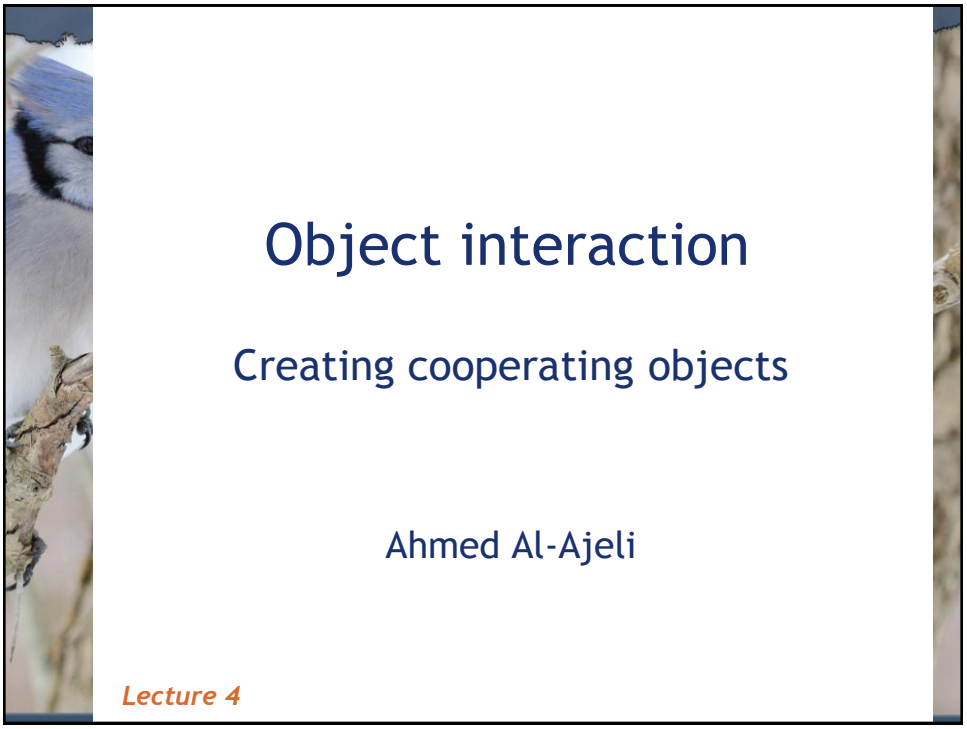
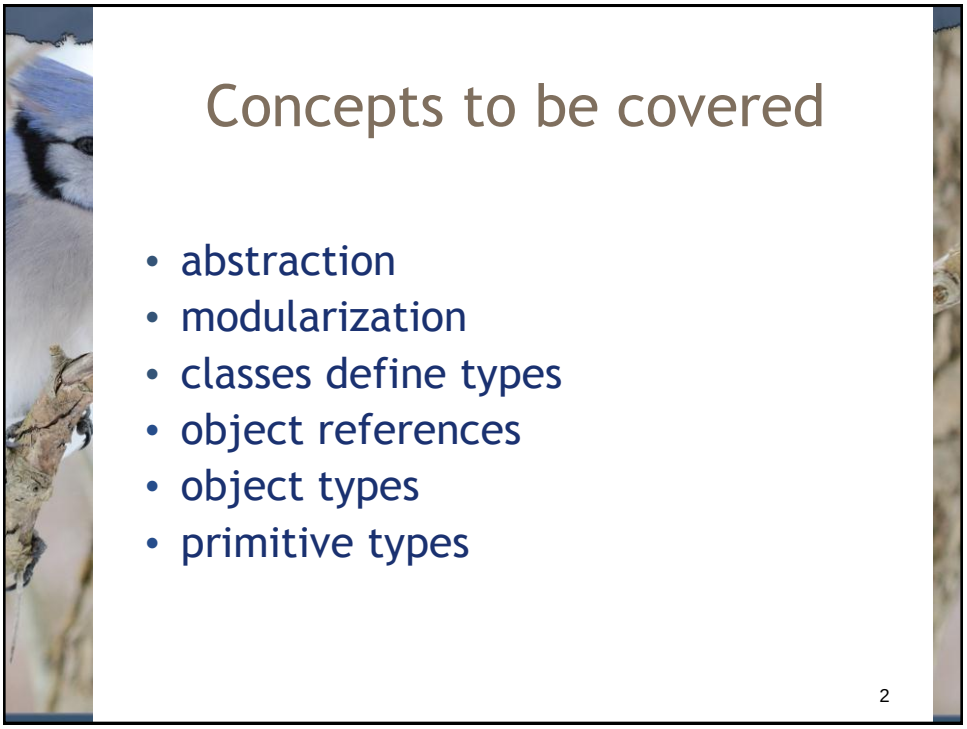


