de módulo

- O operador de 'divisão' (/), quando aplicado ao operando int, retorna o *resultado* de uma *divisão de inteiro*.
- O operador de 'módulo' (%) retorna o *resto* de uma divisão de inteiro.
- Por exemplo, em geral:  
 $17 / 5 = \text{resulta em } 3, \text{ restando } 2$
- Em Java:  
 $17 / 5 = 3$   
 $17 \% 5 = 2$

# Questionário

- Qual é o resultado da expressão  $(8 \% 3)$  ?
- Quais são todos os possíveis resultados da expressão  $(n \% 5)$  ?



# Código-fonte: NumberDisplay

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

# Conceitos

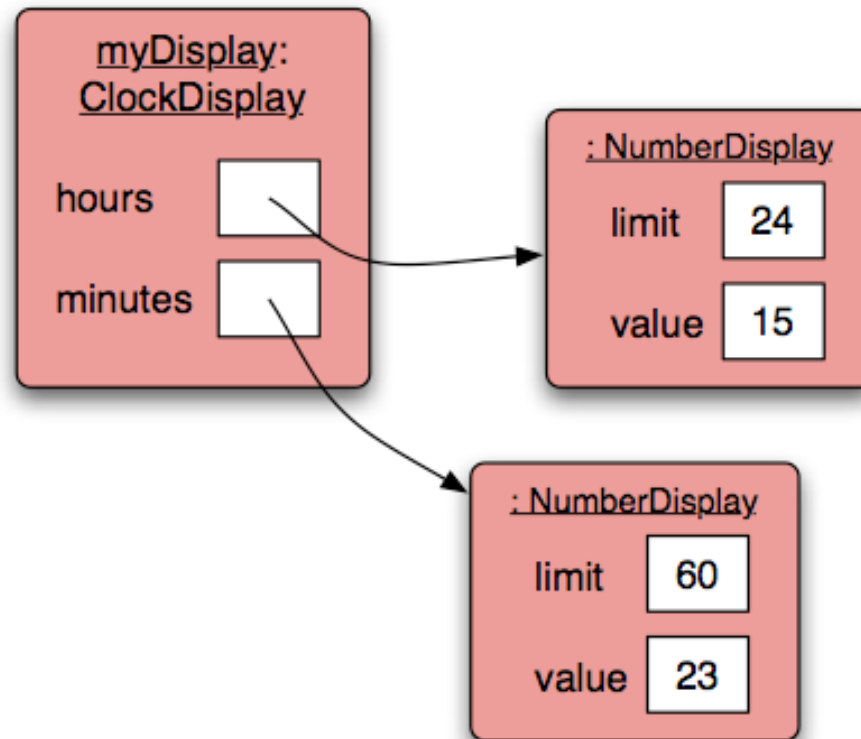
- abstração
- modularização
- classes definem tipos
- diagrama de classes
- diagrama de objetos
- referências de objeto
- tipos primitivos
- tipos de objeto

# Objetos que criam objetos

```
public class ClockDisplay
{
    private NumberDisplay hours;
    private NumberDisplay minutes;
    private String displayString;

    public ClockDisplay()
    {
        hours = new NumberDisplay(24);
        minutes = new NumberDisplay(60);
        updateDisplay();
    }
}
```

## Diagrama de objeto ClockDisplay



# null

- `null` é um valor especial em Java.
- Todas as variáveis de objeto são inicializadas como `null`.
- Você pode atribuir e testar a existência de `null`:

```
private NumberDisplay hours;
```

```
if(hours == null) { ... }
```

```
hours = null;
```

# Objetos que criam objetos

na classe NumberDisplay:

```
public NumberDisplay(int rollOverLimit);
```

*parâmetro formal*

na classe ClockDisplay:

```
hours = new NumberDisplay(24);
```

*parâmetro real*



# Chamada de método

```
public void timeTick()  
{  
    minutes.increment();  
    if(minutes.getValue() == 0) {  
        // ele acabou de retornar!  
        hours.increment();  
    }  
    updateDisplay();  
}
```

## Método interno

```
/**  
 * Atualiza a string interna que  
 * representa o mostrador.  
 */  
private void updateDisplay()  
{  
    displayString =  
        hours.getDisplayValue() + ":" +  
        minutes.getDisplayValue();  
}
```

# Chamadas de método (1)

- chamadas de método interno:

```
updateDisplay();
```

```
...
```

```
private void updateDisplay()
```

- chamadas de método externo:

```
minutes.increment();
```

# Chamadas de método (2)

*object . nomeDoMétodo( lista-de-parâmetros )*

# Conceitos

- criação de objeto
- sobrecarga
- chamada de método interno/externo
- depurador