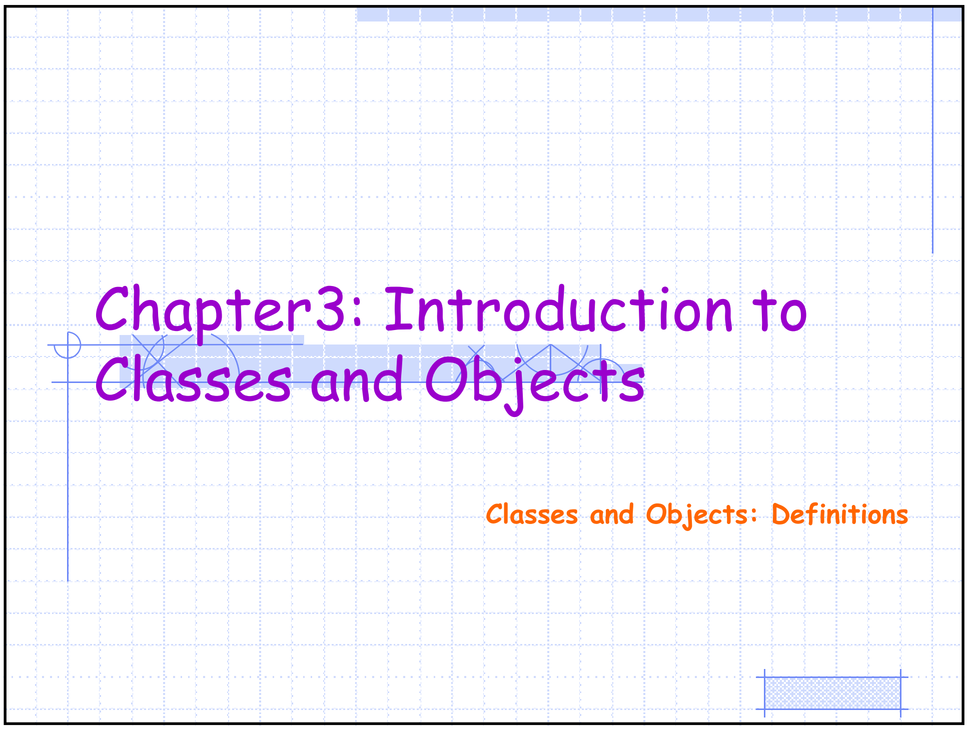
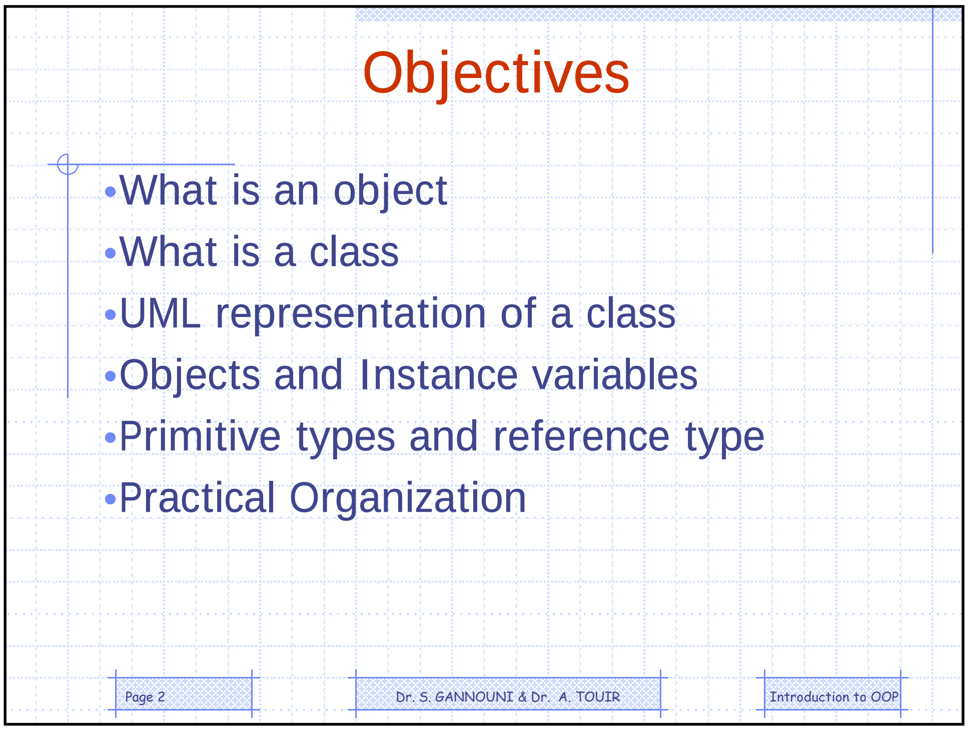


