



# Object Oriented DBMS


- 
- Database that stores data elements as objects.

Uses object-oriented concepts.

- The term object oriented is abbreviated by OO or O-O

# Object


- The entity that contains both attributes as well as the actions associated with it
- The object has two components
  - ✓ State
  - ✓ behavior
- Attributes can be classified into simple and complex
- An object is described by following characteristics
  - Identifier: a system-wide unique id for an object
  - Name: an object may also have a unique name in DB (optional)
  - Lifetime: determines if the object is persistent or transient

- 
- A simple attribute can be an integer, string, real and so on. Which takes on specific values.
  - A complex attribute can contain collection and/or references.
  - A reference attribute represent a relationship between objects and contain a value or values.

# Example

Object attributes for branch instance

Attribute	Values
BranchNo	B003 → simple
Street	163 main st
City	Islamabad
Postcode	22010
SalesStaff	Ali khan; Atif → c/r khan Amjad khan
Manager	

- 
- BranchNo is example of simple attribute with value B003
  - The attribute Salesstaff is collection of staff objects.
  - Salestaff is example of reference attribute.
  - The reference attribute is similar to foreign key in the relational DBMS.

# Features of OO

## 1.Object Identity

- An OO database provides a unique identity to each independent object stored in the database.
- This unique identity is implemented via a unique system generated object identifier OID.
- The value of OID is not visible to the user but it is used internally by the system to identify each object uniquely.

- The main property required of an OID is that it be immutable that is the OID value of a particular object should not change.
- It is also desirable that each OID be used only once, that is even the object is removed from the database its OID should not be assigned to another object.
- OID cannot be modified by the user.

## 2. Abstraction

- Abstraction is the process of identifying the essential aspects of an entity and ignoring the unimportant properties.
- There are two fundamental aspects of abstraction
  3. Encapsulation
  4. Information hiding

# Encapsulation

- The concept of encapsulation means that an object contains both data structure and the set of operations that can be used to manipulate it.
- Example

Define class Employee:

```
type tuple( name:      string;
            birthdate:  date;
            salary:     float; );
operations create-emp:  employee;
            destroy-emp: boolean;
```

End employee;

# Information hiding

- The concept of information hiding is that we separate the external aspects of an object from its internal details.
- The internal details are hidden from the user.
- So that the internal details can be changed without affecting the application that use it, that is the external details remain the same.
- User only knows available methods and how to call them.

# 3.Methods and Messages

- In object technology functions are usually called methods.
- Methods define the behavior of the object.
- They can be used to change the object's state by modifying its attribute values
- A method consist of a name and a body that perform the action associated with method name

# Example

```
method void updatesalary(float increment)
{
Salary=salary+increment
}
```

It is a method use to update a member of staff's salary.

## Messages

Messages are the means by which objects communicate. A message is simply a request from one object to another asking the object to execute one of its methods.



- Example

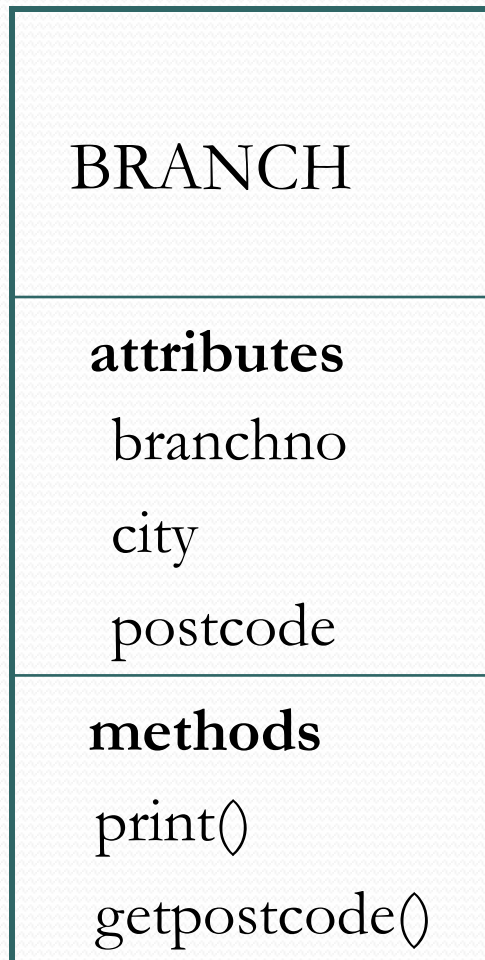
Staffobject.updatesalary(100)

## 4.Classes

Classes are used to define a set of similar objects.

- Objects having same attributes and respond to same messages can be grouped together to form a class.

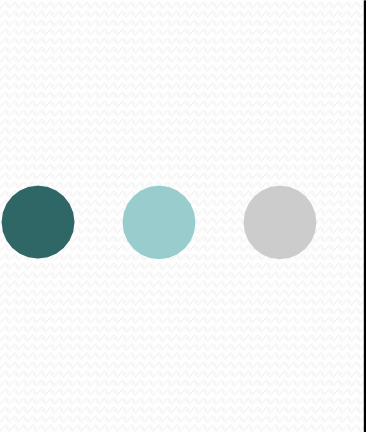
# Example class



Branchno=b003  
City=london  
Postcode=78jj

Branchno=b005  
City=london  
Postcode=09jik

# Subclasses, Superclasses and inheritance

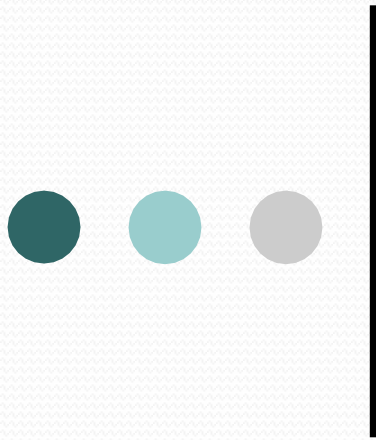


Some objects may have similar but not identical attributes and methods. If there is a large degree of similarity, it would be useful to be able to share the common properties.

Inheritance allows one class to be defined as a special case of a more general class.

These special cases are known as subclasses and the more general cases are known as superclasses.

e

- 
2. Single inheritance
  3. Multiple inheritance
  4. Repeated inheritance
  5. Selective inheritance

○ **Single inheritance:**

Single inheritance refers to the fact that the subclasses inherit from no more than one superclass.



**THANK YOU**