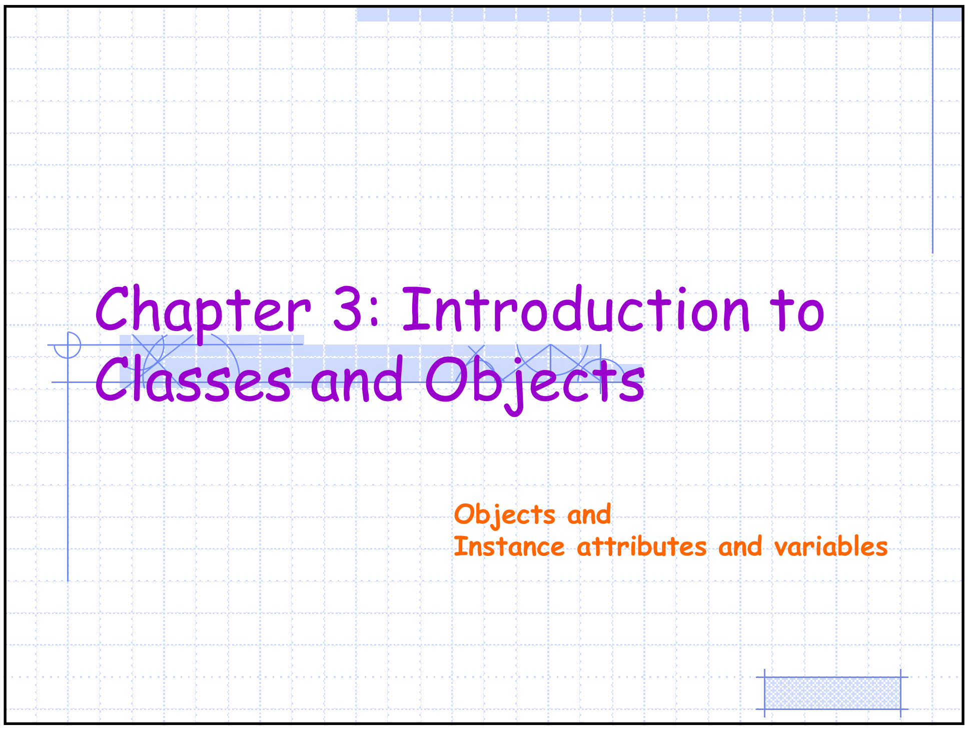
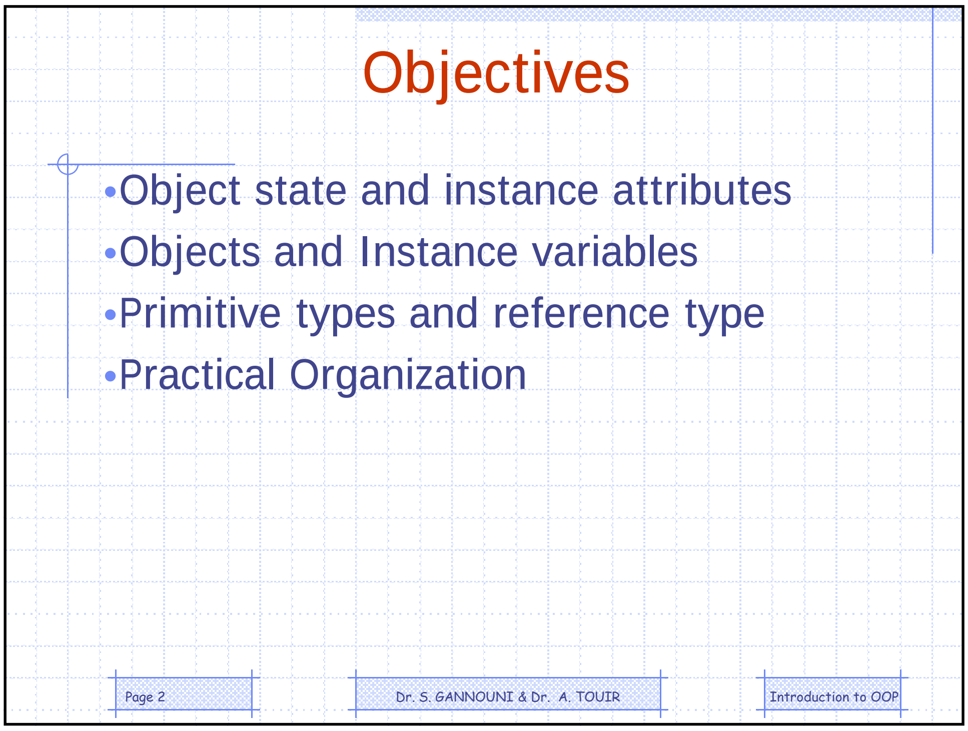


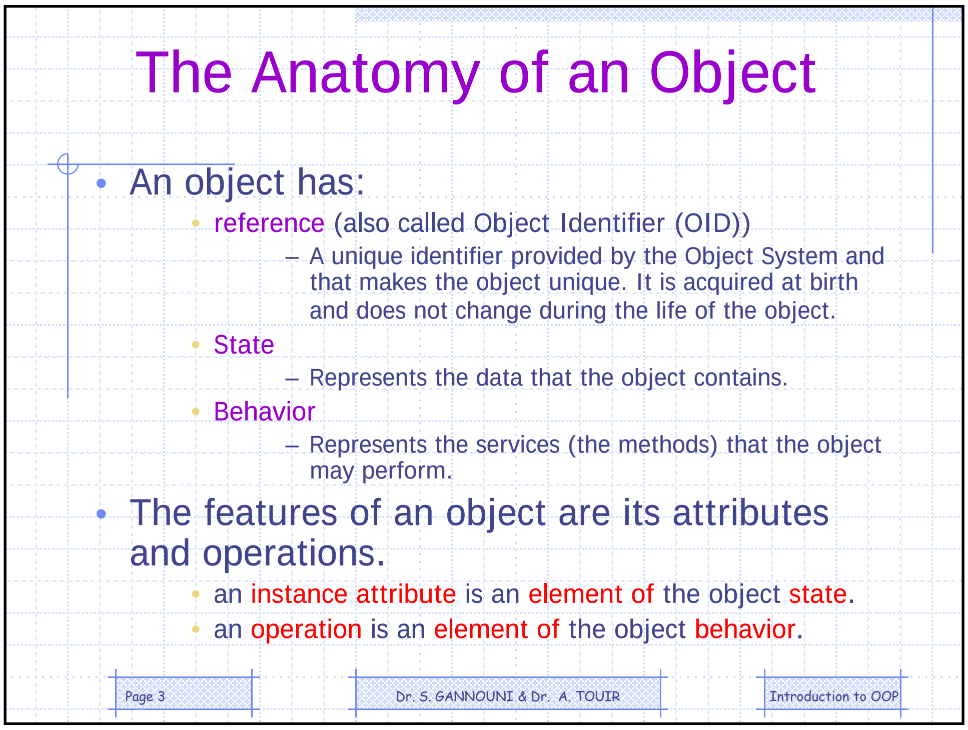
Chapter 3: Introduction to Classes and Objects