Object interaction

Creating cooperating objects

Ahmed Al-Ajeli

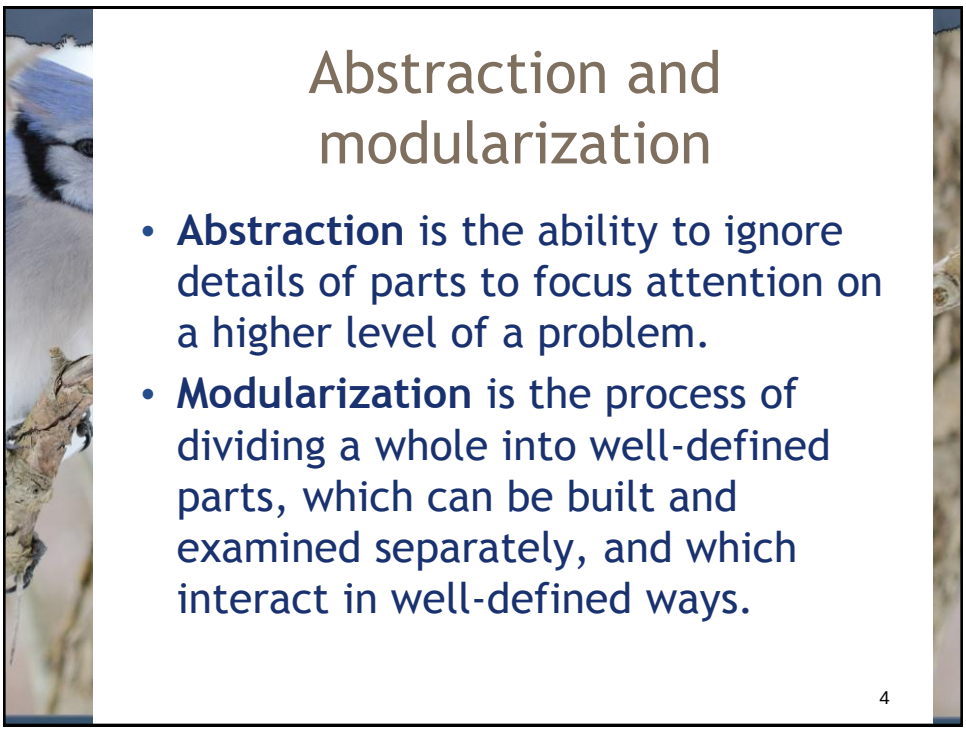
Lecture 4

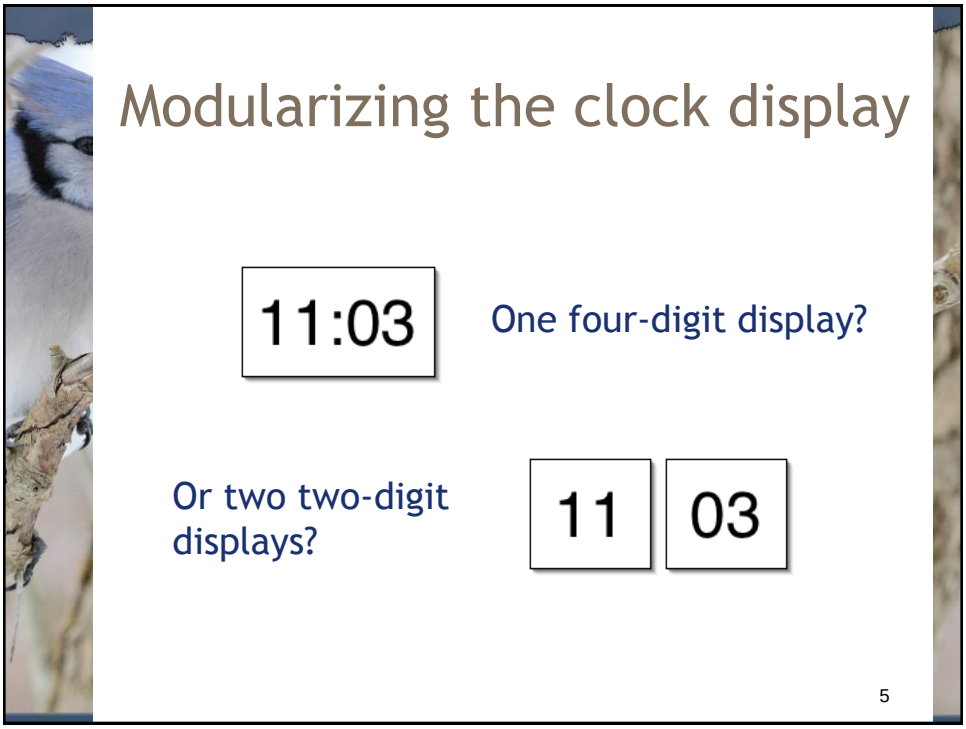


