1.00/1.001
Introduction to Computers and Engineering Problem Solving

Recitation 3
Class and Objects

Spring 2012

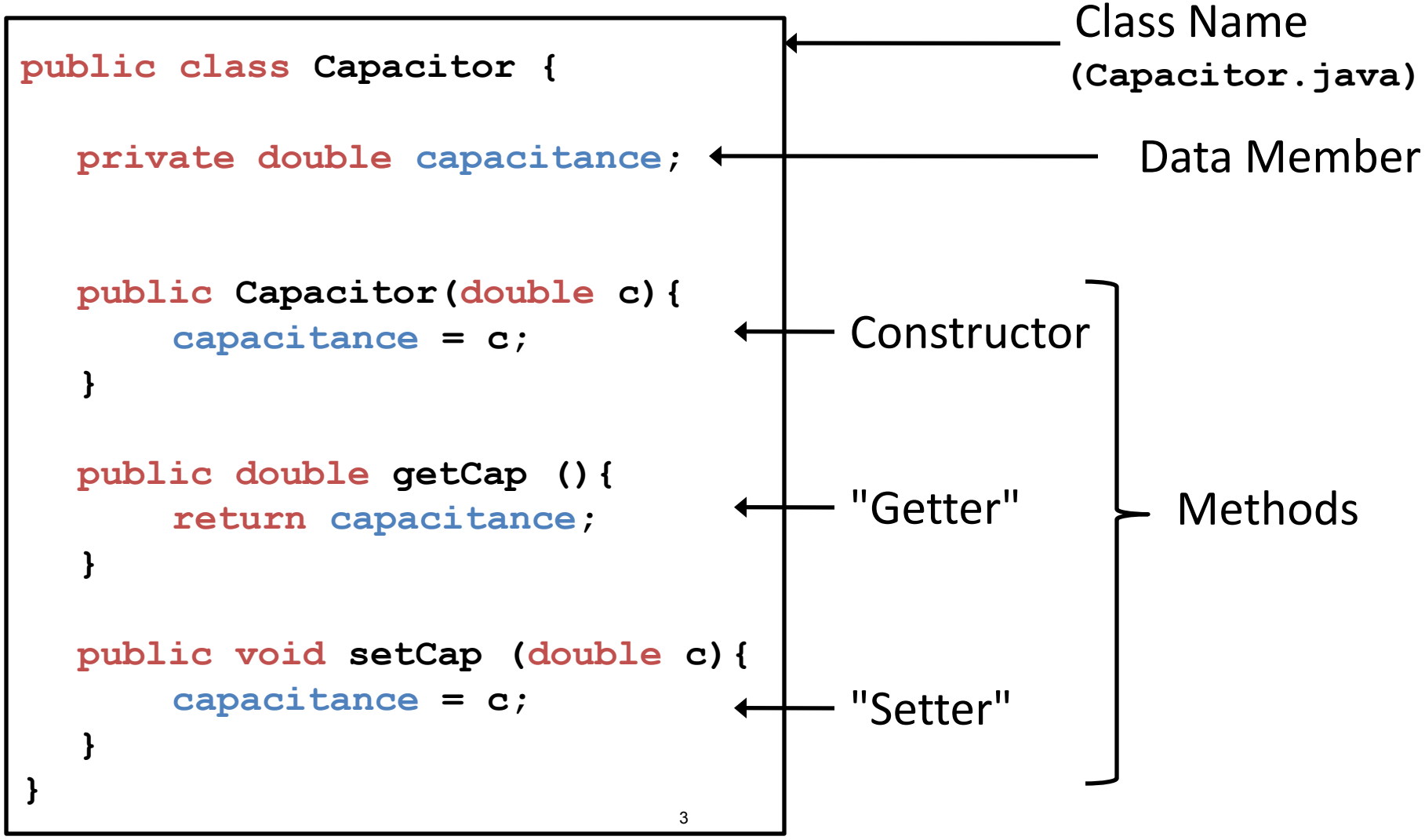
Scope

- 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:

```
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?

Anatomy of a Class



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;
```

Capacitor is the type of the variable **cp**
(like **int n, double x, ...**)