Objects and
Instance attributes and variables



Objectives

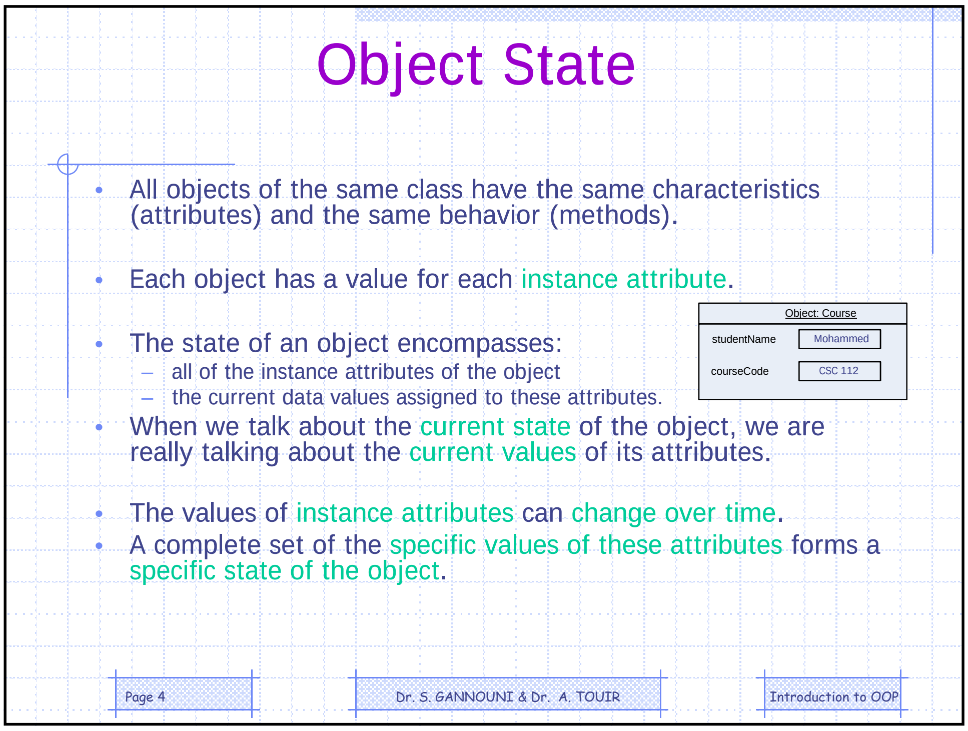
- Object state and instance attributes
- Objects and Instance variables
- Primitive types and reference type
- Practical Organization



The Anatomy of an Object

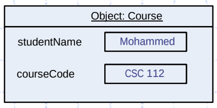
- An object has:
  - reference (also called Object Identifier (OID))
    - A unique identifier provided by the Object System and that makes the object unique. It is acquired at birth and does not change during the life of the object.
  - State
    - Represents the data that the object contains.
  - Behavior
    - Represents the services (the methods) that the object may perform.
- The features of an object are its attributes and operations.
  - an instance attribute is an element of the object state.
  - an operation is an element of the object behavior.

Page 3 | Dr. S. GANNOUNI & Dr. A. TOUIR | Introduction to OOP



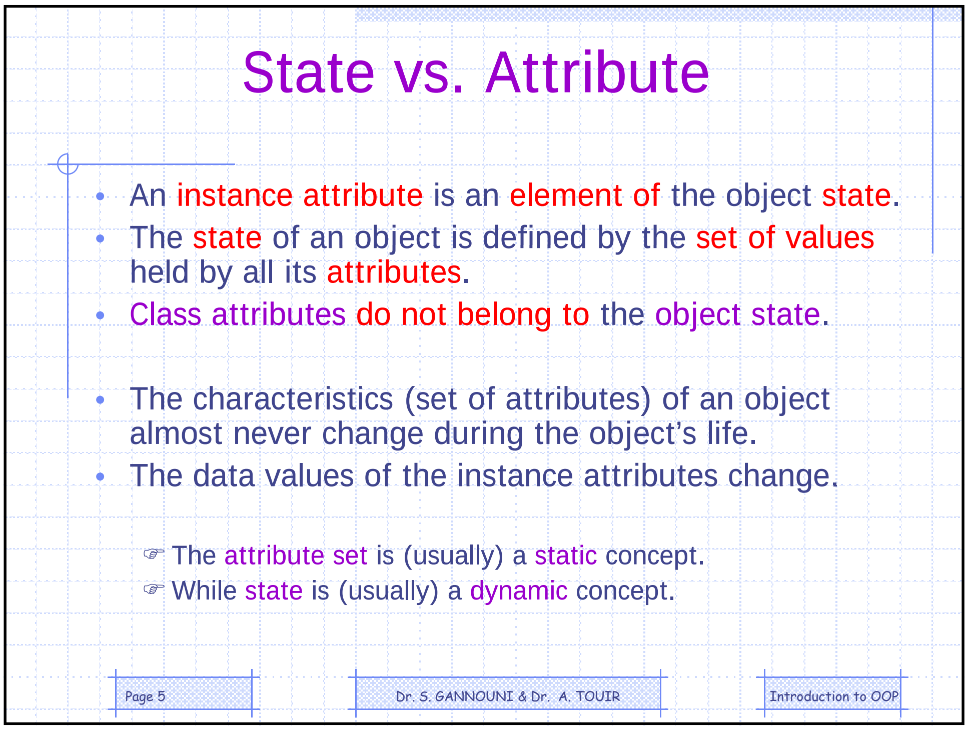
Object State

- All objects of the same class have the same characteristics (attributes) and the same behavior (methods).
- Each object has a value for each instance attribute.
- The state of an object encompasses:
  - all of the instance attributes of the object
  - the current data values assigned to these attributes.