Chapter3: Introduction to Classes and Objects

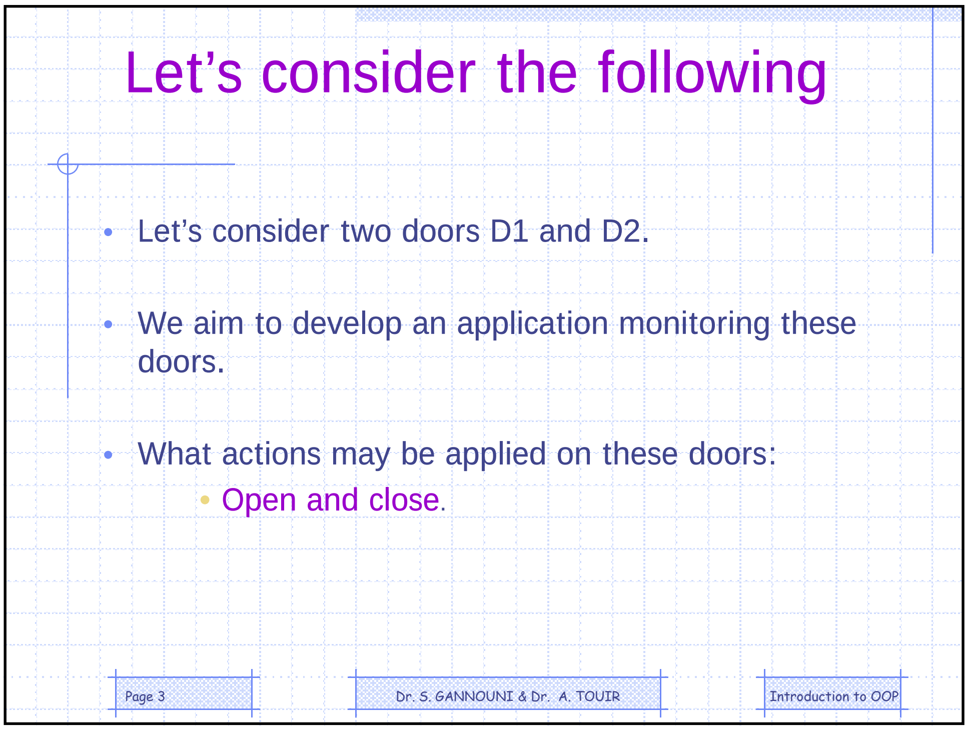
Classes and Objects: Definitions



Objectives

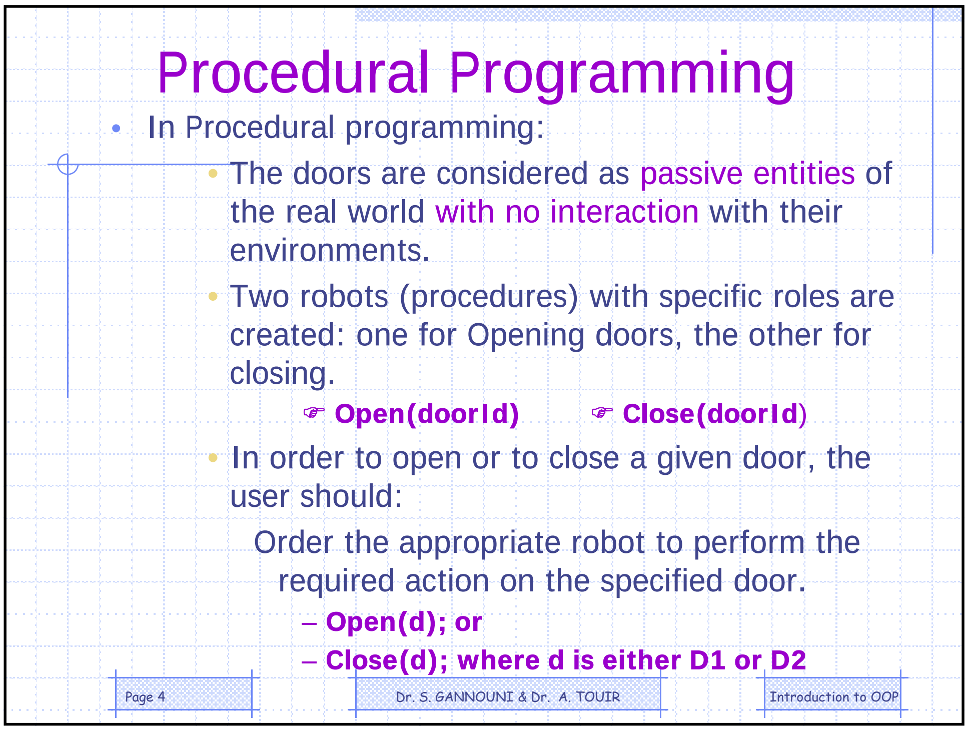
- What is an object
- What is a class
- UML representation of a class
- Objects and Instance variables
- Primitive types and reference type
