Class members vs. Instance members

Object-Oriented Programming
Outline

- Class methods vs. instance methods
- Class variables vs. instance variables

Readings:
- HFJ: Ch. 10.
- GT: Ch. 10.
Class methods

- **Examples:**
  ```java
double x = Math.round(42.2);
int y = Math.abs(-10);
  ```

- Methods in the Math class don't use any instance variable values. So they don't need to know about a specific Math object. All we need is the Math class.

- Math functions were written as **class** methods, or **static** methods.

- A class method (static method) is one that runs *without any instance of the class.*
Instance methods vs. class methods

**Instance (regular) methods**

class Cow {
    String name;
    public String greeting() {
        return ("Hi, I am " + name);
    }
}

- instance variable `name` affects the behavior of `greeting()`
- **MUST** be called using a reference variable
  `s = cow1.greeting();`

**Class (static) methods**

class Math {
    public static int abs(int a) {
        if (a > 0) return a;
        return -a;
    }
    ...
}

- `abs()` has absolutely nothing to do with any Math instance variables
- **CAN** be called using the class name:
  `int a = Math.abs(-10);`
This won't compile

```java
public class Duck {
    private int size;

    public static void main(String[] args) {
        System.out.println("Size of duck is " + size);
    }

    public void setSize(int s) {
        if (s > 0) size = s;
    }

    public int getSize() {
        return size;
    }
}
```

Which duck? Whose size?

If there's a duck or ten ducks on the heap somewhere, the static method doesn't know about any of them.

I've no idea which duck you are talking about!
Class method can't use instance variables or instance methods

- Class methods can be called using class name
  - no instance, no `this` reference, no owner object

```java
public class Duck {

    private int size;

    public static void main( String[] args) {
        Duck d = new Duck();
        setSize(10);
        System.out.println("Size of duck is " + size);
    }

    public void setSize (int s) {...}

    ... 

    private int size;
}
```
Class method can't use instance variables or instance methods

- Class methods can be called using class name
  → no instance, no this reference, no owner object

```java
public class Duck {
    private int size;

    public static void main(String[] args) {
        Duck d = new Duck();
        d.setSize(10);
        System.out.println("Size of duck is " + d.size);
    }

    public void setSize(int s) {
        ...}
```

So you must specify: 
"d is the duct whose size I want"
Actually, it’s better to put the program in a separate class.
- Class Duck should define Duck objects only.
- Different programs can use the same Duck class.

```java
public class Duck {
    private int size;
    public void setSize(int s) {...}
    ...
}

public class DuckProgram {
    public static void main(String[] args) {
        Duck d = new Duck();
        setSize(10);
        System.out.println("Size of duck is " + size);
    }
}
```
Class variables

- A class variable (or static variable) belong to the class, not any object.
- One copy shared among all class instances.

```java
public class Duck {
    private int size;
    public static int count = 0;

    public Duck() {
        count++;
    }

    ...
}
```

Each duck has its own size. But all ducks share the same count.
public class Duck {
    private int size;
    public static int count = 0;

    public Duck() {
        count++;
    }
    ...
}

public class DuckTestDrive {
    public static void main(String[] args) {
        System.out.println(Duck.count);
        Duck d = new Duck();
        System.out.println(d.count);
    }
}

% java DuckTestDrive
0
1

before any ducks are made

after the first duck is created
Class variables vs. Instance variables

Class/static variables
- belong to a class
- one copy shared among all instances of the class
- initialized before any objects of the class

Instance variables
- belong to an instance
- each instance has its own copy
- initialized when the owner object is created

```java
class Duck {
    private int size = 0;
    public static int count = 0;

    public Duck() {
        count++; size++;
    }
}
```
Design pattern : Singleton

- Singleton: Ensure a class has only ONE instance, and provide a global point of access to it.

<table>
<thead>
<tr>
<th>Singleton</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>static</strong> instance: Singleton</td>
</tr>
<tr>
<td>- Singleton()</td>
</tr>
<tr>
<td>+ getInstance(): Singleton</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

- Uses
  - In place of global variables
  - In system resource management
    - Avoid conflicting accesses from concurrent processes

```java
if (instance == null) {
    instance = new Singleton();
}
return instance;
```