- When we talk about the current state of the object, we are really talking about the current values of its attributes.
- The values of instance attributes can change over time.
- A complete set of the specific values of these attributes forms a specific state of the object.

Page 4 | Dr. S. GANNOUNI & Dr. A. TOUIR | Introduction to OOP

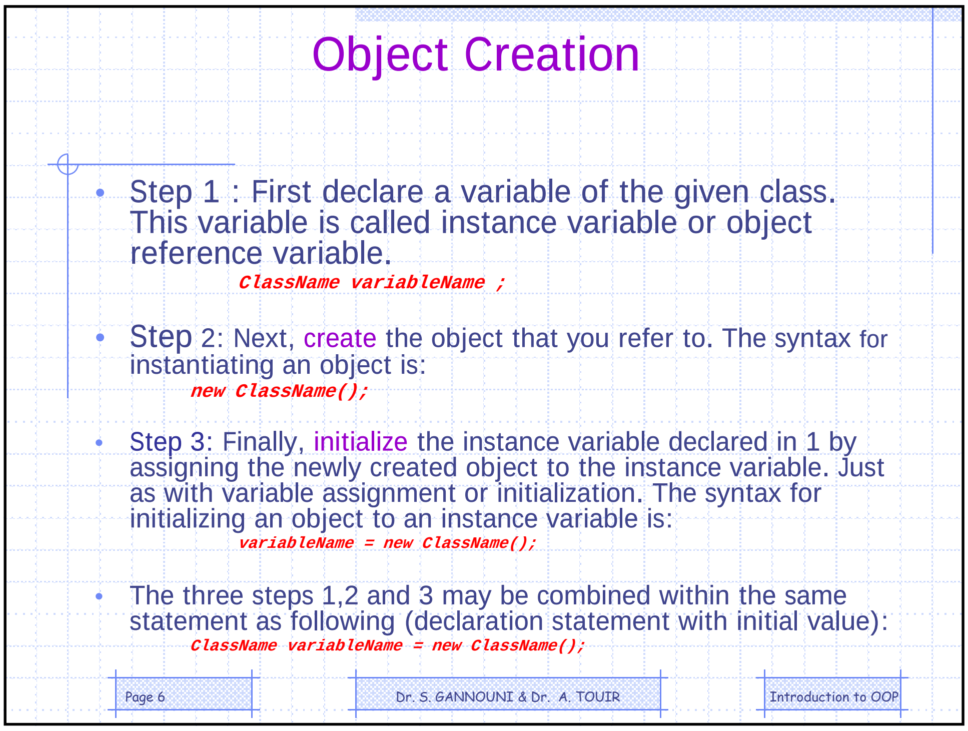


State vs. Attribute

- An instance attribute is an element of the object state.
- The state of an object is defined by the set of values held by all its attributes.
- Class attributes do not belong to the object state.

- The characteristics (set of attributes) of an object almost never change during the object's life.
- The data values of the instance attributes change.

  - ☞ The attribute set is (usually) a static concept.
  - ☞ While state is (usually) a dynamic concept.



Object Creation

- Step 1 : First declare a variable of the given class. This variable is called instance variable or object reference variable.

```
ClassName variableName ;
```

- Step 2: Next, create the object that you refer to. The syntax for instantiating an object is:

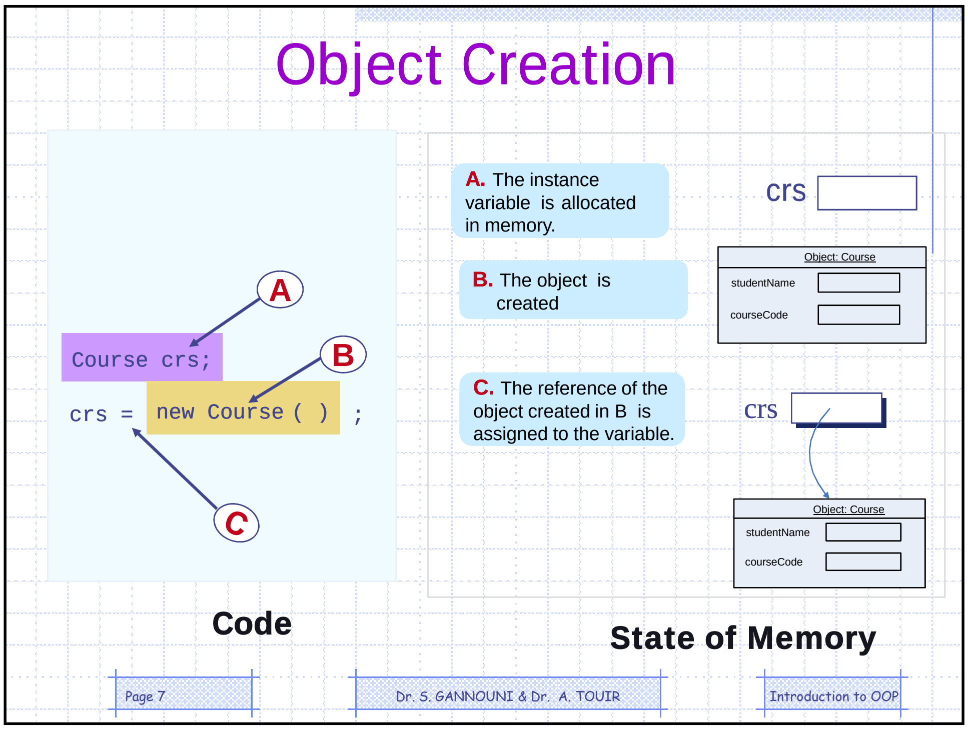
```
new ClassName();
```

- Step 3: Finally, initialize the instance variable declared in 1 by assigning the newly created object to the instance variable. Just as with variable assignment or initialization. The syntax for initializing an object to an instance variable is:

```
variableName = new ClassName();
```