- Practical Organization

Page 2 Dr. S. GANNOUNI & Dr. A. TOUIR Introduction to OOP



Let’s consider the following

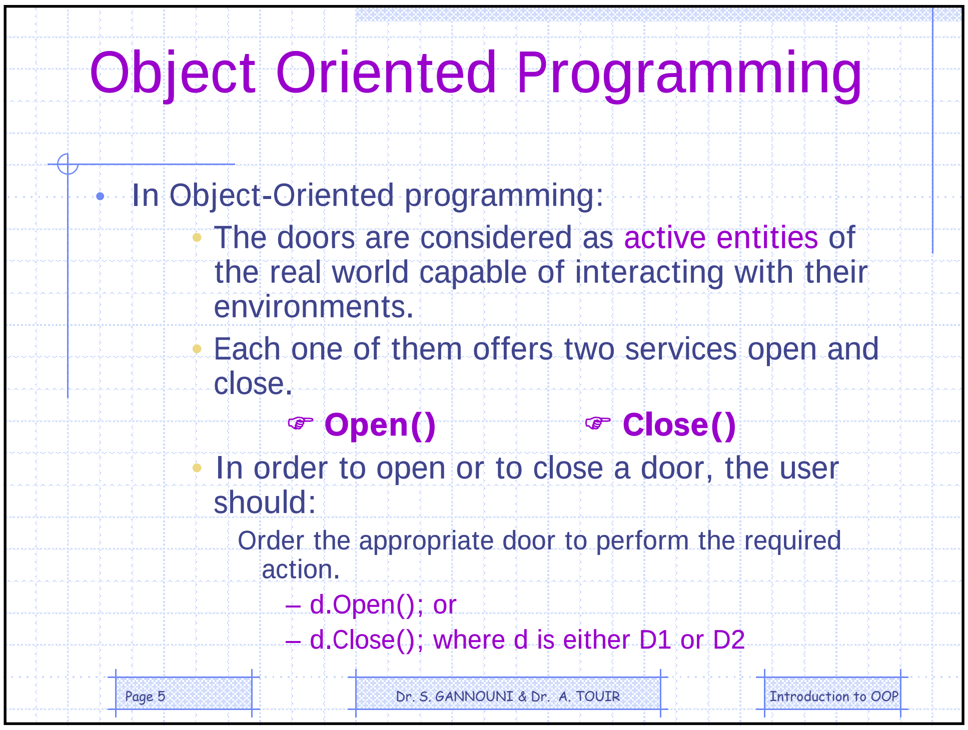
- Let’s consider two doors D1 and D2.
- We aim to develop an application monitoring these doors.
- What actions may be applied on these doors:
  - Open and close.



