Panel 1

Last time:

Classes: - blueprints to define new types
- fields (store data) + methods (to do stuff)
- constructor (class name, no return type)

Objects: - instantiated from classes via "new"
- they exist in Memory for real
- calls constructor

1

Panel 2

Data Encapsulation: protect integrity of a class

* private: only objects of same class can access
* public: everyone can access
* protected: } everyone in same package or
none (friendly): } directory has access

Only applies to fields / methods, not local vars

Rule: fields should be private
methods could be public
be as restrictive as possible

2

Panel 3

Class Temperature

(fields) has: value and unit (100 °C or 100 K)

do: convert to C, K, F

add / subtract / display

ensure integrity of Temp.

```
class Temperature
{
    double value
    char unit
    Temperature(double value, char unit)
  TemperatureToF()
        add(
}
```

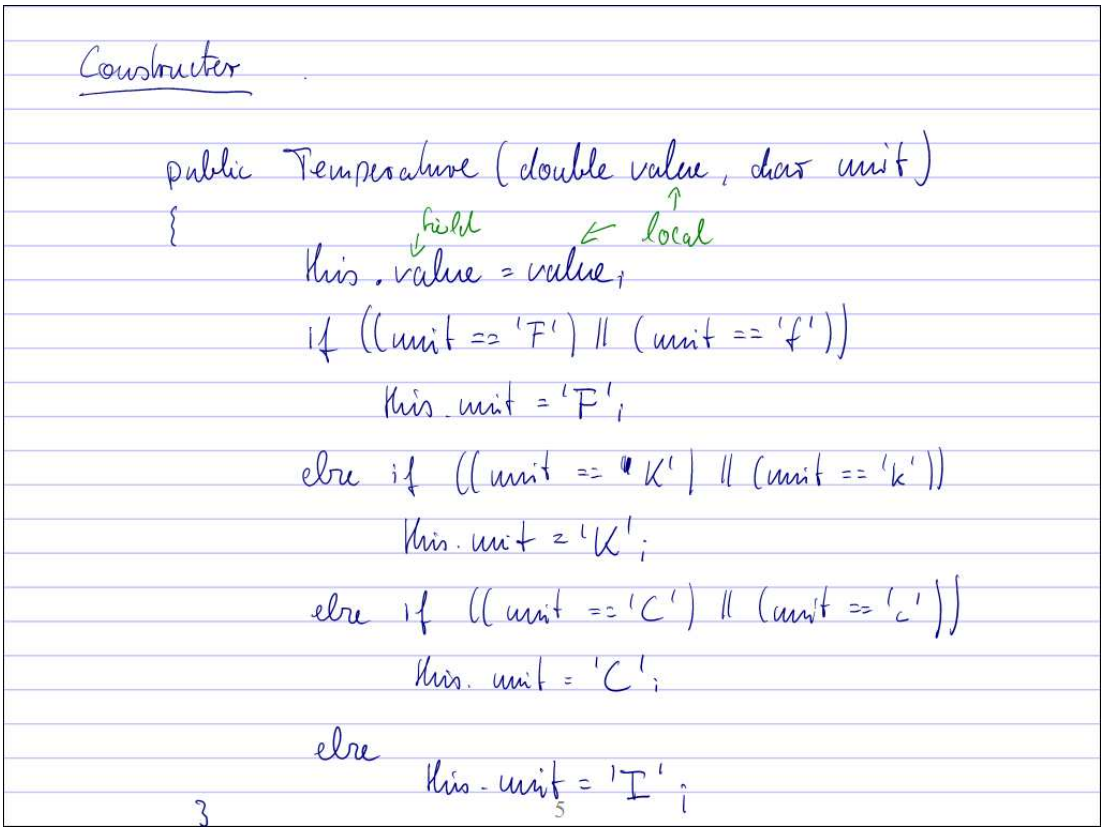
3

Panel 4

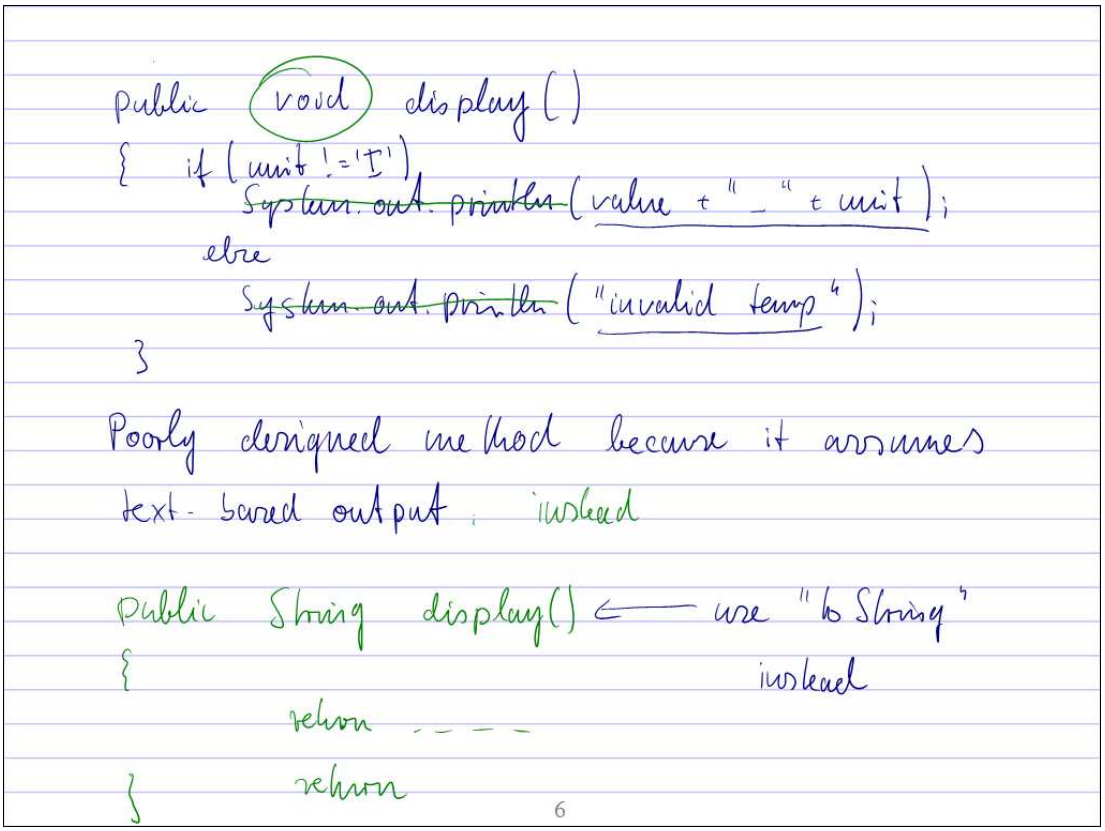
```
public class Temperature
{
  private double value = 0.0;
  private double unit = 'C';

⇒ public Temperature(double value, double unit)

  public TemperatureToF()   (etc)

  public Temperature add(Temperature t)   (etc)

}
```