- The three steps 1,2 and 3 may be combined within the same statement as following (declaration statement with initial value):

```
ClassName variableName = new ClassName();
```



Object Creation

```
Course crs;
crs = new Course ( ) ;
```

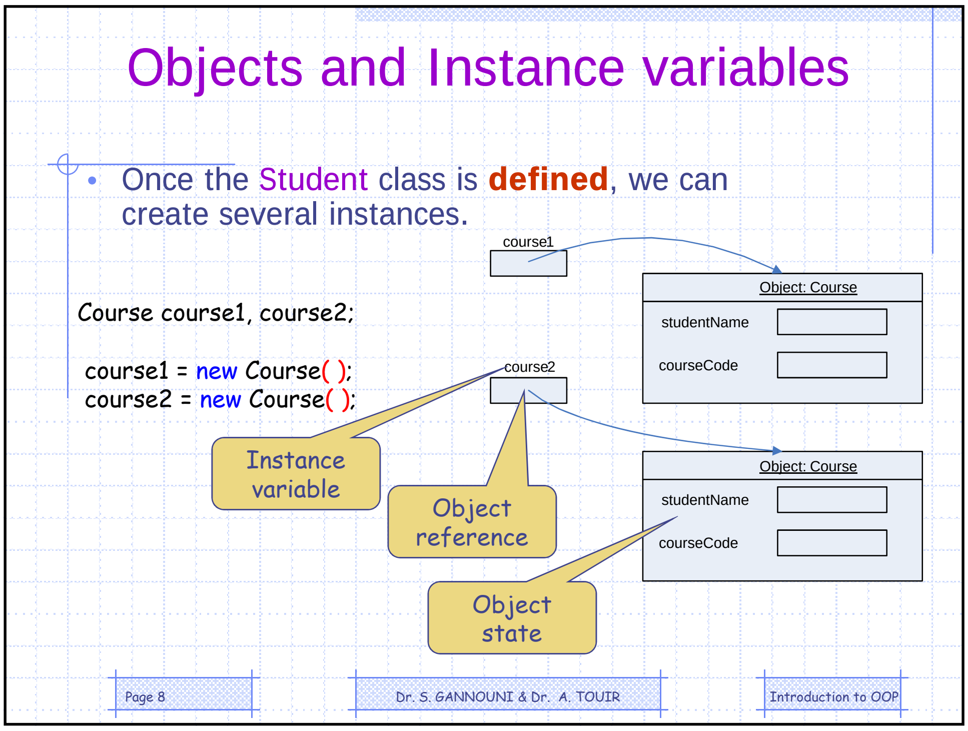
A. The instance variable is allocated in memory.

B. The object is created

C. The reference of the object created in B is assigned to the variable.

Code

State of Memory



Objects and Instance variables

- Once the Student class is **defined**, we can create several instances.

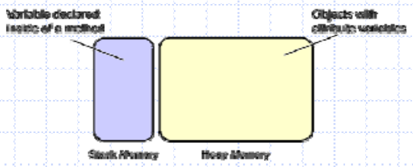
```
Course course1, course2;

course1 = new Course( );
course2 = new Course( );
```

Instance VS. Primitive Variables

- Primitive variables hold values of primitive data types.
- Instance variables hold references of objects: the location (memory address) of objects in memory.
- **Note:** Memory addresses are usually written in hexadecimal notation, beginning with a 0x (for example, 0x334009). These addresses are unique for each object and are assigned while a program runs.

Heap and Stack Memory

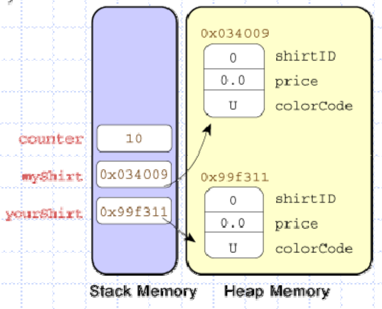
- Objects and their attributes and methods are usually stored in **heap memory**.
  - Heap memory is dynamically allocated memory chunks containing objects while they are needed by the program.
- Other variables are usually stored in **stack memory**.
  - Stack memory is used for storing items which are only used for a brief period of time (shorter than the life of an object), such as variables declared inside of a method.


Variable declared inside of a method
Objects with attribute variables
Stack Memory
Heap Memory

How Objects, Primitive and Instance Variables are Stored in Memory

- Primitive variables are stored in the **stack memory**.
- Instance variables are stored in **stack memory**.
- Objects are stored in **heap memory**.
- The ***myShirt*** and ***yourShirt*** instance variables are referring to different ***Shirt*** objects.

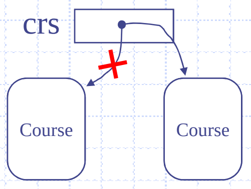
```
public static void main (String args[]) {

  int counter;
  counter = 10;
  Shirt myShirt = new Shirt ( );
  Shirt yourShirt = new Shirt ( );
}
```



Assigning Objects' References to the same Instance Variable

```
Course crs;

crs = new Course ( );

crs = new Course ( );
```