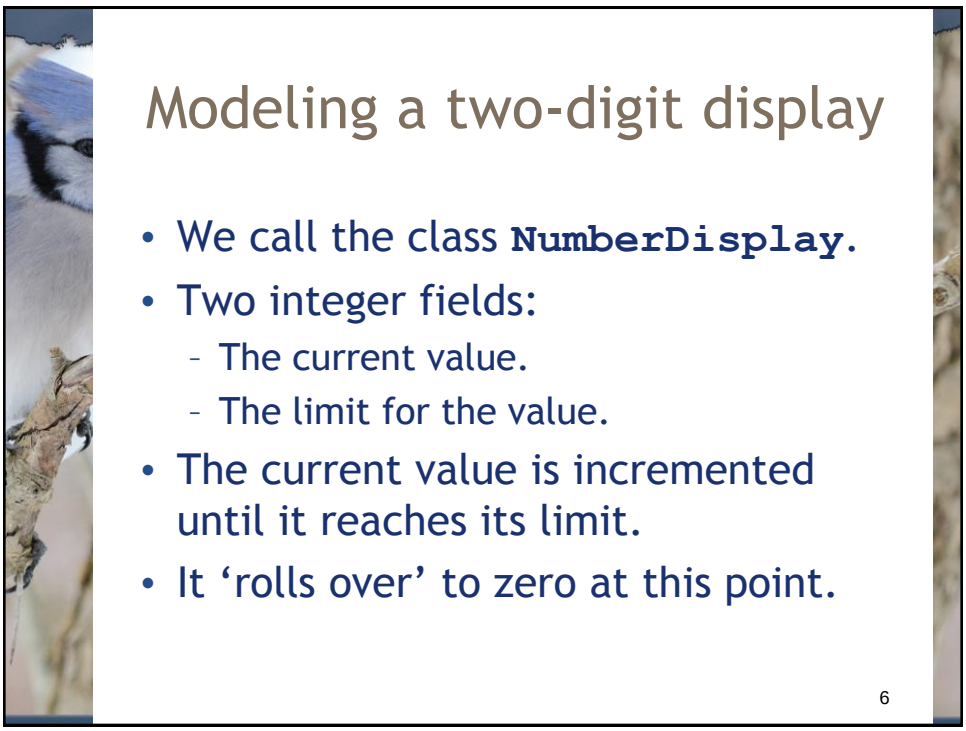
Concepts to be covered

- abstraction
- modularization
- classes define types
- object references
- object types
- primitive types