Procedural Programming

- In Procedural programming:
  - The doors are considered as passive entities of the real world with no interaction with their environments.
  - Two robots (procedures) with specific roles are created: one for Opening doors, the other for closing.
    - ☞ **Open(doorId)** ☞ **Close(doorId)**
  - In order to open or to close a given door, the user should:

    Order the appropriate robot to perform the required action on the specified door.
    - **Open(d); or**
    - **Close(d); where d is either D1 or D2**

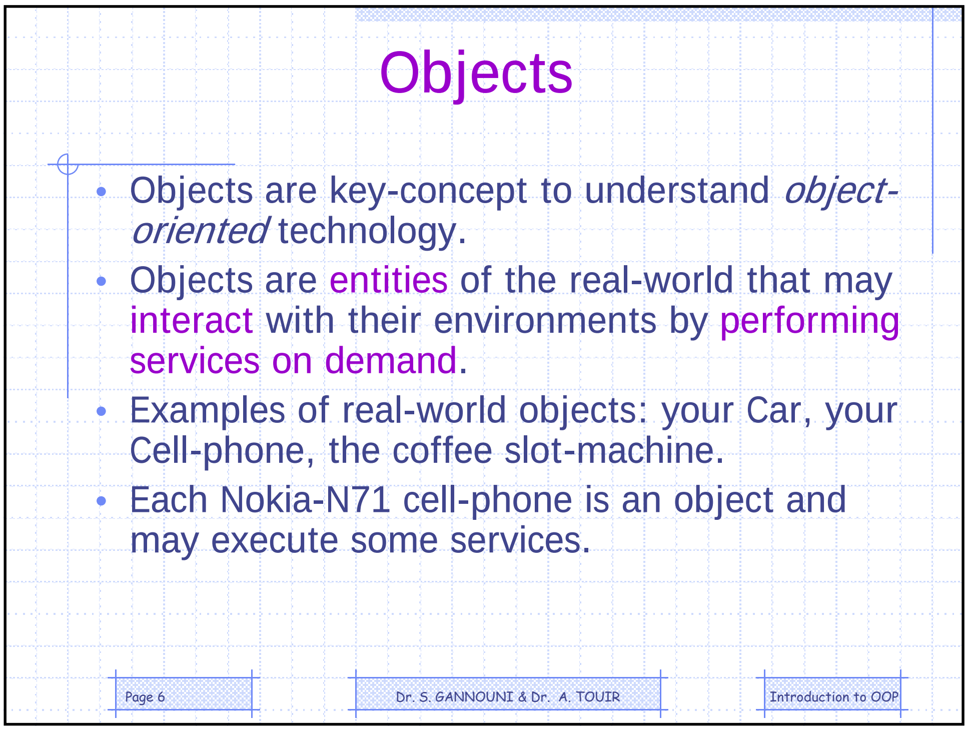


Object Oriented Programming

- In Object-Oriented programming:
  - The doors are considered as active entities of the real world capable of interacting with their environments.
  - Each one of them offers two services open and close.