A B C

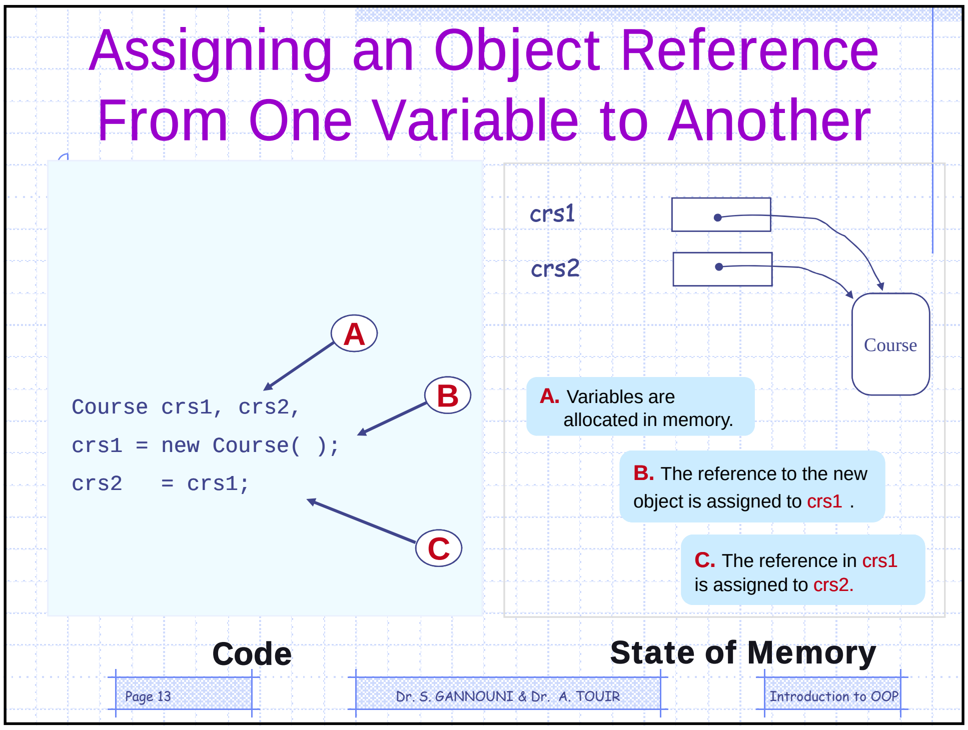


A. The variable is allocated in memory.

B. The reference to the new object is assigned to crs.

C. The reference to another object overwrites the reference in crs.

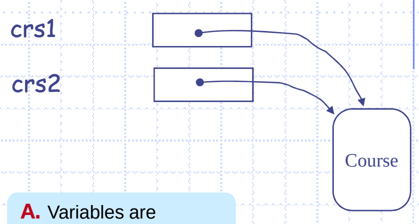
Code **State of Memory**



Assigning an Object Reference From One Variable to Another

```
Course crs1, crs2,
crs1 = new Course( );
crs2   = crs1;
```

A
B
C



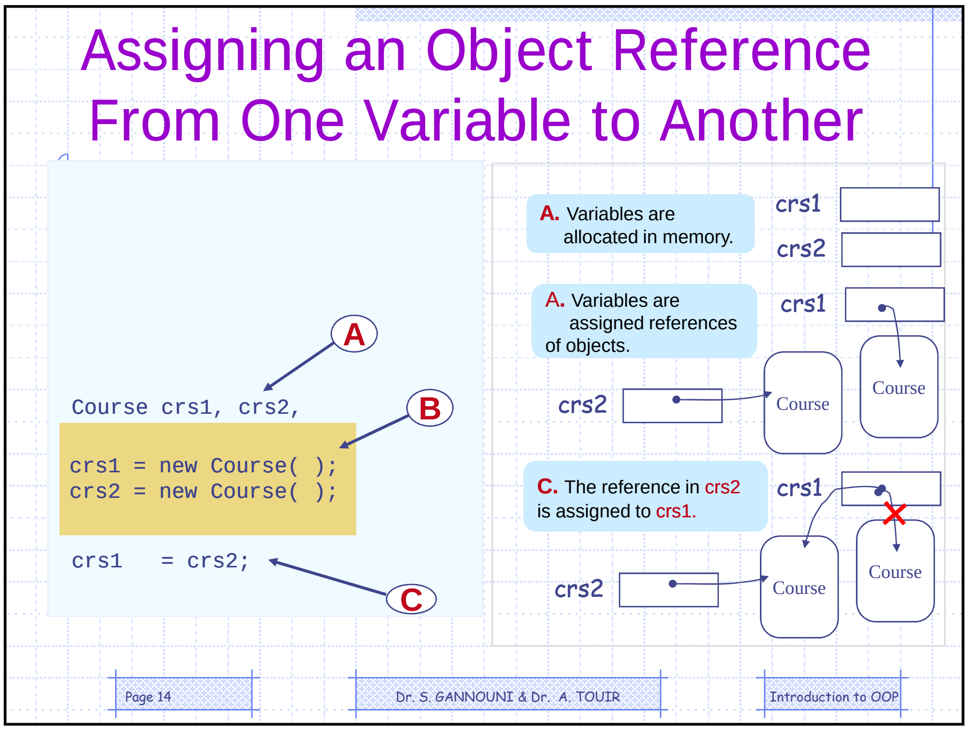
A. Variables are allocated in memory.

B. The reference to the new object is assigned to crs1 .

C. The reference in crs1 is assigned to crs2.

Code

State of Memory



Assigning an Object Reference From One Variable to Another

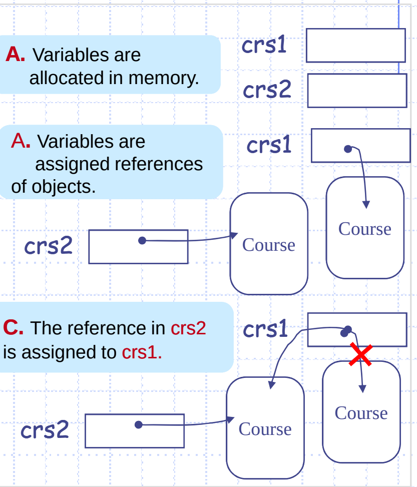
```
Course crs1, crs2,

crs1 = new Course( );
crs2 = new Course( );

crs1   = crs2;
```

