

ESc101 : Fundamental of Computing

I Semester 2008-09

Lecture 24

Object Oriented programming

- Quiz (access control for a member of class)
- Static attribute/field of a class
- Static method of a class

Access control

- access control is performed on a per class basis and not per object basis.
- members of a class are always accessible from all code written in that class **regardless of which instance the code is being applied to.**

Consequence

Even if some attribute of an object is private, it **can be accessed and modified** during execution of a method of that class even though the method is being called on some another instance (object) of the class.

The **box.java** and **box_example.java** files available on the website highlight this very important point.

Revisit Point class

```
public class Point
{
    double x;
    double y;

    public Point(double x1, double y1)
    {
        x = x1;
        y = y1;
    }

    public double distance_from_origin()
    {
        double dist;
        dist = Math.sqrt(x*x + y*y);
        return dist;
    }
}
```

Each Point object will have its own copy of attributes x and y.

Since an object is an instance of its class, these attributes are also called

instance attributes or **instance variables**

Question ?

**Can there be a need of an attribute which is common to all objects of a class
?**

Question ?

**Can there be a need of an attribute which is common to all objects of a class
?**

Answer : yes

Example : Bank account class

Bank account class

```
public class bank_account
{ String name;
  long account_num;
  double balance;

bank_account(String s)
{ name = s;
  balance = 0;
  account_num = ??
}
// The methods for bank account :
// balance inquiry(),
// deposit(double amount),
// withdraw(double amount);
}
```


How to assign a unique number to each customer's account ?

How to assign a unique number to each customer's account ?

Solution :

1. Define an attribute **nextN** which stores the next account number available.
2. When an account is created, assign **nextN** to **account_num**;
3. Increment **nextN** by one;

How to assign a unique number to each customer's account ?

Solution :

1. Define an attribute **nextN** which stores the next account number available.
2. When an account is created, assign **nextN** to account_num;
3. Increment **nextN** by one;

How to ensure these features ?

Solution : declare nextN as a **static** attribute

Bank account class

```
public class bank_account
{ static long nextN=1000;

    String name;
    long account_num;
    double balance;

    bank_account(String s)
    { name = s;
      balance = 0;
      account_num = nextN;
      nextN=nextN+1;
    }
    // The methods for bank account :
    // balance inquiry(),
    // deposit(double amount),
    // withdraw(double amount);
}
```

Static attribute

1. Only one copy of a static attribute exists at any moment of time irrespective of the number of objects of the class created.
2. A static attribute is available even before an object is created.

Rest everything is same between static and instance attributes. For example, a static attribute can be accessed and manipulated by any objects of the class.

The reason behind choosing **static** word

Objects are created dynamically during the execution of a program. Hence the instance attributes are also called non-static attributes.

Whereas the static attributes stay throughout, so are called static attributes.

- static attributes are usually called class attribute
- instance attribute are usually called non-static attribute

What about static method ?

Static methods

- A static method is invoked on behalf of an entire class, not on a specific object.
- A static method might perform some general task (not specific to an object of the class), whereas, a non-static method is used for accessing, manipulating a specific object.

How to invoke a static method, say *method1* ?

Within the class you may invoke it in a method directly by calling *method1* and passing arguments if any.

Though you may invoke a static method as *refer.method1* where *refer* is a reference to an object of the same class (whose member is *method1*), but it is considered a bad programming practice.

Outside the class you may invoke it by *class_name.method1*, where *class_name* is the class in which *method1* is defined.

Example : Bank account class

What if we want to ensure that

- **nextN** should not be modifiable by any method outside the **bank_account** class ?
- **nextN** should be allowed to be read from a method outside the **bank_account** class.

Example : Bank account class

What if we want to ensure that

- **nextN** should not be modifiable by any method outside the **bank_account** class ?
- **nextN** should be allowed to be read from a method outside the **bank_account** class.

Solution :

- declare **nextN** as private
- and introduce a **static method** which returns **nextN**.

Example : Bank account class

```
public class bank_account
{ private static long nextN;

    public static long get_nextN() { return nextN; }

    String name;
    long account_num;
    double balance;

    bank_account(String s)
    {
        name = s;
        balance = 0;
        account_num = nextN;
        nextN=nextN+1;
    }
    // The methods for bank account :
}
```

Example : What will be output of the following program ?

```
import Geometry.Point;
class class1
{
    public Point Mid_point(Point P, Point Q)
    {
        double mid_x = (P.getX() + Q.getX())/2;
        double mid_y = (P.getY() - Q.getY())/2;
        return (new Point(mid_x,mid_y));
    }

    public static void main(String args[])
    {
        Point P = new Point(4,4);
        Point Q = new Point(4,0);
        Point mid = Mid_point(P,Q)
        System.out.println(mid.getX()+" , "+mid.getY());
    }
}
```

Compilation error

non-static method Mid_point(Geometry.Point,Geometry.Point)
can not be referenced from a static context

```
Point mid = Mid_point(P,Q);  
            ^
```

1 error

For more examples go through the files :

class1.java, class2.java on the course website.

Question : Can a static method access non-static attributes or non-static method ?

Question : Can a static method access non-static attributes or non-static method ?

Answer : NO

Reason :

Since a non-static attribute or method have to be accessed via an object reference.

Rules for static method

- A static method of a class can access only static methods or static attributes of its class

Summary : a member(attribute or methods) of a class

- can be static or non-static (default)
- can have its own access control : private, package(default), public

Summary : A class may consists of

- only methods also, and they should be declared static as a good programming practise so that these methods can be invoked from other static methods directly.
- only the attributes.
- both attributes as well as methods.

Complete picture of a class

class class_name

{

public static double pi=3.1426 ;
private static int count =0 ;

●
●
●

public static void method1()
{ }
private static int method2(int n)
{ }

Static attributes
and
Static methods

private double balance ;
public int id_num ;

●
●
●

private void methodk()
{ }
public int methodj(int n)
{ }

Non-static attributes
and
non-static methods

}