

Object Interaction

Creating cooperating objects

Abstraction

- **Abstraction** is the ability to ignore details of parts to focus attention on a higher level of a problem.

Modularization

- **Modularization** is the process of dividing a whole into well-defined parts, which can be built and examined separately, and which interact in well-defined ways.

A digital clock

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 3D appearance.

11:03

Modularizing the clock display

11:03

One four-digit display?

Or two two-digit displays?

11 03

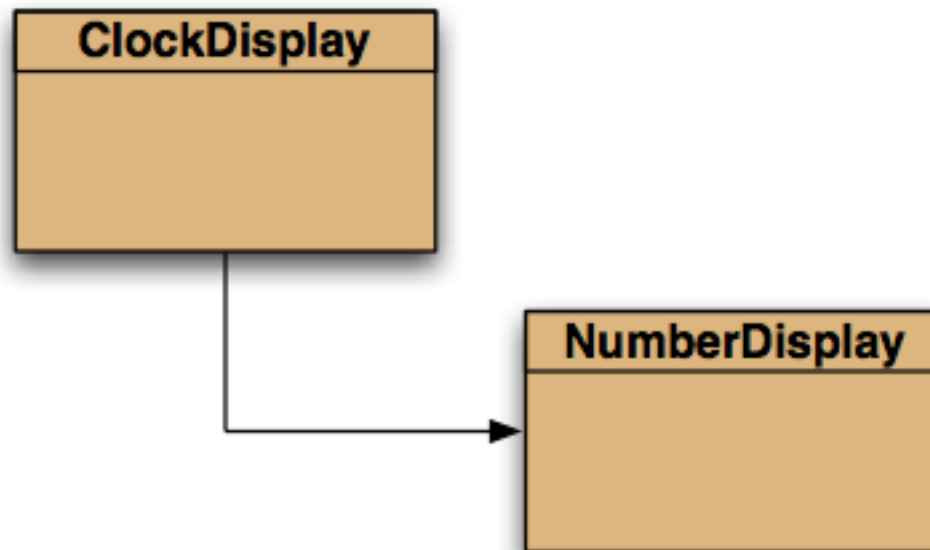
Implementation - NumberDisplay

```
public class NumberDisplay {  
  
    private int limit;  
    private int value;  
  
    Constructor and  
    methods omitted.  
  
}
```

Implementation - ClockDisplay

```
public class ClockDisplay {  
  
    private NumberDisplay hours;  
    private NumberDisplay minutes;  
  
    Constructor and  
    methods omitted.  
  
}
```