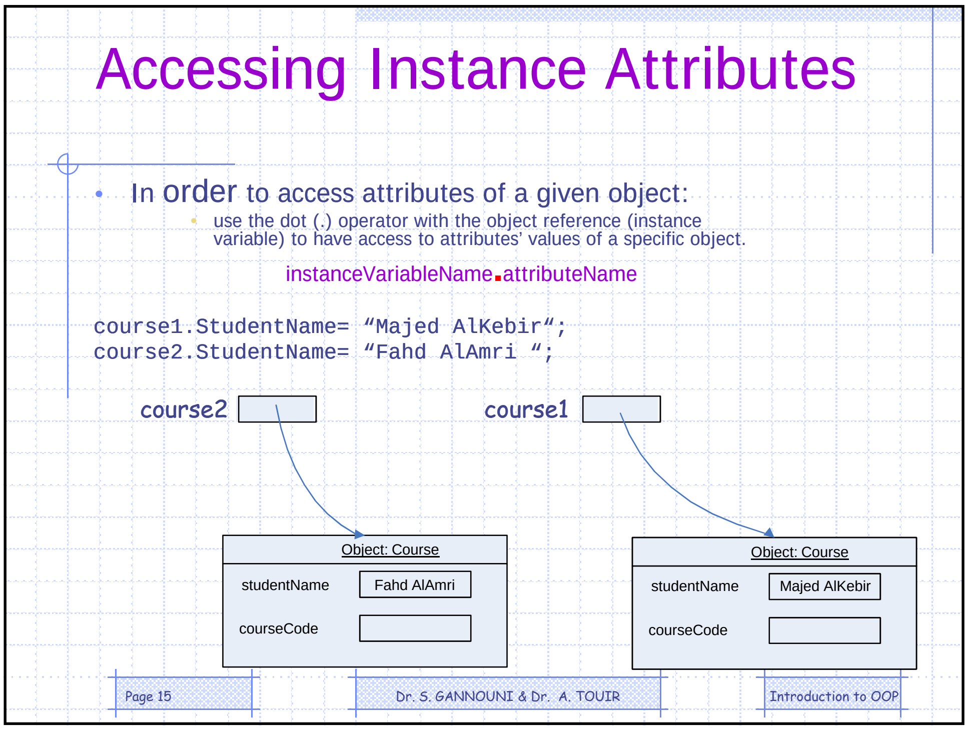
A
B
C

A. Variables are allocated in memory.

A. Variables are assigned references of objects.

C. The reference in crs2 is assigned to crs1.





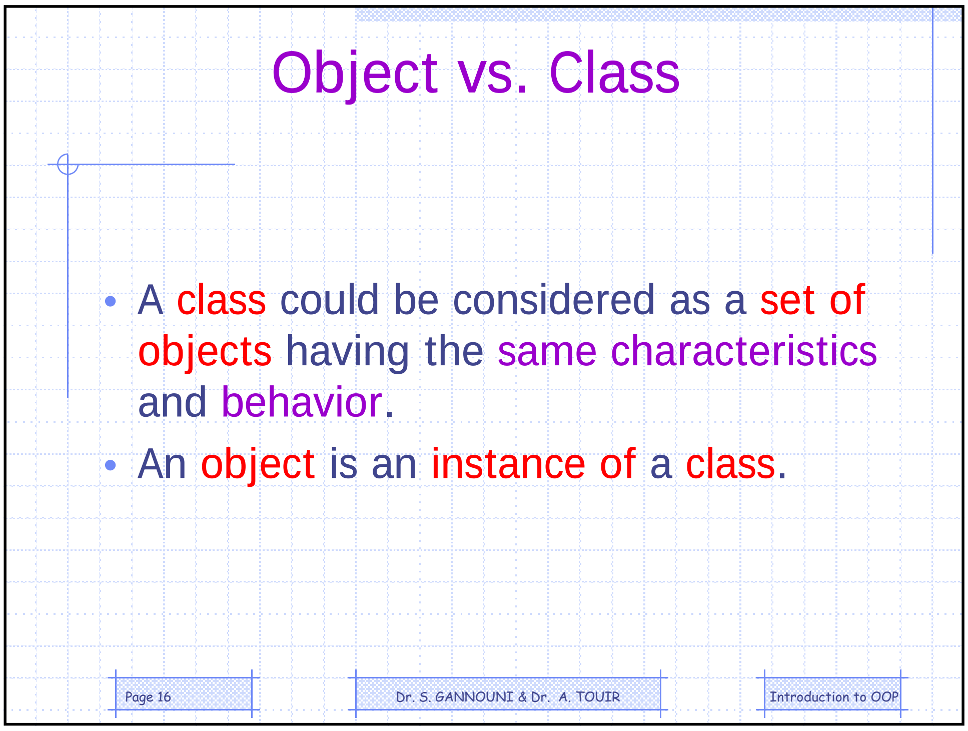
Accessing Instance Attributes

- In order to access attributes of a given object:
  - use the dot (.) operator with the object reference (instance variable) to have access to attributes' values of a specific object.

instanceVariableName.attributeName

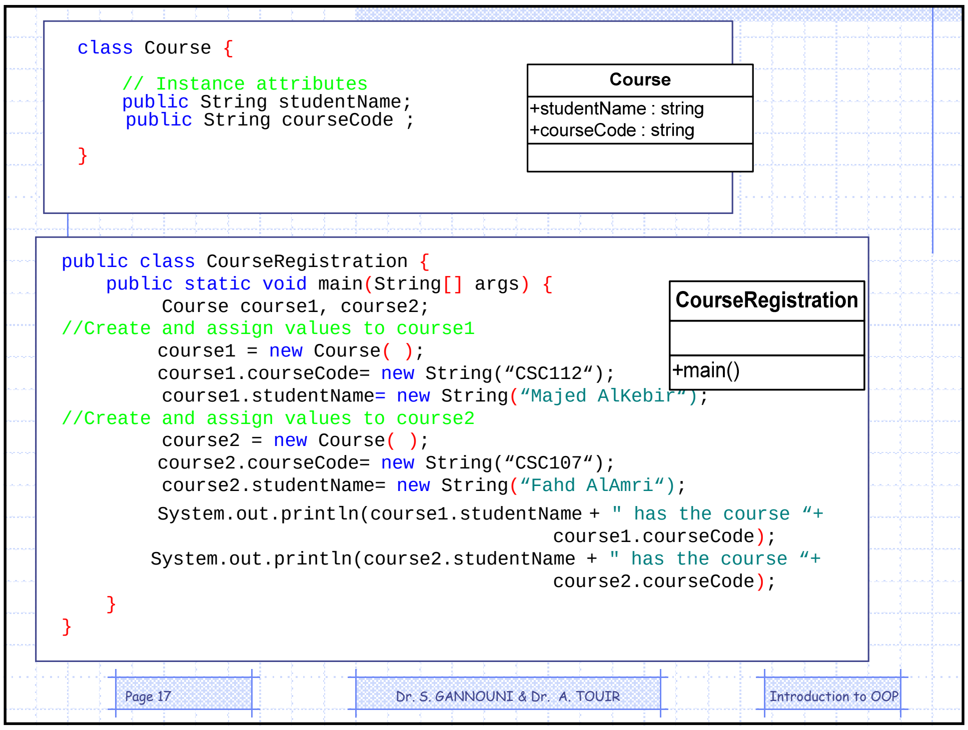
```
course1.StudentName= “Majed AlKebir“;
course2.StudentName= “Fahd AlAmri “;
```

course2
course1
Object: Course
studentName
Fahd AlAmri
courseCode
Object: Course
studentName
Majed AlKebir
courseCode

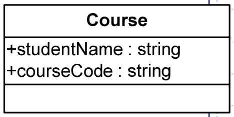


Object vs. Class

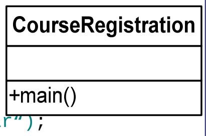
- A class could be considered as a set of objects having the same characteristics and behavior.
- An object is an instance of a class.