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

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

```
// 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:

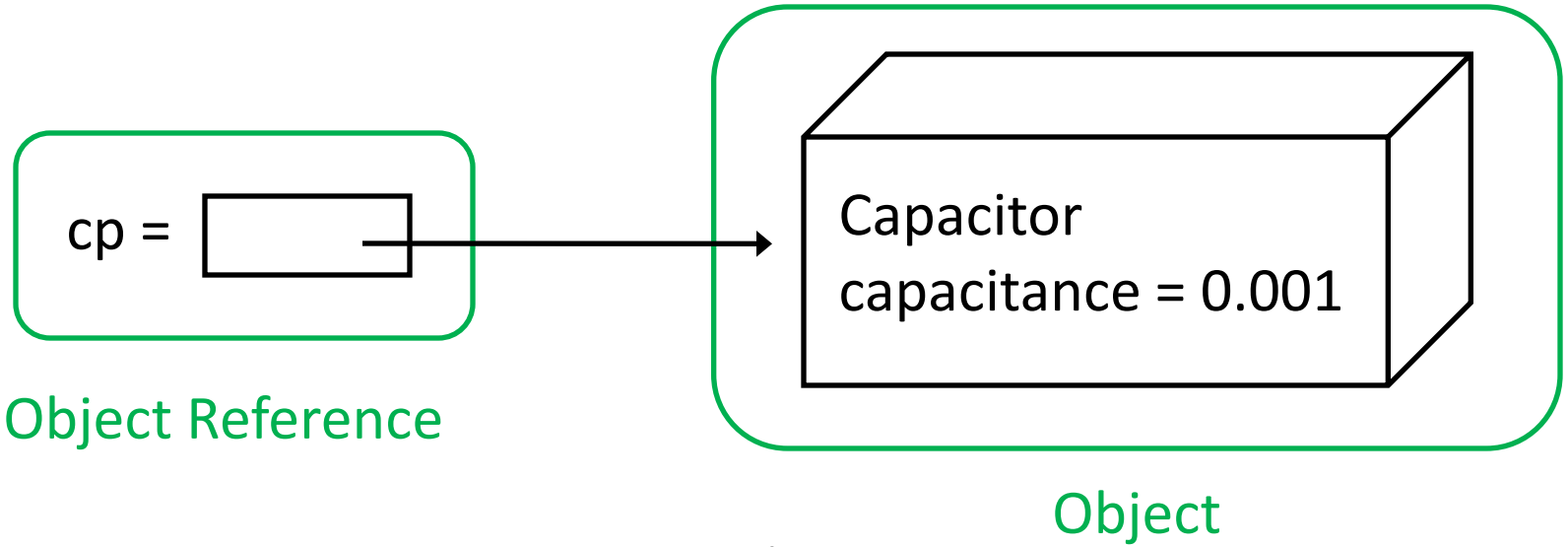
- Create a first capacitor of capacitance 0.05
- Create a second capacitor of same capacitance
- Double the capacitance of the first capacitor
- Set the capacitance of the second capacitor to be twice the capacitance of the first one.
- Print out both capacitances.

Objects and Object References

```
Capacitor cp;
```

cp =

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



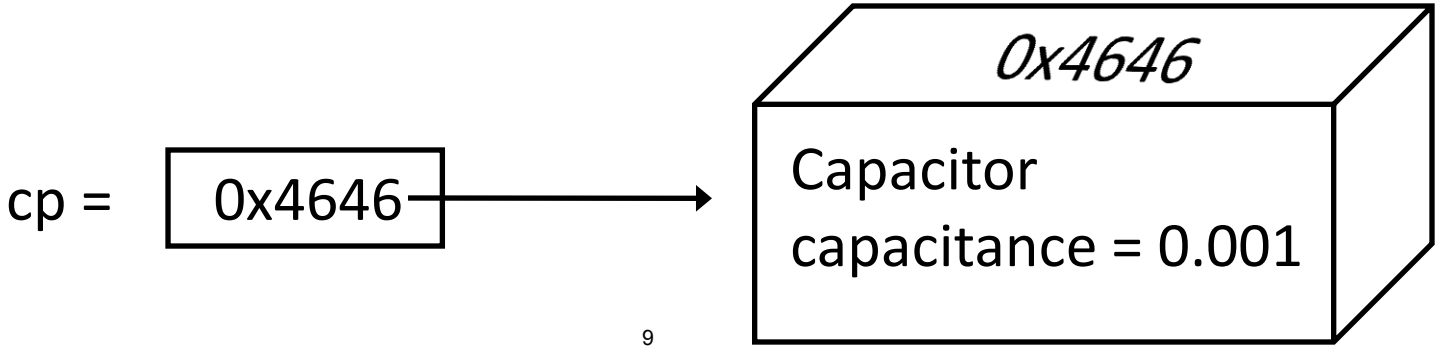
Objects and Object References

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

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

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