4

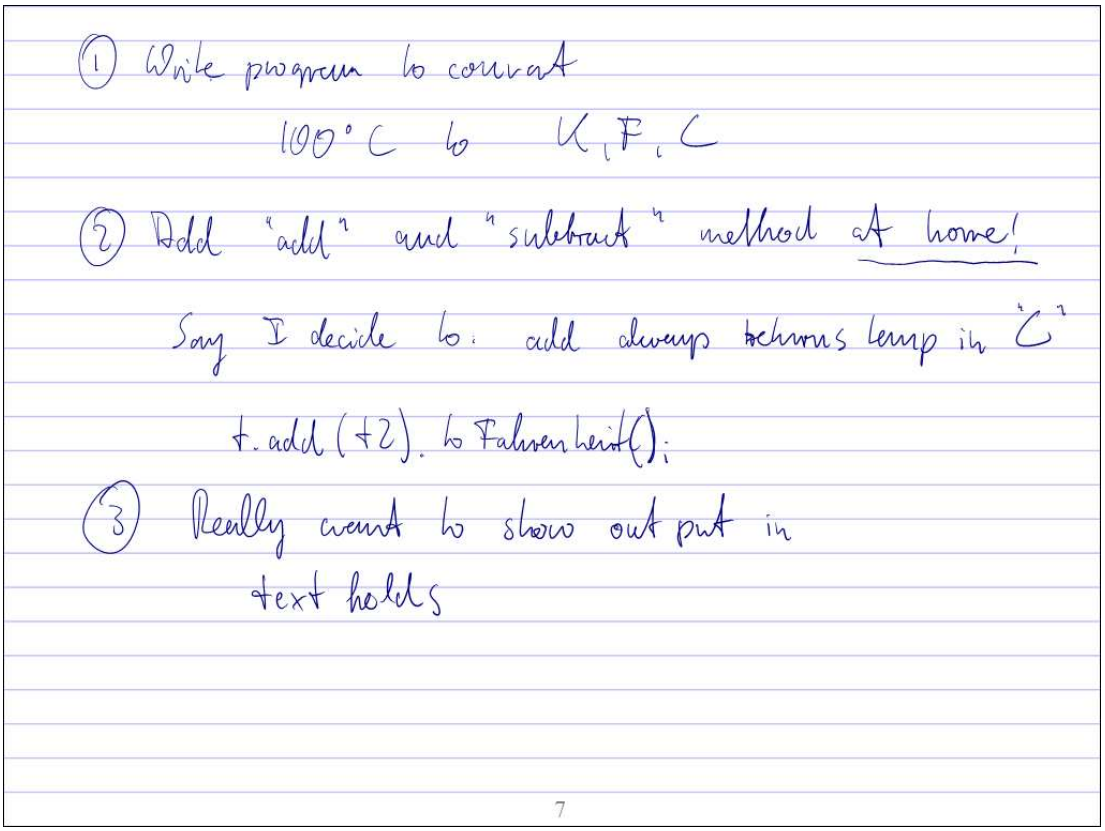
Panel 5



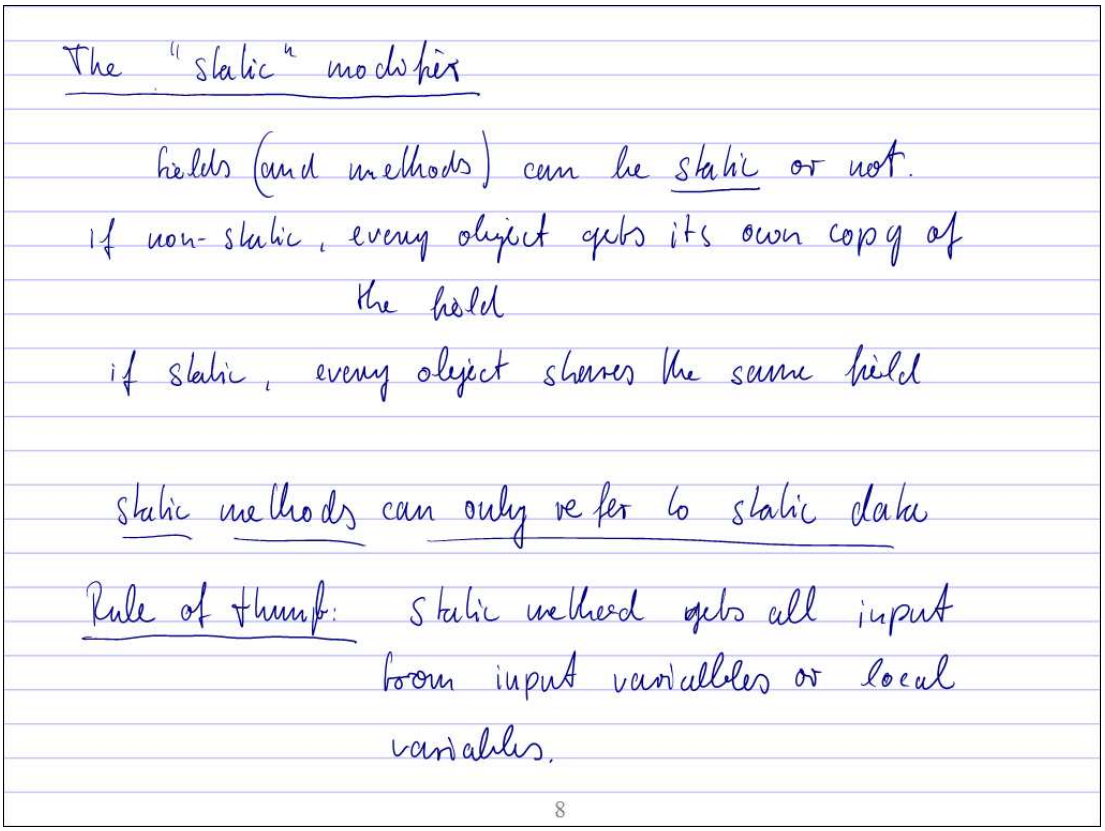
Panel 6



Panel 7



Panel 8



Panel 9

```
public class Temperature
{   double value;

    public Temperature toF()
    {
       // refers to fields
    }

    public static double convertToFahrenheit(double value)
    {
       // convert input value, not field.
    }
```

9

Panel 10

Static methods are often "computational".
They can be used without "new" but by giving class Name.

Ex: double x = Math.sin(Math.PI);

(annotations: Math — class; sin — method (static public); Math — class; PI — field public static ?)

Commonly used static methods:

Math.sin(#), Math.cos(#), ...

JOptionPane.showMessageDialog(null, "...");

JOptionPane.showInputDialog("...");

(annotation: JOptionPane — class)

10

Panel 11

```
import javax.swing.JOptionPane;

public class TempTest
{
   public static void main(String args[])
   {
      JOptionPane.showInputDialog("Enter temperature");

      Temperature t = new Temperature(100.0, 'C');

      String s = t.toString() + "\n";
      s += "    = " + t.toFahrenheit().toString() + "\n";
      s += "    = " + t.toKelvin().toString() + "\n";
      s += "    = " + t.toCelsius().toString() + "\n";
      JOptionPane.showMessageDialog(null, s);
   }
}
```

11