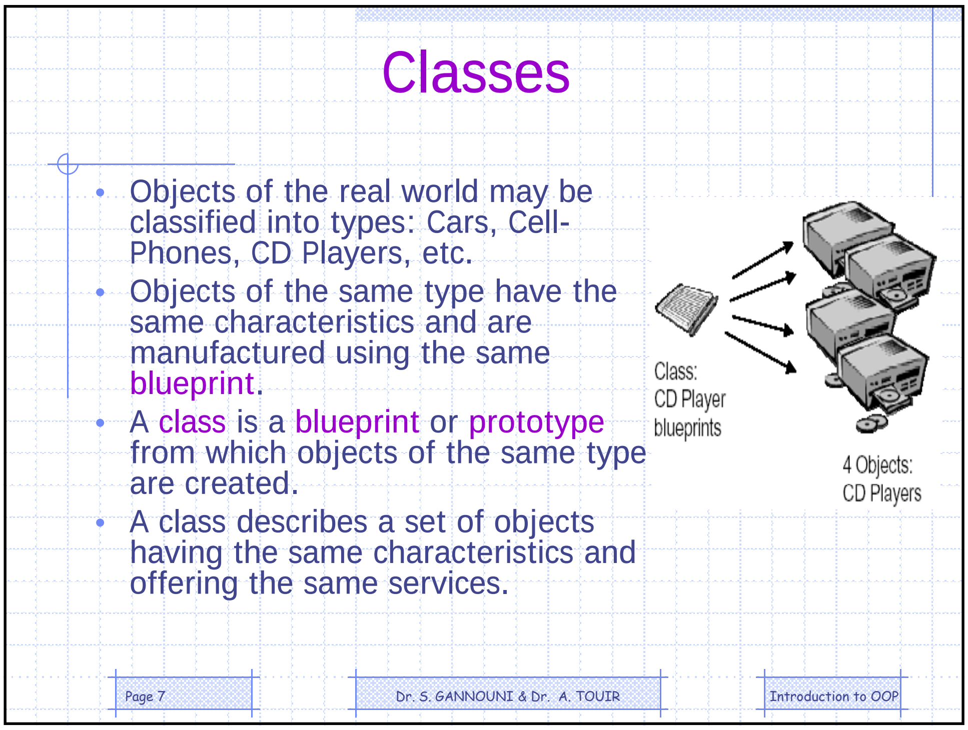
    ☞ **Open()** ☞ **Close()**
  - In order to open or to close a door, the user should:

    Order the appropriate door to perform the required action.
    – d.Open(); or
    – d.Close(); where d is either D1 or D2



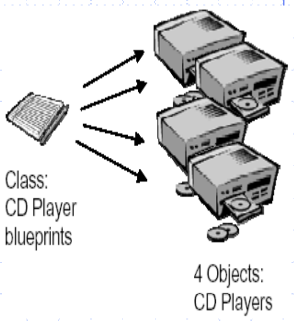
Objects

- Objects are key-concept to understand *object-oriented* technology.
- Objects are entities of the real-world that may interact with their environments by performing services on demand.
- Examples of real-world objects: your Car, your Cell-phone, the coffee slot-machine.
- Each Nokia-N71 cell-phone is an object and may execute some services.