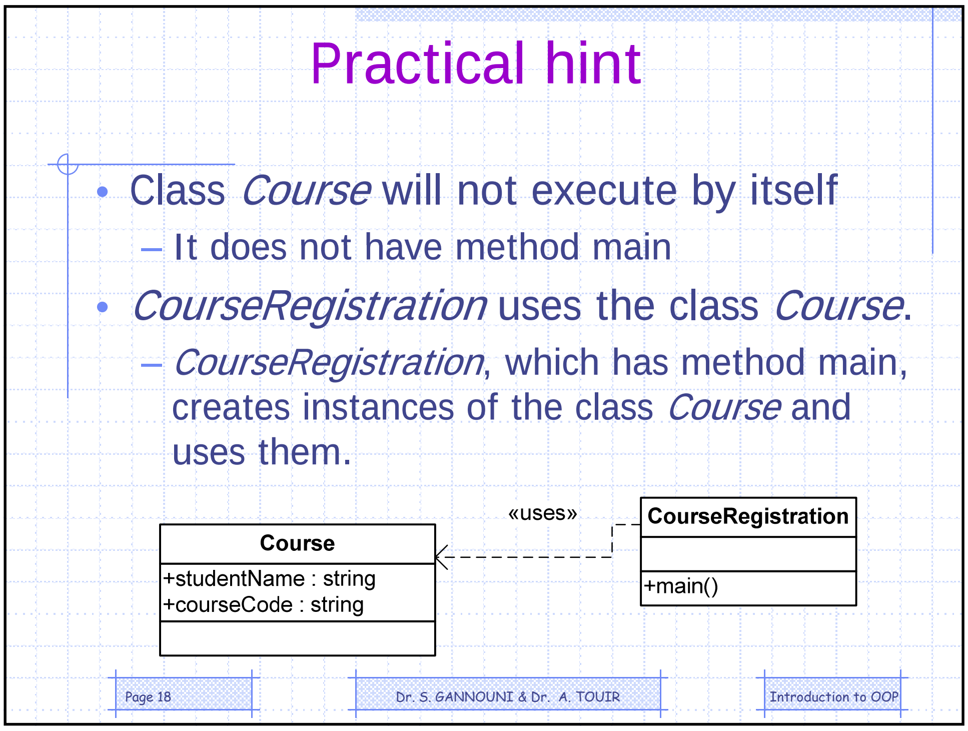


```
class Course {

    // Instance attributes
    public String studentName;
    public String courseCode ;

}
```



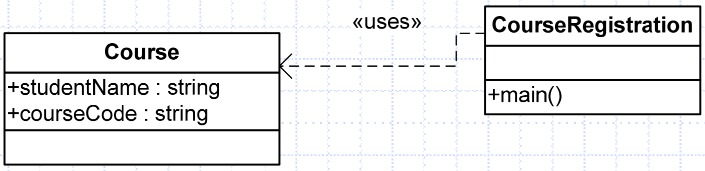
```
public class CourseRegistration {
    public static void main(String[] args) {
         Course course1, course2;
//Create and assign values to course1
        course1 = new Course( );
        course1.courseCode= new String("CSC112");
         course1.studentName= new String("Majed AlKebir");
//Create and assign values to course2
         course2 = new Course( );
        course2.courseCode= new String("CSC107");
         course2.studentName= new String("Fahd AlAmri");

        System.out.println(course1.studentName + " has the course "+
                                           course1.courseCode);
        System.out.println(course2.studentName + " has the course "+
                                           course2.courseCode);
    }
}
```

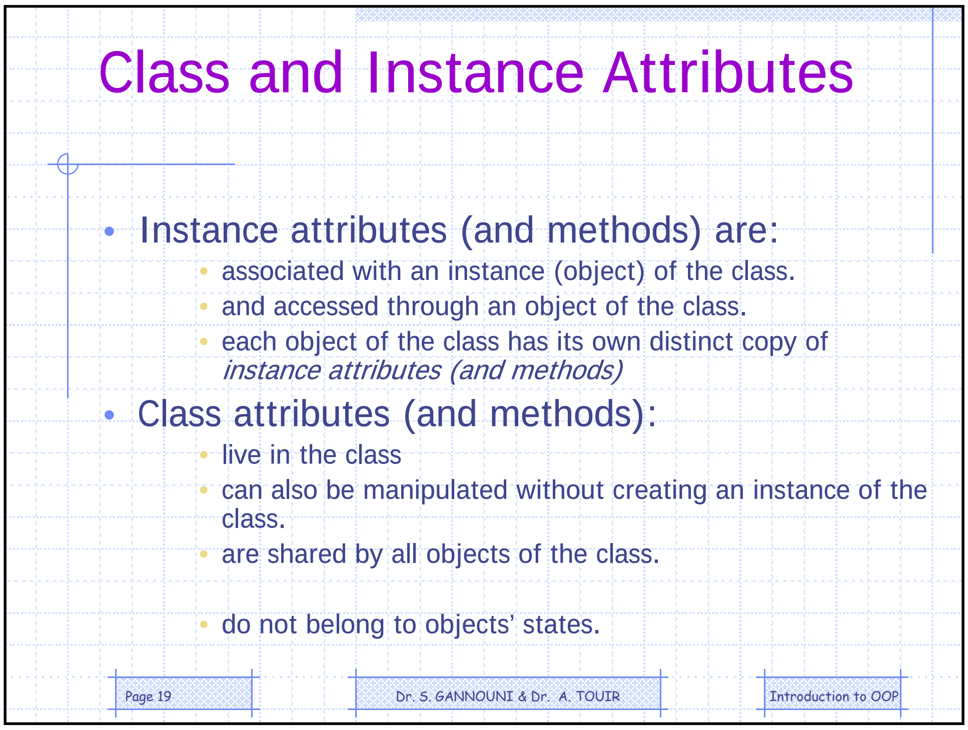




Practical hint

- Class *Course* will not execute by itself
  - It does not have method main
- *CourseRegistration* uses the class *Course*.
  - *CourseRegistration*, which has method main, creates instances of the class *Course* and uses them.

