Recitation 3
Class and Objects

Spring 2012
Scope

- One method **cannot** see variables in another;
- Variables created inside a block: `{ ... }` exist from point of definition to end of block;
- Variable declaration and assignment can be separated to control scope:

```java
public static void main (String[] args){
    int repeat;  // declaration
    do{
        String input = JOptionPane.showInputDialog("1-cont, 0-stop");
        int repeat = Integer.parseInt(input); // assignment
    } while (repeat == 1);
}
```

How to fix it?
public class Capacitor {
    private double capacitance;

    public Capacitor(double c) {
        capacitance = c;
    }

    public double getCap() {
        return capacitance;
    }

    public void setCap(double c) {
        capacitance = c;
    }
}
Class Naming Conventions

public class Capacitor {
    private double capacitance;

    public Capacitor(double c){
        capacitance = c;
    }

    public double getCap (){ return capacitance; }

    public void setCap (double c){
        capacitance = c;
    }
}

- Class name is capitalized
- Variable names should start with a lowercase
- Constructor name matches class name (mandatory)
- Getters and Setters: getVariable setVariable
- Method names should start with lower case (except constructor)
Creating Objects

A class is a template to create objects.
Terminology: An object is an instance of a class.

// declare capacitor
Capacitor cp;

// call constructor
cp = new Capacitor(0.001);

Capacitor is the type of the variable cp (like int n, double x, ...)
The new keyword allocates memory for the object and calls its constructor

// declare and instantiate in one line
Capacitor cp = new Capacitor(0.001);
Using Objects

Use **public** methods to access and modify **private** data members

```java
// create a capacitor
Capacitor cp = new Capacitor(0.001);

// change its capacitance
cp.setCap(0.05);

// print the capacitance
System.out.println(cp.getCap());
```

The *dot* operator calls a method of a class on a particular instance of that class.

The `getCap()` method returns a **double**

→ `cp.getCap()` is "seen" as a **double**
Objects Exercise

In the `main` method of a `test` class:

- Create a first capacitor of capacitance 0.05
- Create a second capacitor of same capacitance
- Double the capacitance of the first capacitor
- Set the capacitance of the second capacitor to be twice the capacitance of the first one.
- Print out both capacitances.
Objects and Object References

Capacitor cp;

\[ cp = \]

Capacitor cp = new Capacitor(0.001);

Object Reference

Object
Objects and Object References

Primitive types are NOT objects. A variable of primitive type holds its data:

```java
int n = 5;
double x = 3.2;
```

A variable of any other type holds a reference to an object:

```java
Capacitor cp = new Capacitor(0.001);
```

```
0x4646
Capacitor
capacitance = 0.001
```
Class Design Exercise

We will model birds sitting on a branch.

- Each bird has its own weight.
- A branch can hold more than one bird but will break if a certain weight is exceeded.

Classes?  Data members?  Methods?
**Keyword: this**

**this:** refers to the current instance (or current object)

```java
public class Tank {
    ...
    public boolean isSameVolume(Tank t) {
        if (this.equals(t)) // Use .equals, not ==
            return true; // when comparing objects
        else
            return (getVolume() == t.getVolume());
    }
}
```

```java
public class TankTest {
    public static void main(String[] args) {
        Tank t0 = new Tank(1, 2, 3);
        Tank t1 = new Tank(1, 1, 3);
        System.out.println(t0.isSameVolume(t1)); // Output: false
    }
}
```

Output: false
null is the reserved constant used in Java to represent a void reference.

Capacitor \( cp = \) new Capacitor(0.001);

\[ cp = \text{null} \]

\[ cp = \text{null} \]
Homework 3: Buy a used car

Score = \( \frac{26,000 - \text{Price}}{3000} - 0.2 \times (\text{Years old}) + 0.2 \times (\text{MPG} - 25) + \text{Driver rating} \)

Classes you’ll need:
- A `UsedCarLot` class
- A `Car` class
- A `DriverRating` class
- A test class with `main()`

What you’ll need to do:
- Print driver rating for each car
- Compute score for each car
- Update scores after additional reviews
- Print cars less than 1 year old
- Print % of bad reviews for each car

<table>
<thead>
<tr>
<th></th>
<th>Toyota</th>
<th>Honda</th>
<th>Chevrolet</th>
<th>BMW</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Price</strong></td>
<td>18,000</td>
<td>20,000</td>
<td>17,000</td>
<td>26,000</td>
</tr>
<tr>
<td><strong>Years old</strong></td>
<td>0.5</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td><strong>MPG</strong></td>
<td>26</td>
<td>25</td>
<td>27</td>
<td>23</td>
</tr>
</tbody>
</table>

**Driver rating** = \( \frac{(\text{Good} + (0.5 \times \text{OK}) - \text{Bad})}{\text{Total}} \)

<table>
<thead>
<tr>
<th>Driver rating</th>
<th>Toyota</th>
<th>Honda</th>
<th>Chevrolet</th>
<th>BMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bad</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>OK</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Good</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

Where to put which method?