A digital clock
11:03
3


Abstraction and modularization
• Abstraction is the ability to ignore details of parts to focus attention on a higher level of a problem.
• 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.
4


Modularizing the clock display
11:03
One four-digit display?
Or two two-digit displays?
11
03
5


Modeling a two-digit display
• We call the class NumberDisplay.
• Two integer fields:
– The current value.
– The limit for the value.
• The current value is incremented until it reaches its limit.
• It 'rolls over' to zero at this point.
6

Implementation - NumberDisplay

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

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

7

Accessor and mutator methods

```
public int getValue()
{
    return value;
}

public void setValue(int replacementValue)
{
    if((replacementValue >= 0) &&
       (replacementValue < limit)) {
       value = replacementValue;
    }
}
```

8

Logic operators

- Logic operators operate on boolean values (***true*** or ***false***) and produce a new boolean value as a result. The logical operators:

  && (and) **||** (or) **!** (not)

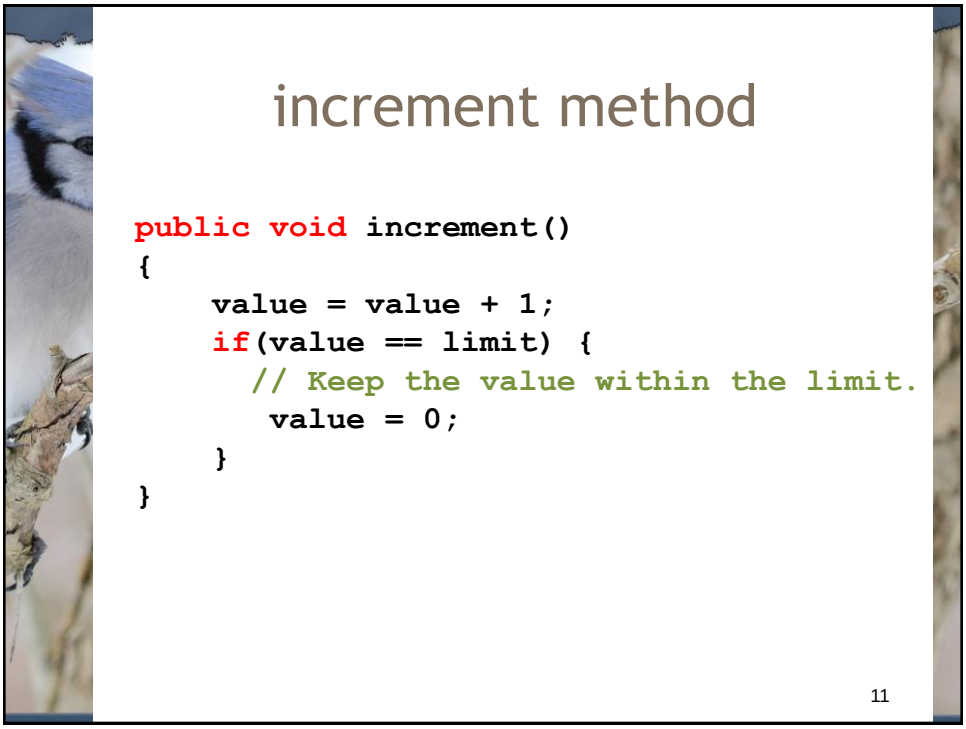
  - **a && b** is ***true*** if both **a** and **b** are ***true***, and ***false*** otherwise.
  - **a || b** is ***true*** if either **a** or **b** or both are ***true***, and ***false*** if they are both ***false***.
  - **!a** is ***true*** if **a** is ***false*** and ***false*** if **a** is ***true***.

