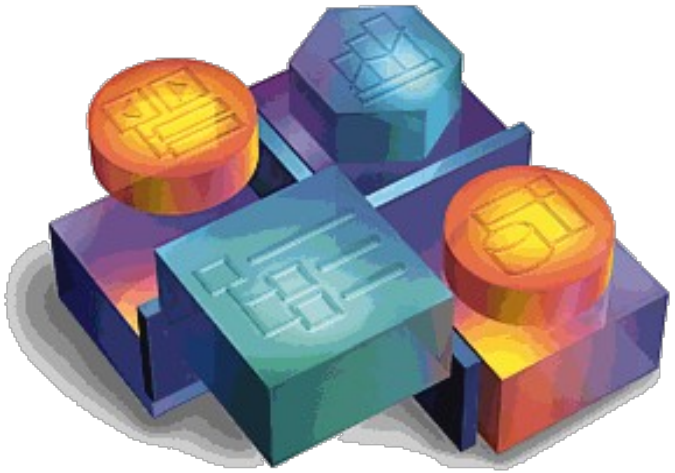
Object-Oriented Programming Design



1. Introduction to Object-Oriented Concepts
2. How to think in Terms of Objects

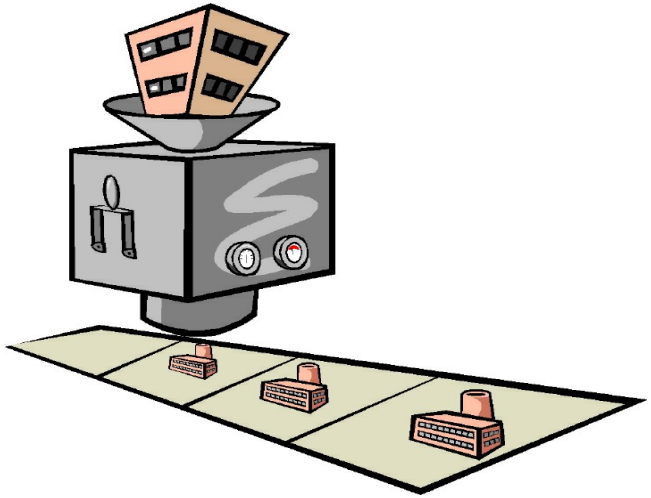
3. Advanced Object-Oriented Concepts

4. The anatomy of a Class
5. Class Design guidelines
6. Designing with objects
7. Mastering Inheritance and Composition
8. Frameworks and Reuse: Designing with interfaces and Abstract classes
9. Building objects
10. Creating Object Models with UML
11. Objects and Portable Data: XML
12. Persistent Objects: Serialization and Relational Databases
13. Objects and the Internet
14. Objects and Client/Server Aplications
15. Design Patterns

3. Advanced Object-Oriented Concepts

Constructors:

- ►Constructors are methods
- ►same name as the class
- ►no return type



3. Advanced Object-Oriented Concepts

Constructors:

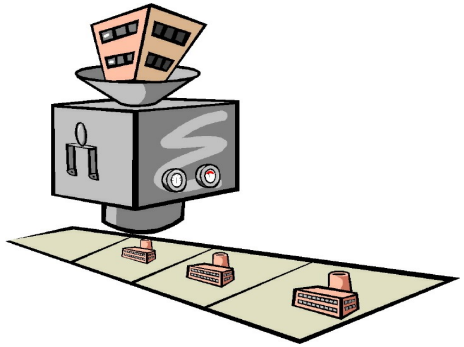
In C++

```
Class A
 {public:
      A(){
                  /* Code to construct the object*/
       }
};
```

In Java

```
Public Class A {
       public A(){
                  /* Code to construct the object*/
       }
}
```

3. Advanced Object-Oriented Concepts



What's inside a constructor?