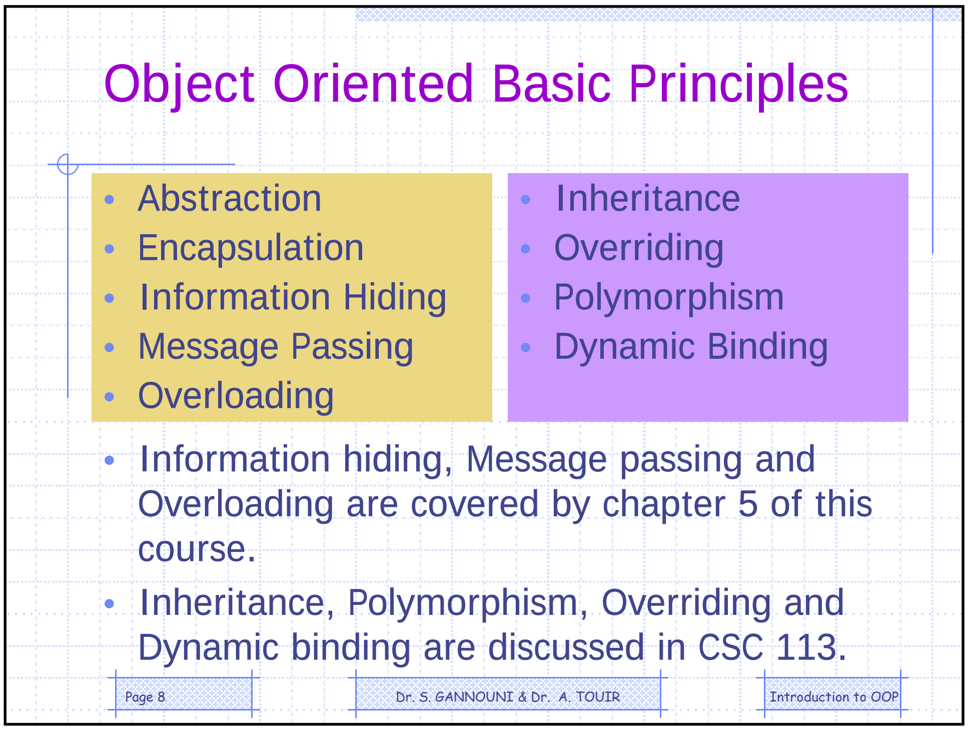


Classes

- Objects of the real world may be classified into types: Cars, Cell-Phones, CD Players, etc.
- Objects of the same type have the same characteristics and are manufactured using the same blueprint.
- A class is a blueprint or prototype from which objects of the same type are created.
- A class describes a set of objects having the same characteristics and offering the same services.



Page 7 | Dr. S. GANNOUNI & Dr. A. TOUIR | Introduction to OOP



Object Oriented Basic Principles

- Abstraction
- Encapsulation
- Information Hiding
- Message Passing
- Overloading

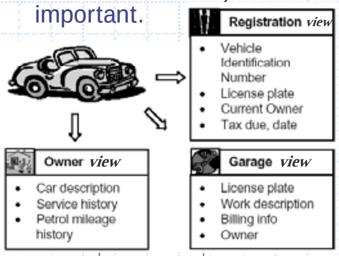
- Inheritance
- Overriding
- Polymorphism
- Dynamic Binding

- Information hiding, Message passing and Overloading are covered by chapter 5 of this course.
- Inheritance, Polymorphism, Overriding and Dynamic binding are discussed in CSC 113.

Page 8 | Dr. S. GANNOUNI & Dr. A. TOUIR | Introduction to OOP

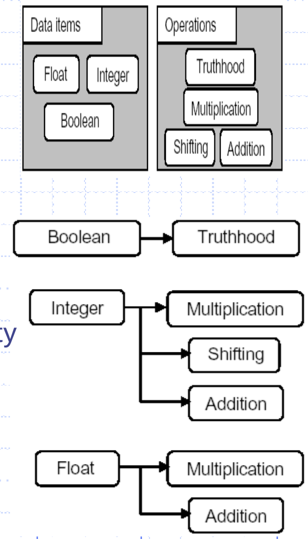
Abstraction Principle

- Data Abstraction
  - In order to process something from the real world we have to extract the essential characteristics of that object.
  - Data abstraction is the process of:
    - Refining away the unimportant details of an object,
    - Keeping only the useful characteristics that define the object.
  - For example, depending on how a car is viewed (e.g. in terms of something to be registered, or alternatively something to be repaired, etc.) different sets of characteristics will emerge as being important.
- Functionality Abstraction
  - Modeling functionality suffers from
    - unnecessary functionality may be extracted,
    - or alternatively, an important piece of functionality may be omitted.
  - Functionality abstraction is the process of determining which functionality is important.