9

Source code: NumberDisplay

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

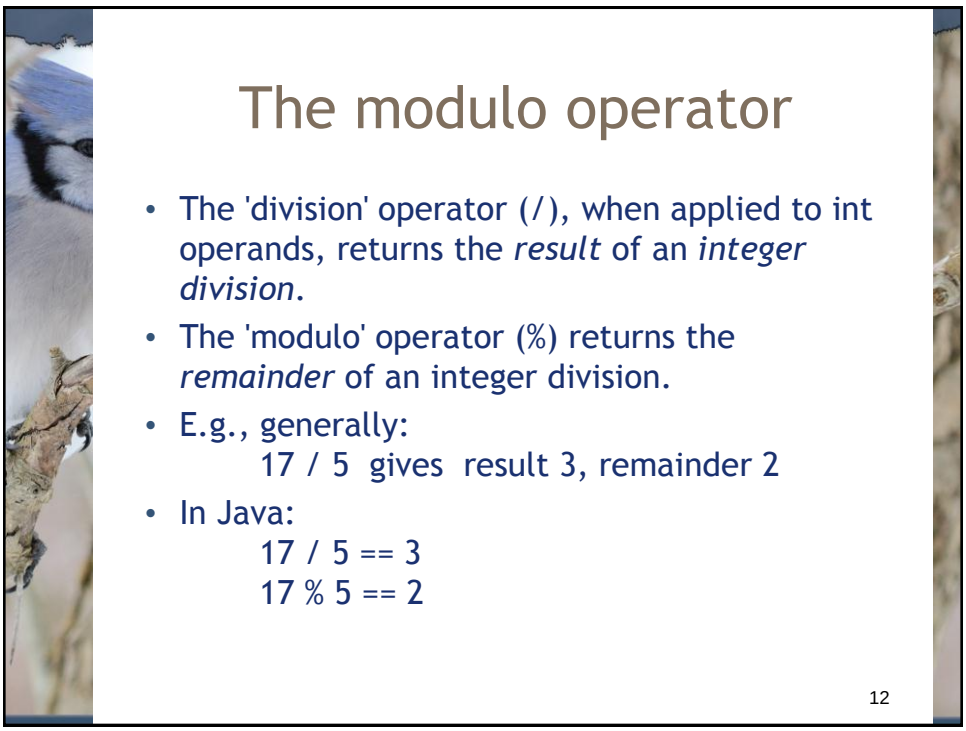
10



increment method

```
public void increment()
{
     value = value + 1;
     if(value == limit) {
       // Keep the value within the limit.
        value = 0;
     }
}
```

