Conception objet en Java avec BlueJ

une approche interactive

3. Interactions entre objets

Créer des objets coopérant

David J. Barnes, Michael Kölling

version française: Patrice Moreaux

Une horloge numérique

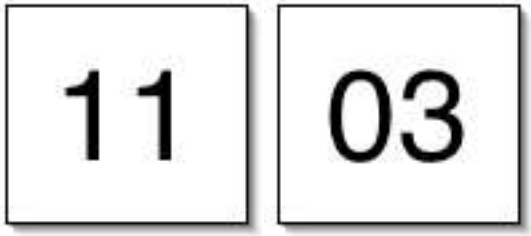
Abstraction et modularisation

- L'a**bstraction** consiste à ignorer les détails des divers éléments d'un problème et se placer à un plus haut niveau.
- La «**modularisation**» est le processus de division d'un tout en plusieurs parties bien définies, que l'on peut construire et analyser séparément et qui interagissent de manière bien précise.

Modulariser l'affichage numérique

Un afficheur à quatre chiffres?

Ou
deux afficheurs à deux chiffres?



Implantation (1)
NumberDisplay

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

    //*** Constructeur et
    //    méthodes non
reproduits.
}
```

Implantation (2)
ClockDisplay

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

    //*** Constructeur et
    //    méthodes non reproduits.
}
```