Encapsulation Principle

- Abstraction involves reducing a real world entity to its abstraction essential defining characteristics.
- Encapsulation extends this idea by also modeling and *linking* each data of an entity to the appropriate functionality of that entity.

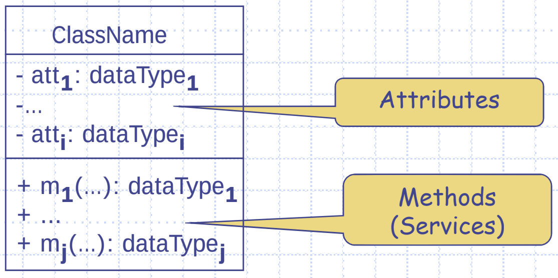


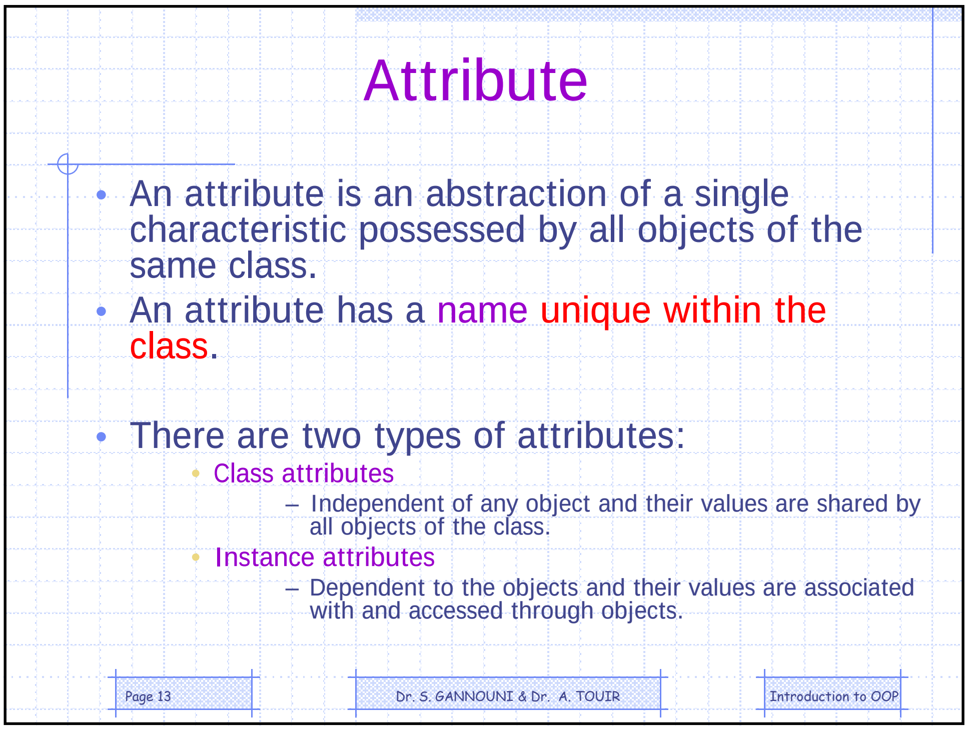
Encapsulation Gives Classes

- OOP makes use of encapsulation to ensure that data is used in an appropriate manner.
  - by preventing from accessing data in a non-intended manner (e.g. asking if an Integer is true or false, etc.).
- Through encapsulation, only a predetermined appropriate group of operations may be applied (have access) to the data.
- Place data and the operations that act on that data in the same class.
- Encapsulation is the OO principle that allows objects to contain the appropriate operations that could be applied on the data they store.
  - My Nokia-N71 cell-phone stores:
    - My contacts,
    - Missed calls
    - ... etc.
  - My Nokia-N71 may perform the following operations on the data it contains:
    - Edit/Update/Delete an existing contact
    - Add a new contact
    - Display my missed calls.
    - ...etc.