```
Capacitor cp   =  new Capacitor(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)**

```
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());
        }
}
```

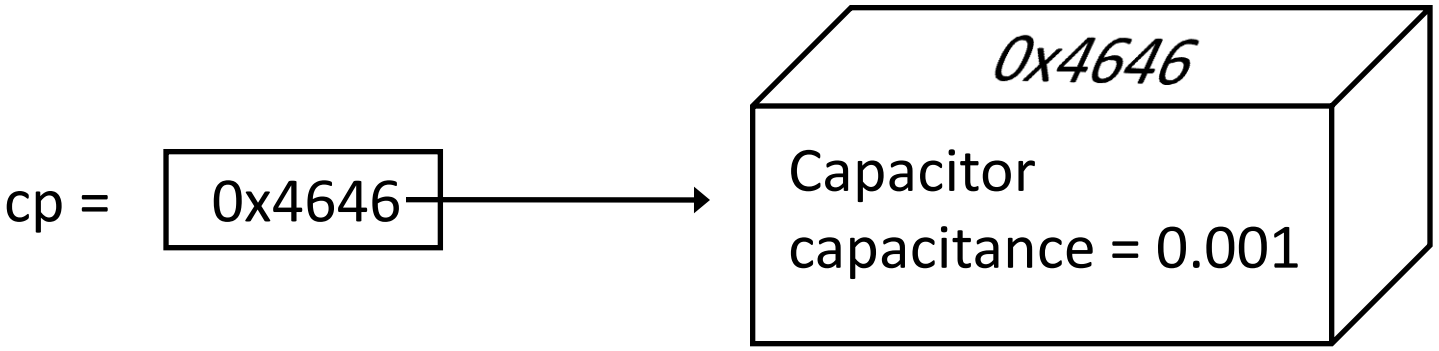
```
public class TankTest {
        public static void main(String[] args) {
                Tank t0 = new Tank(1, 2, 3);
                Tank t1 = new Tank(1, 1, 3);
                System.out.println(t0.isSameVolume(t1));
        }
}
```

Output: false

Keyword : `null`

`null` is the reserved constant used in Java to represent a void reference.

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




```
cp = null;
```

Homework 3: Buy a used car

Score = (26,000 – Price)/3000 – 0.2*(Years old) + 0.2*(MPG-25) + Driver rating

Classes you'll need:

- A `UsedCarLot` class
- A `Car` class
- A `DriverRating` class
- A test class with `main()`

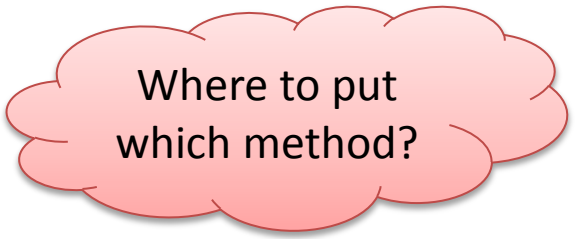
	Toyota	Honda	Chevrolet	BMW
Price	18,000	20,000	17,000	26,000
Years old	0.5	1	1	4
MPG	26	25	27	23

Driver rating = (Good + (0.5*OK) - Bad)/Total

Driver rating	Toyota	Honda	Chevrolet	BMW
Bad	3	1	1	0
OK	2	2	4	1
Good	1	6	5	3

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



1.00 / 1.001 / 1.002 Introduction to Computers and Engineering Problem Solving

Spring 2012