Diagramme d'objets

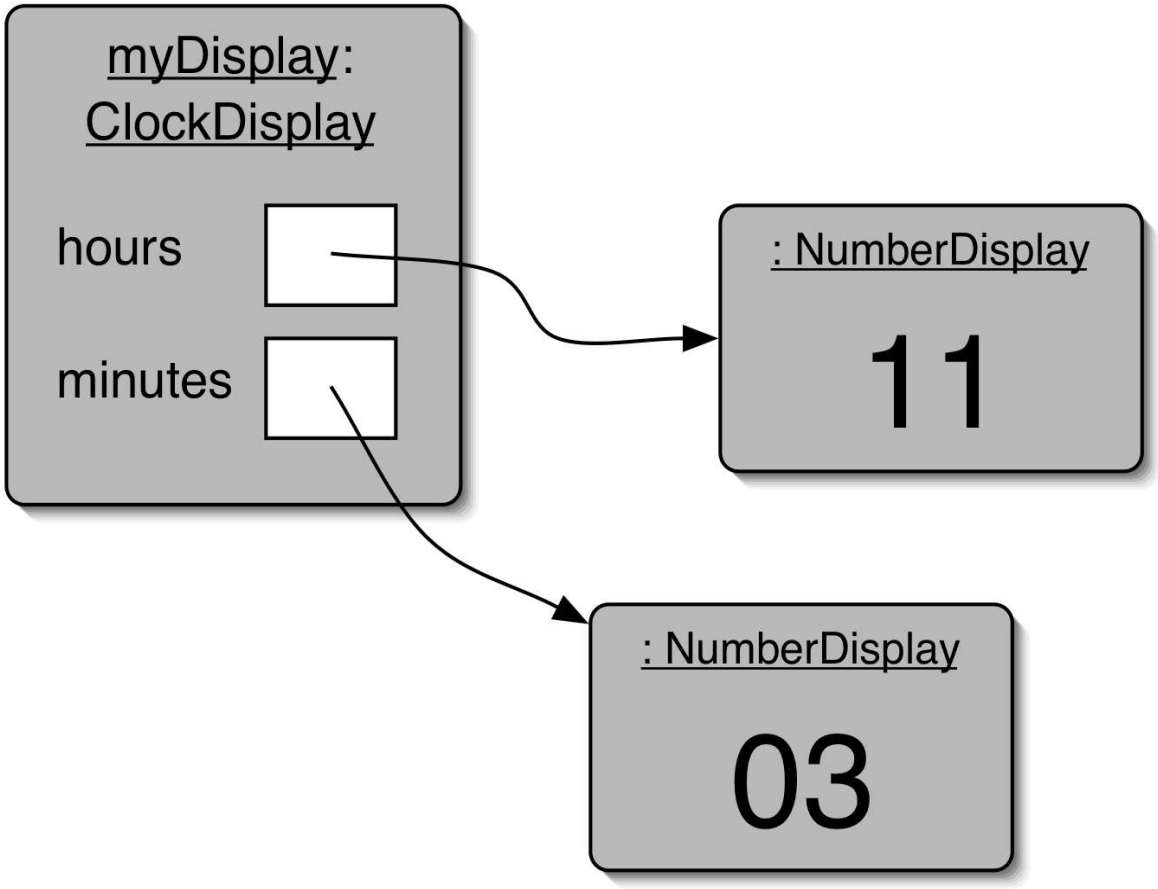
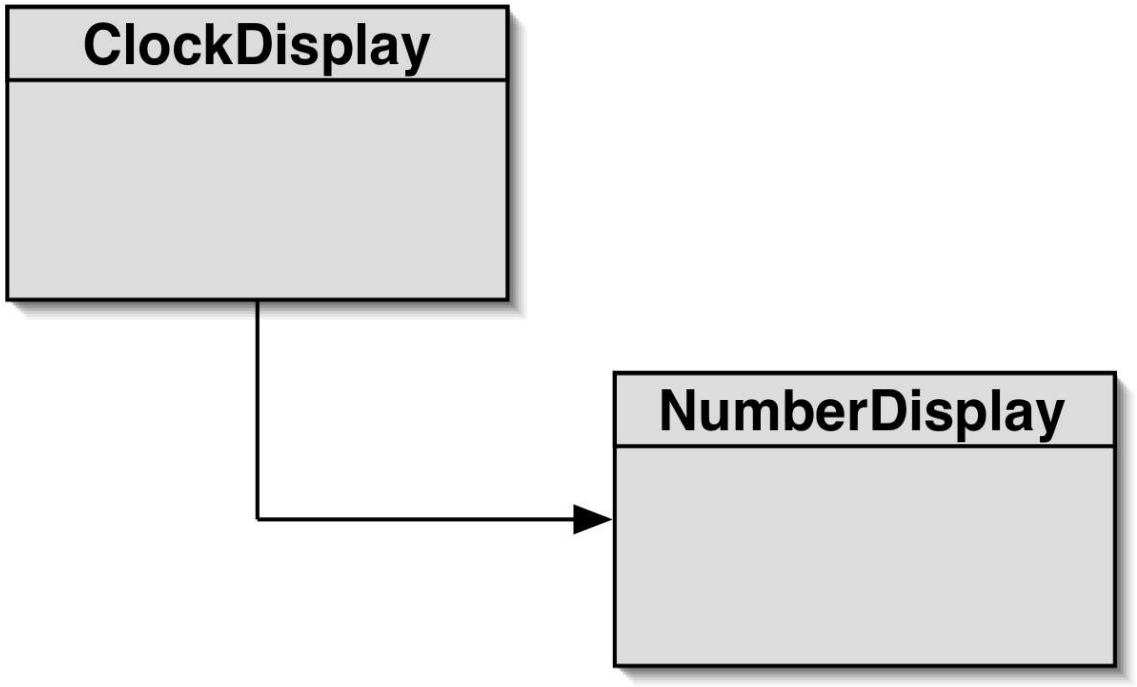


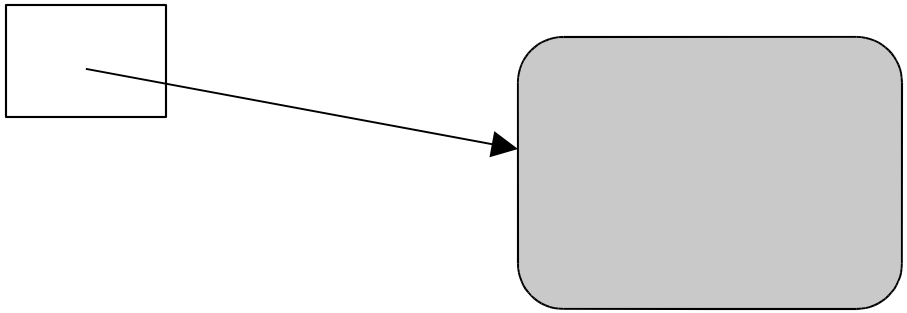
Diagramme de classe



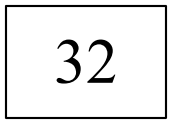
Types primitifs versus types objet (1)

```
SomeObject obj;
```