►Code included inside a **constructor** should set the newly created object to its initial, stable, **safe state.**

For example if you have counter object with an attribute called *count,* you need to set *count* to zero in the constructor:

Count=0;

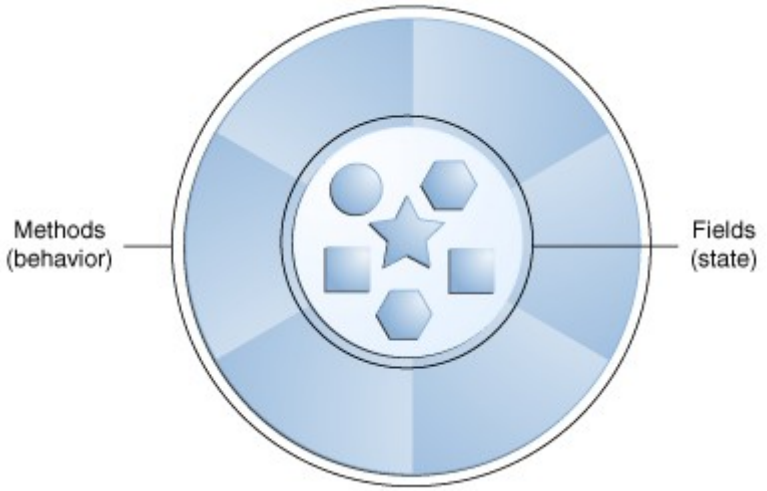
►Initializing attributes is a common function performed within a constructor.

An example in Java

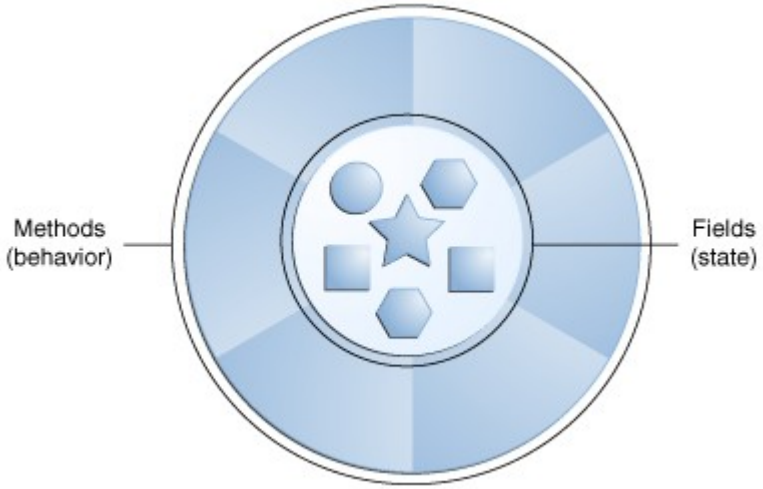
Class: First let's create Classes:

```java
//Create the main Class
public class Main {
    public static void main(String[] args) {

    }
}
```

```java
//Create a Vehicle 车 Class
public class Vehicle {
    String color;

    public Vehicle(){
        color=new String("红色");
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public void move(){
        System.out.println("My color is "+color+" and I'm moving...");
    }
}
```



Objects: let's create object



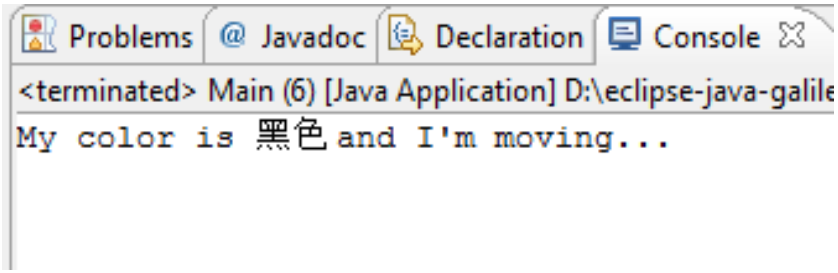

```
//Create object
public class Main {
    public static void main(String[] args) {
        Car myGeelyCar=new Car();
        myGeelyCar.setColor("黑色");
        myGeelyCar.move();
    }
}
```

Create an object myGeelyCar

Encapsulation

Send message

Run...



3. Advanced Object-Oriented Concepts

Name of class: A

Attributes
Int x

Methods
A()
show()

Class diagram

An example in C++

```
class A
{public:
    A()                          //定义构造函数,函数名与类名相同
    {   x=0;                     //利用构造函数对对象中的数据成员x赋初值
        cout<<"Constructor is Executed!"<<endl;
    }
    void show() { cout<<"x="<<x<<endl; }
private:
    int x;
};
```

3. Advanced Object-Oriented Concepts

An example in C++

```
int main()
{   A a1;
    A1.show();
    A a2
    a2.show();
    return 0;
}
```

Name of class: A

Attributes
Int x

Methods
A()
show()

instantiate 2 objects
(Create 2 objects)

object:
a1
call method:
a1.show()

object:
a2
call method:
a2.show()

Constructor is executed!
X=0
Constructor is executed!
X=0

3. Advanced Object-Oriented Concepts

The default constructor:

- If the class provides no explicit constructor, a default constructor will be provided.