UML Representation of a Class

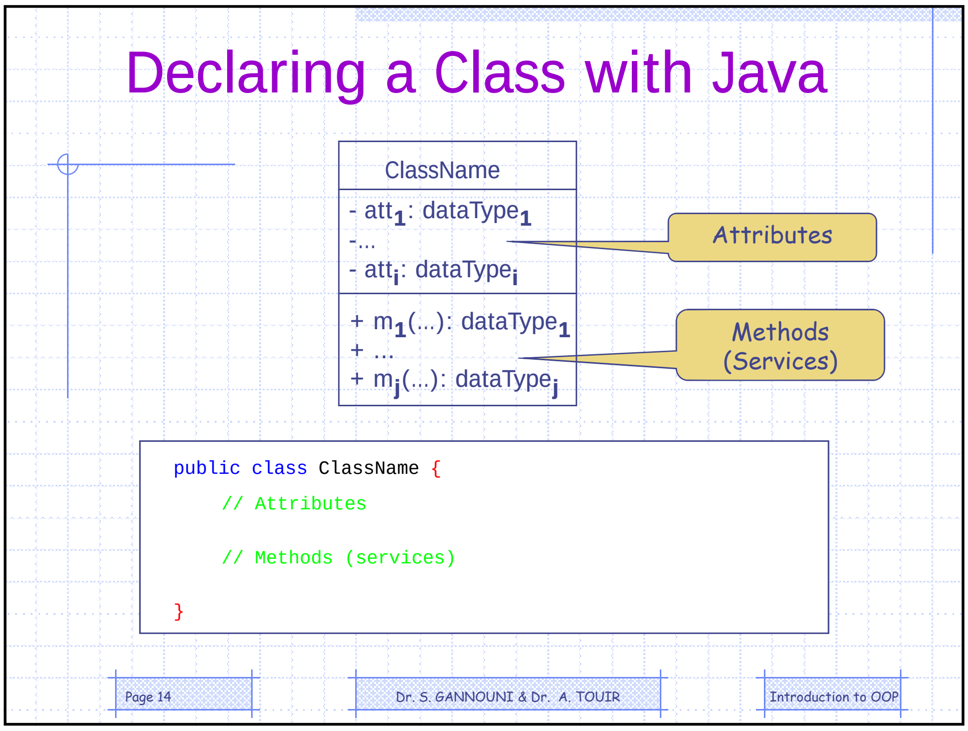
- UML represents a class with a rectangle having 3 compartments stacked vertically.
  - The top compartment shows the class's name.
  - The middle compartment lists the attributes.
  - The bottom compartment lists the operations: methods or services.



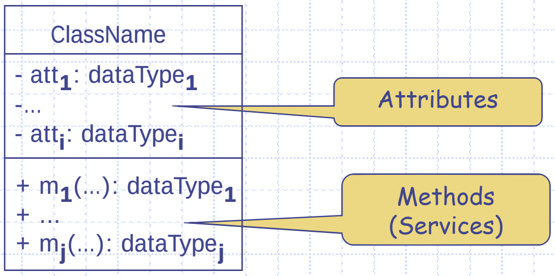


Attribute

- An attribute is an abstraction of a single characteristic possessed by all objects of the same class.
- An attribute has a name unique within the class.
- There are two types of attributes:
  - Class attributes
    - Independent of any object and their values are shared by all objects of the class.
  - Instance attributes
    - Dependent to the objects and their values are associated with and accessed through objects.



Declaring a Class with Java



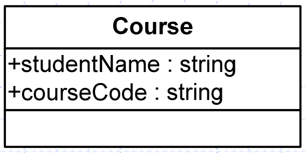
```
public class ClassName {
    // Attributes

    // Methods (services)

}
```

Declaring Attributes With Java

```
<modifiers>  <data type> <attribute name> ;
```

Modifiers	Data Type	Name
public	String	studentName ;

Example of a Class Declaration with Java



```
public class Course {

    // Attributes
    public String studentName;
    public String courseCode ;
    // No method Members

}
```