Type objet

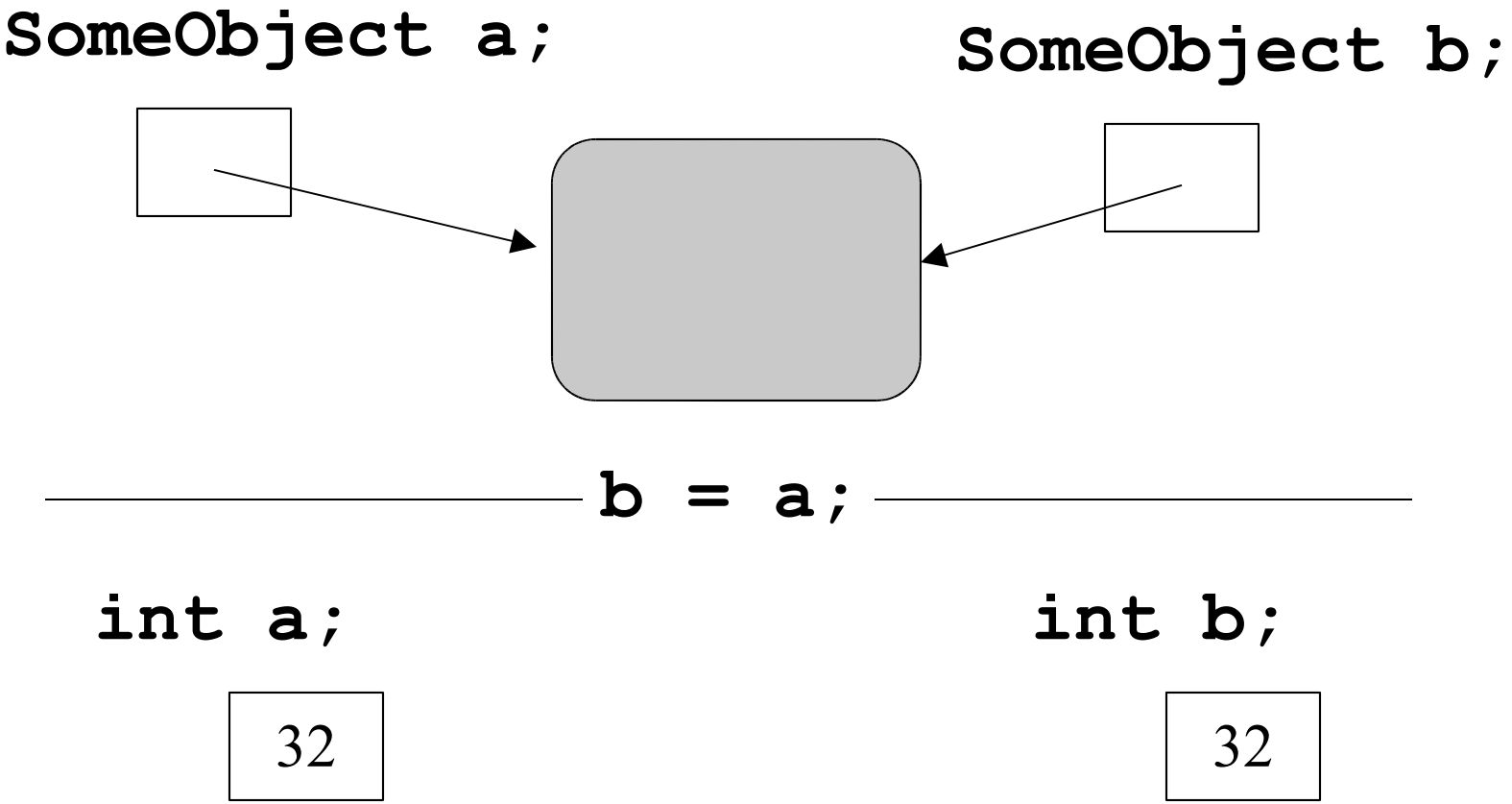


```
int i;
```



Type primitif

Types primitifs versus types objet (2)