- Besides the creation of the object itself, the only action that a default constructor takes is to call the constructor of its superclass.

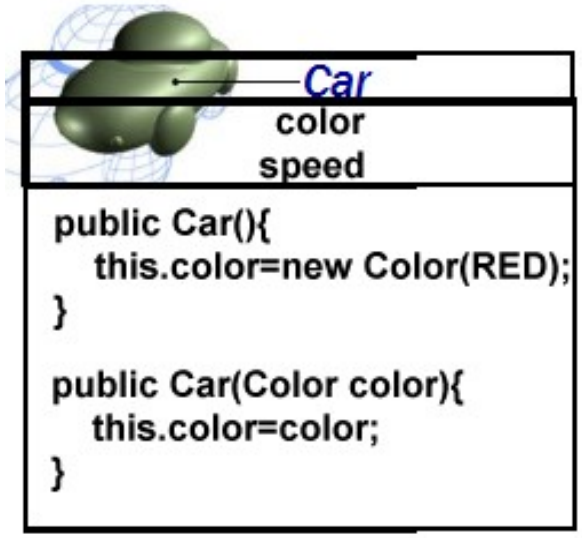
3. Advanced Object-Oriented Concepts

Using Multiple Constructors:

►In many cases, an object can be constructed in more than one way. To accomodate this situation, you need to provide more than one constructor

3. Advanced Object-Oriented Concepts

Using Multiple Constructors:



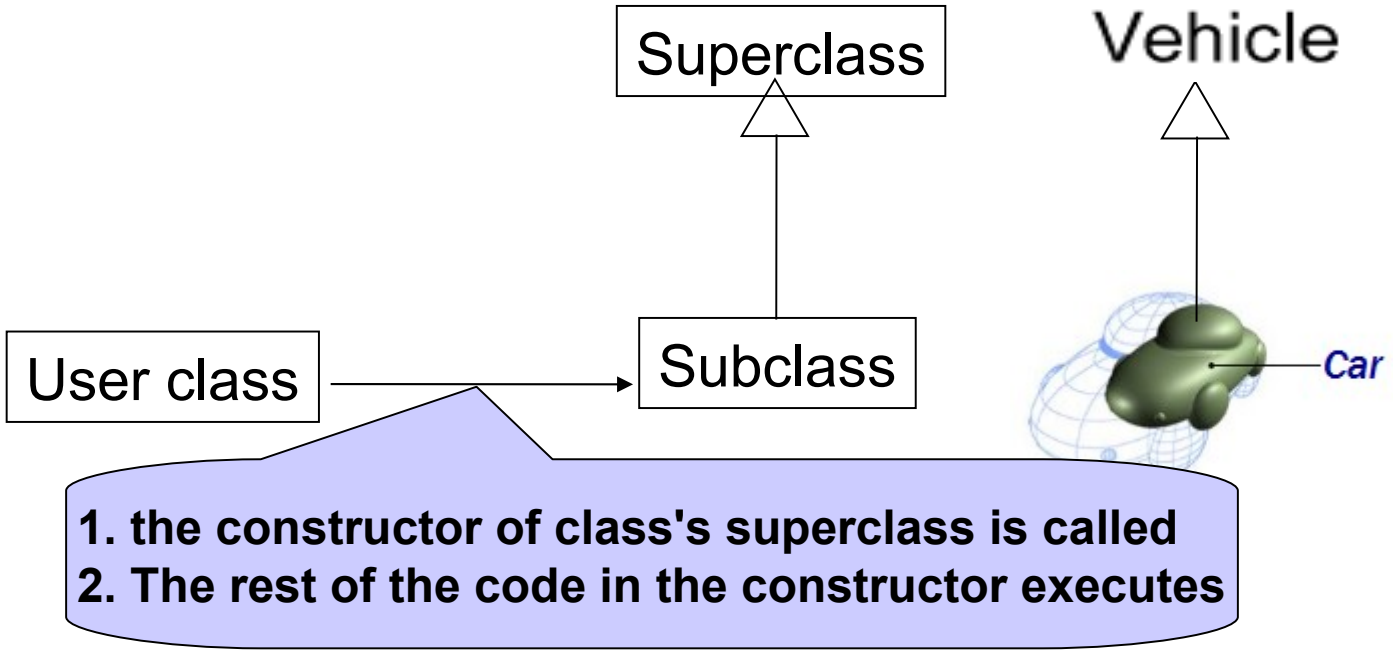
new Car();

new Car( );

new Color(BLUE);

3. Advanced Object-Oriented Concepts

How the Superclass is Constructed



3. Advanced Object-Oriented Concepts

The design of Constructors:

►When designing a class it is good practice to initialize all the attributes.