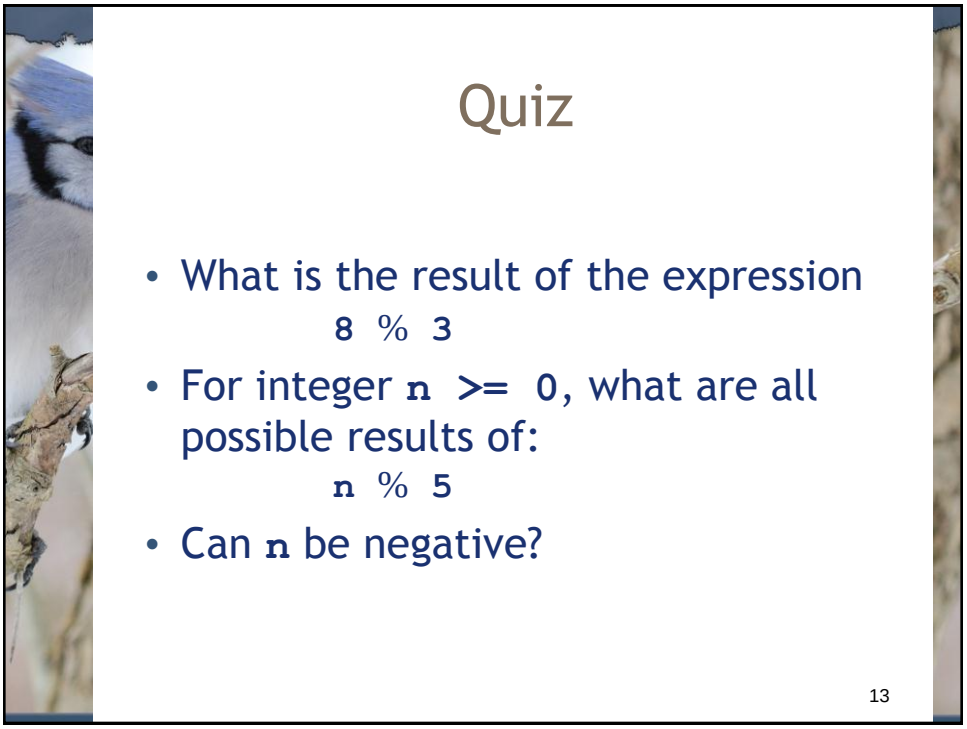
11

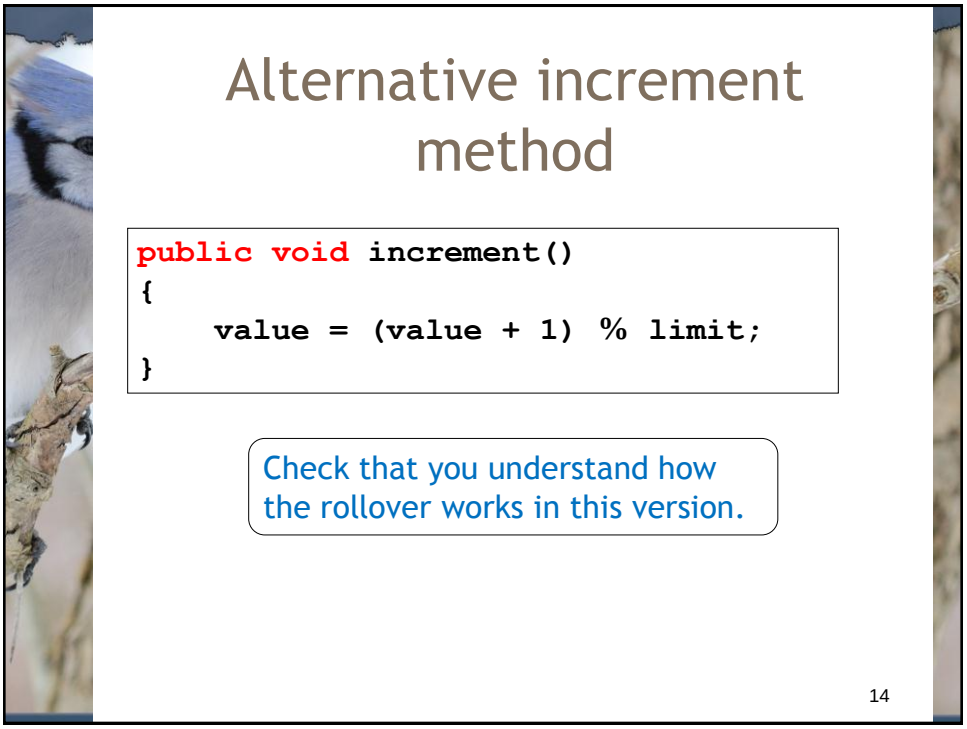


The modulo operator

- The 'division' operator (/), when applied to int operands, returns the *result* of an *integer division*.
- The 'modulo' operator (%) returns the *remainder* of an integer division.
- E.g., generally:
  17 / 5 gives result 3, remainder 2
- In Java:
  17 / 5 == 3
  17 % 5 == 2

12


Quiz
• What is the result of the expression
8 % 3
• For integer n >= 0, what are all possible results of:
n % 5
• Can n be negative?
13


Alternative increment method
public void increment()
{
value = (value + 1) % limit;
}
Check that you understand how the rollover works in this version.
14