Code source: NumberDisplay (1)

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

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

Code source: **NumberDisplay** (2)

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

Objets qui créent des objets

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

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

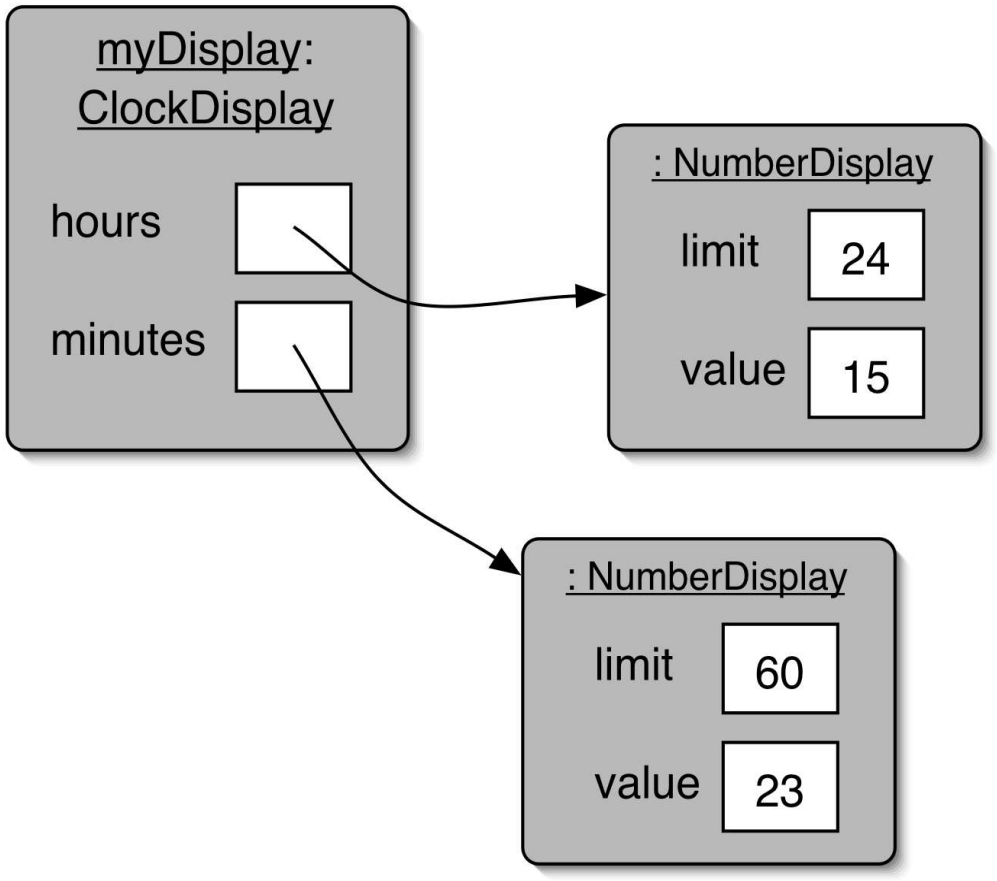
Appel de méthode

```
public void timeTick()
{
    minutes.increment();
    if(minutes.getValue() == 0) {
        // une heure de plus!
        hours.increment();
    }
    updateDisplay();
}
```

Méthode interne

```
/**
 * Mettre à jour la chaîne interne
 * qui représente l'affichage.
 */
private void updateDisplay()
{
    displayString =
        hours.getDisplayValue() + ":" +
        minutes.getDisplayValue();
}
```

Diagramme d'objets de `ClockDisplay`



Objets qui créent des objets

dans la classe **NumberDisplay**:

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

paramètre formel

dans la classe **ClockDisplay**:

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

paramètre effectif

Appels de méthode (1)

- appel de méthodes internes

```
updateDisplay();
```

```
private void updateDisplay()
```

- appel de méthodes externes

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

Appels de méthode (2)

```
objet.nomDeMethode(

  liste_de_paramètres )
```

Concepts

- abstraction
- modularisation
- les classes définissent des types
- diagramme de classe
- diagramme d'objets
- références à objets

- types primitifs
- types objet
- création d'objet
- surcharge
- appel de méthode interne/externe
- débogueur

Sommaire général