►**Constructors** are used to **ensure** that the application is in a **stable state.**

3. Advanced Object-Oriented Concepts

The design of Constructors:

- For example, initializing an attribute to zero, when it is instended for use as a denominator in a division operation, might lead to an **unstable** application.

  Take in consideration the fact that a division by zero is an illegal operation.

- Initializing to zero is not always the best policy.
- During the design, it is good practice to identify a **stable state** for all attributes and then initialize them to this stable state in the constructor.

3. Advanced Object-Oriented Concepts

Error handling: Throwing an Exception

►Most OO languages provide a feature called **exceptions**

►**Exceptions are unexpected events that occur within a system.**

►**Exceptions** provide a way to detect problems and them handle them.

In Java, C#, C++, exceptions are handled by the keywords **catch** and **throw**.

try/catch block:

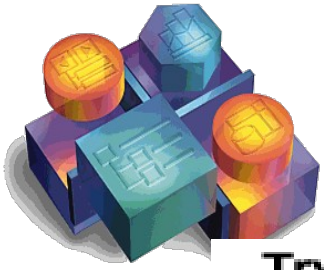
```
Try {
    // possible nasty code
} catch(Exception e){
    // code to handle an exception
}
```

3. Advanced Object-Oriented Concepts

Error handling: Throwing an Exception

Example in java:

```
Try {
    // possible nasty code
    count=0;
    count=5/count;
} catch(ArithmeticException e){
    // code to handle an exception
    count=1;
}
```



3. Advanced Object-Oriented Concepts

Error handling: Throwing an Exception

```
Try {
    // possible nasty code
    count=0;
    count=5/count;
} catch(ArithmeticException e){
    // code to handle an exception
    count=1;
}
```

count=5/1

5/0

3. Advanced Object-Oriented Concepts

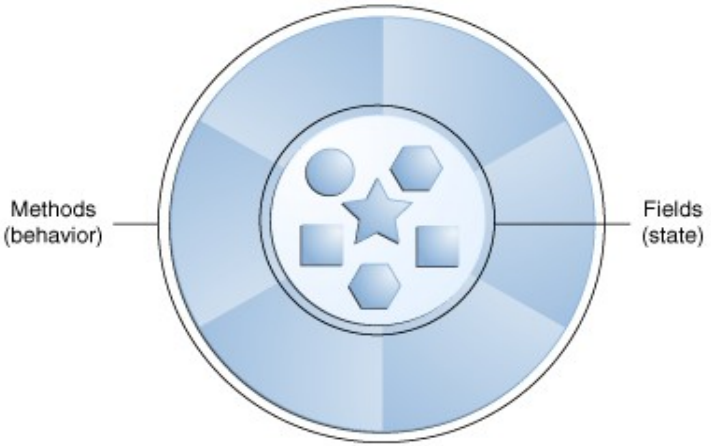
The concept of scope:

Methods represent the behaviors of of an object.

The **state** of the object is reprensented by attributes.

There are three types of attributes:

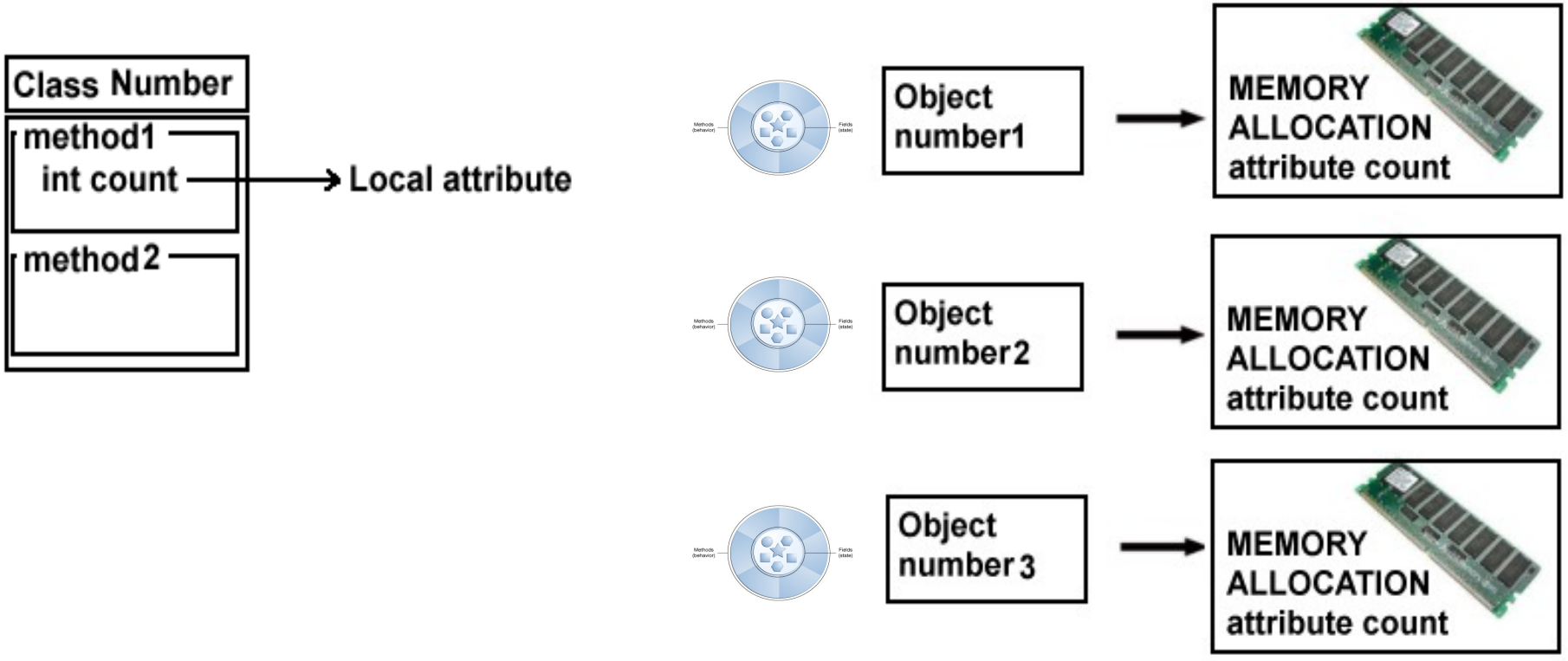
- **Local attributes**
- **Object attributes**
- **Class attributes**



3. Advanced Object-Oriented Concepts

Local attributes:

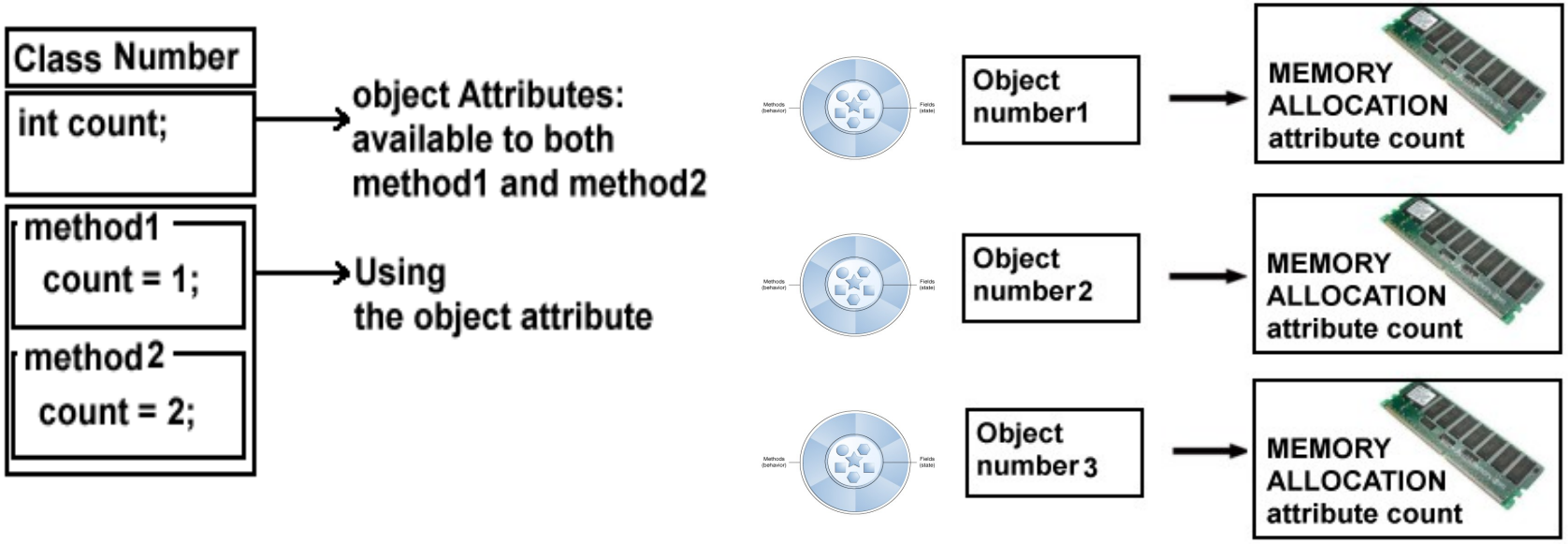
Local attributes are owned by a specific method.



