Object interaction

Creating cooperating objects
A digital clock

11:03
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.
Modularizing the clock display

One four-digit display?

Or two two-digit displays?
Implementation - NumberDisplay

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

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

    // Constructor and methods omitted.
}

Object diagram
Class diagram

[Diagram showing a class diagram with 'ClockDisplay' and 'NumberDisplay' classes and a relationship between them]
Primitive types vs. object types

SomeObject obj;

int i;

object type

primitive type
Quiz: What is the output?

- ```
   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());
```
Primitive types vs. object types

```java
ObjectType a;
int a;

ObjectType b;
int b;

b = a;
```

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling
public NumberDisplay(int rollOverLimit)
{
    limit = rollOverLimit;
    value = 0;
}

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

at least two other ways to write increment()
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
Quiz

• What is the result of the expression
  \[ 8 \% 3 \]

• For integer \( n \geq 0 \), what are all possible results of:
  \[ n \% 5 \]

• Can \( n \) be negative?
public String getDisplayValue()
{
    if(value < 10) {
        return "0" + value;
    }
    else {
        return "" + value;
    }
}
Concepts

- abstraction
- modularization
- classes define types
- class diagram

- object diagram
- object references
- object types
- primitive types
Objects creating objects

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

    public ClockDisplay() {
        hours = new NumberDisplay(24);
        minutes = new NumberDisplay(60);
        ...
    }
}
```
Objects creating objects

in class ClockDisplay:

hours = new NumberDisplay(24);

actual parameter

in class NumberDisplay:

public NumberDisplay(int rollOverLimit);

formal parameter
ClockDisplay object diagram
public void timeTick() {
    minutes.increment();
    if (minutes.getValue() == 0) {
        // it just rolled over!
        hours.increment();
    }
    updateDisplay();
}
External method call

- external method calls

  minutes.increment();

  object.methodName(paramater-list)
Internal method call

- internal method calls
  
  `updateDisplay();`

- No variable name is required.

- `this`
  - could be used as a reference to the invoking object, but not used for method calls.
Internal method

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

- **Model:**
  - internal String field represents display

- **View:**
  - method invocation/print to terminal
    ```java
    System.out.println( getTime() );
    ```
  - GUI (Graphical User Interface) display

- **Controller:**
  - `timeTick()` invoked by code alone
  - `timeTick()` invoked by button/clock
Method calls

- NB: A method call on another object of the same type would be an external call.
- ‘Internal’ means ‘this object’.
- ‘External’ means ‘any other object’, regardless of its type.
- Method call = Method invocation (use these terms interchangeably)
null

- **null** is a special value in Java
- Object fields are initialized to **null** by default.
- You can test for and assign **null**:

```java
private NumberDisplay hours;

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

hours = null;
```
The debugger

• Useful for gaining insights into program behavior ...
• ... whether or not there is a program error.
• Set breakpoints.
• Examine variables.
• Step through code.
The debugger
Concept summary

- object creation
- overloading
- internal/external method calls
- debugger