Class diagram



Source code: NumberDisplay

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

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

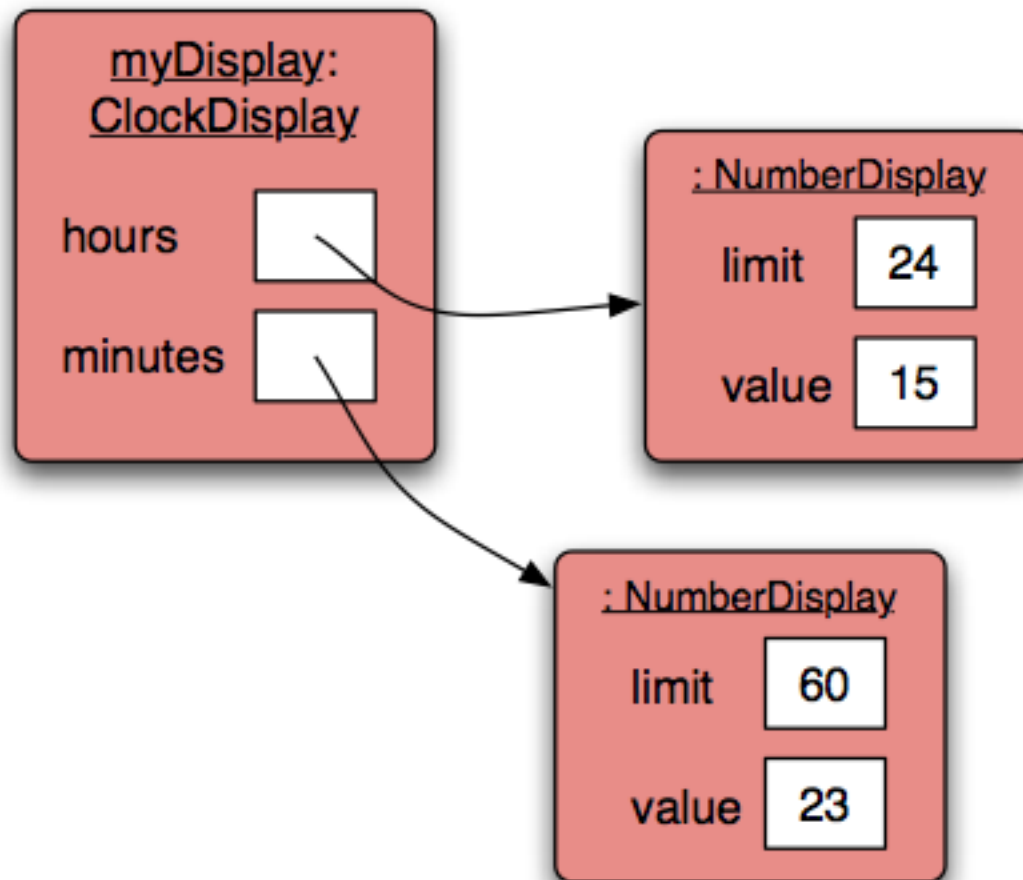
Source code: NumberDisplay

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

Objects creating objects

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

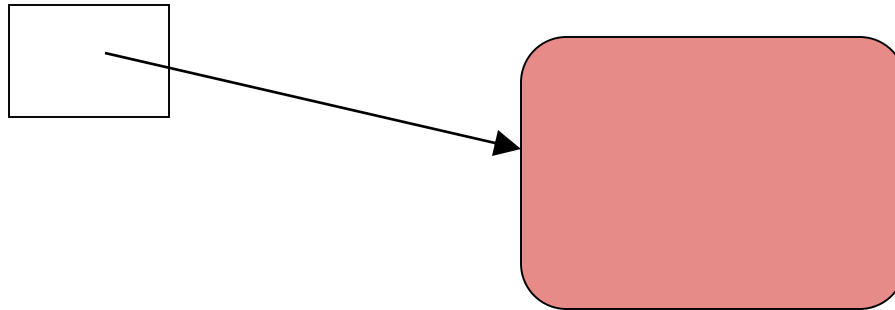
ClockDisplay object diagram



Primitive types vs. object types

SomeClass obj;

object type



int i;

primitive type

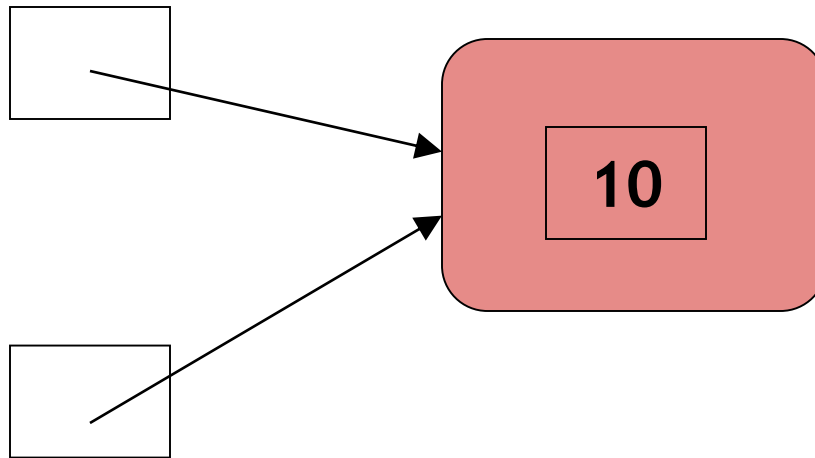


Quiz: What is the output?

- ```
int a;
int b;
a = 10;
b = a;
a = a + 1;
System.out.println(b);
```
- Write a class named 'Int' and try above again.  
What changed?

# Primitive types vs. object types

```
Int a = new Int(10);
```



```
Int b = a;
```

# null

- `null` is a special value in Java
- All object variables are initialized to `null`.
- You can assign and test for `null`:

```
private NumberDisplay hours;
 if (hours == null) { ... }
 hours = null;
```



# Objects creating objects

in class NumberDisplay:

```
public NumberDisplay(int rollOverLimit) {
```

*formal parameter*

in class ClockDisplay:

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

*actual parameter*

# Method calling

```
public void timeTick()
{
 minutes.increment();
 if(minutes.getValue() == 0) {
 // it just rolled over!
 hours.increment();
 }
 updateDisplay();
}
```

# Internal method

```
/**
 * Update the internal string that
 * represents the display.
 */
private void updateDisplay()
{
 displayString =
 hours.getDisplayValue() + ":" +
 minutes.getDisplayValue();
}
```

# Internal Method calls

*methodName ( parameter-list )*

# External Method calls

*object . methodName ( parameter-list )*

# Method calls

- internal method calls

```
updateDisplay();
```

```
...
```

```
private void updateDisplay()
```

- external method calls

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

# Method / Constructor Overloading

- **Overloading:**  
with a different set of parameters:

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

public ClockDisplay(int hour, int minute) {
 hours = new NumberDisplay(24);
 minutes = new NumberDisplay(60);
 setTime(hour, minute);
}
```

# Quiz: is this correct ?!

```
private int value;

public void setValue(int value) {
 value = value;
}
```



# 'this' keyword

```
private int value;
```

```
public void setValue(int value) {
 this.value = value;
}
```

# Concepts

- abstraction
- modularisation
- classes define types
- class diagram
- object diagram
- object references
- primitive types
- object creation
- overloading
- internal/external method call
- this keyword

## شعر امروز

گویند مرا که این همه درد چراست  
وین نعره و آواز و رخ زرد چراست

گویم که چنین مگو که اینکار خطاست  
رو روی مهش ببین و مشکل برخاست