3. Advanced Object-Oriented Concepts

Object attributes:

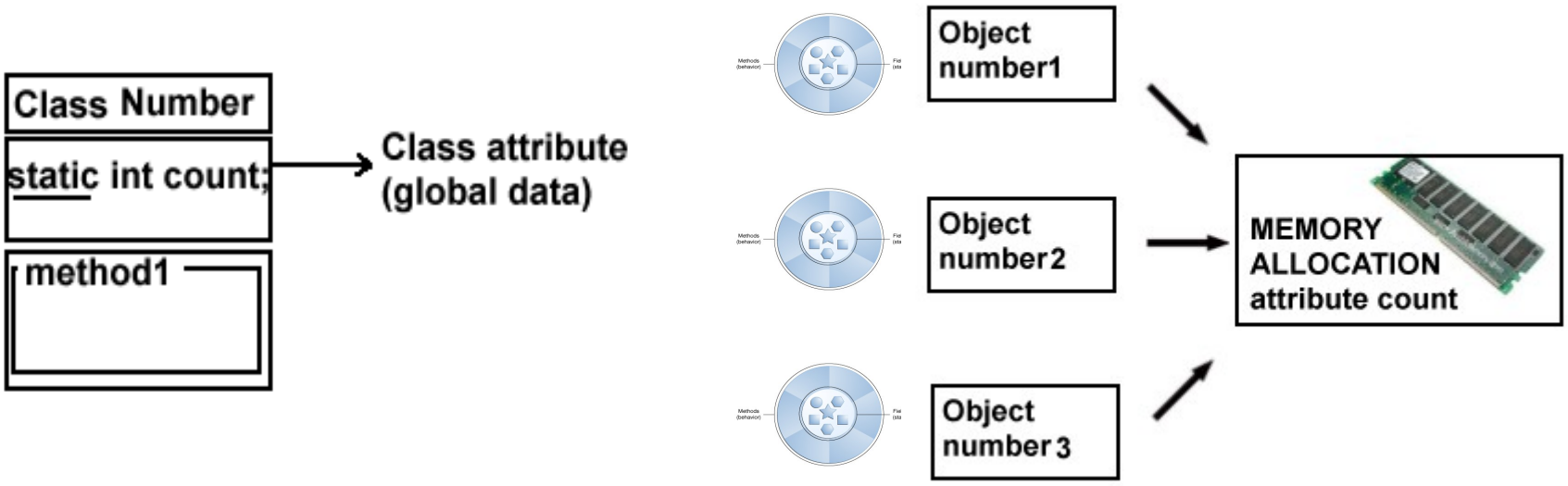
►There are many design situations in which an attribute must be shared by several methods within the same object.



3. Advanced Object-Oriented Concepts

Class attributes:

- ►It is possible for two or more objects to share attributes. In java, C#, and C++, you do this by making the attribute **static.**
- ►All objects of the class use the same memory location for **count.**



1. Introduction to Object-Oriented Concepts
2. How to think in Terms of Objects
3. Advanced Object-Oriented Concepts

4. The anatomy of a Class

5. Class Design guidelines
6. Designing with objects
7. Mastering Inheritance and Composition
8. Frameworks and Reuse: Designing with interfaces and Abstract classes
9. Building objects
10. Creating Object Models with UML
11. Objects and Portable Data: XML
12. Persistent Objects: Serialization and Relational Databases
13. Objects and the Internet
14. Objects and Client/Server Aplications
15. Design Patterns