Implementation - ClockDisplay

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

    Constructor and
    methods omitted.
}
```

15

Primitive types

Data can be classified under many different types. *Primitive type* is the one predefined by the Java language.

Type name	Description	Example literals	
Integer numbers			
`byte`	byte-sized integer (8 bit)	24	-2
`short`	short integer (16 bit)	137	-119
`int`	integer (32 bit)	5409	-2003
`long`	long integer (64 bit)	423266353L	55L
Real numbers			
`float`	single-precision floating point	43.889F	
`double`	double-precision floating point	45.63	2.4e5
Other types			
`char`	a single character (16 bit)	'm'	'?'
`boolean`	a boolean value (true or false)	true	false

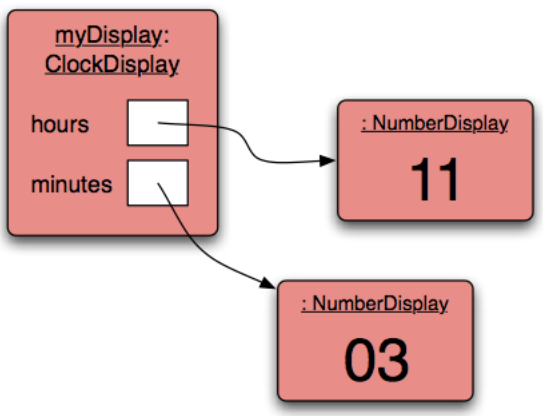
16

Classes as types

- In addition to primitive types, every class is a unique data type; e.g. **`String`**, **`TicketMachine`**, **`NumberDisplay`**.
- Data types, therefore, can be composites and not simply values.

17

Object diagram



18

Objects creating objects (1)

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

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

19

Objects creating objects (2)

in class ClockDisplay:

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

actual parameter

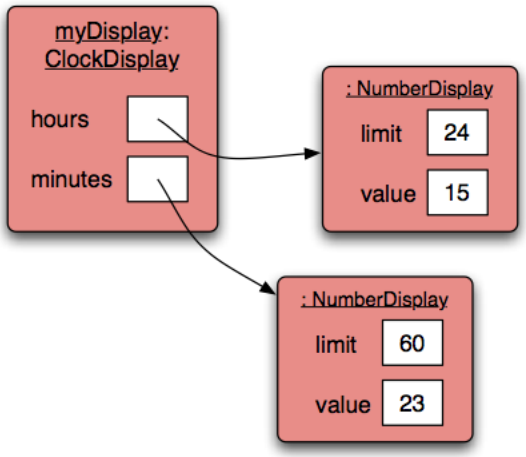
in class NumberDisplay:

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

formal parameter

20

ClockDisplay object diagram



21

Quiz: What is the output?

- ```
  int a;
  int b;
  a = 32;
  b = a;
  a = a + 1;
  System.out.println(b);
  ```

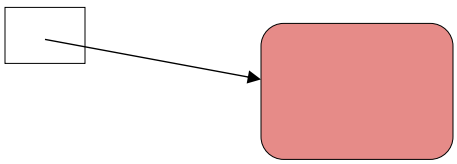
- ```
  Student a;
  Student b;
  a = new Student("Mohammed","Moh1440");
  b = a;
  a.changeName("Ahmed");
  System.out.println(b.getName());
  ```

22

Primitive types vs. object types

```
SomeObject obj;
```

object type



```
int i;
```

```
32
```

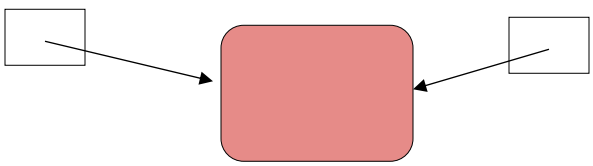
primitive type

23

Primitive types vs. object types

```
ObjectType a;
```

```
ObjectType b;
```



```
b = a;
```

```
int a;
```

```
32
```

```
int b;
```

```
32
```

24