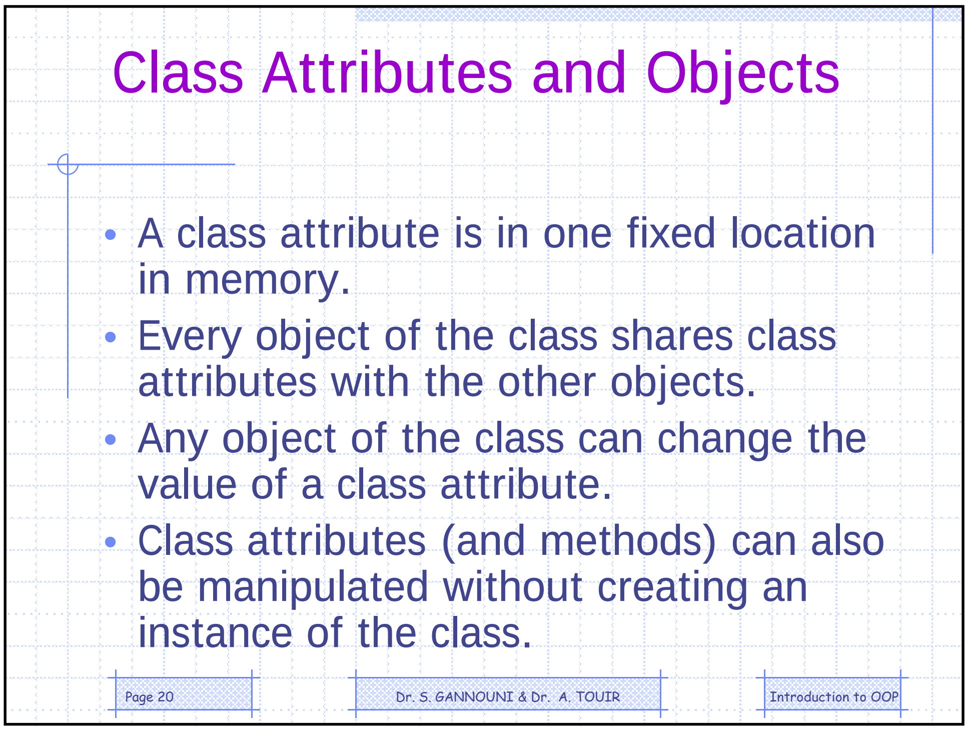




Class and Instance Attributes

- Instance attributes (and methods) are:
  - associated with an instance (object) of the class.
  - and accessed through an object of the class.
  - each object of the class has its own distinct copy of *instance attributes (and methods)*
- Class attributes (and methods):
  - live in the class
  - can also be manipulated without creating an instance of the class.
  - are shared by all objects of the class.
  - do not belong to objects' states.

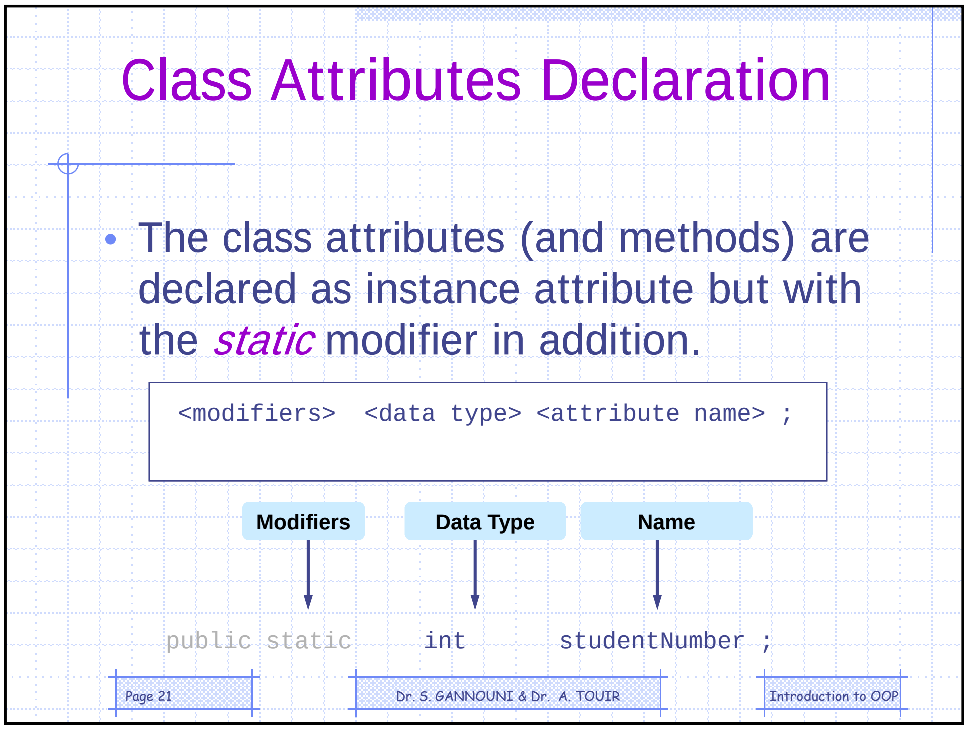
Page 19 | Dr. S. GANNOUNI & Dr. A. TOUIR | Introduction to OOP



Class Attributes and Objects

- A class attribute is in one fixed location in memory.
- Every object of the class shares class attributes with the other objects.
- Any object of the class can change the value of a class attribute.
- Class attributes (and methods) can also be manipulated without creating an instance of the class.

Page 20 | Dr. S. GANNOUNI & Dr. A. TOUIR | Introduction to OOP



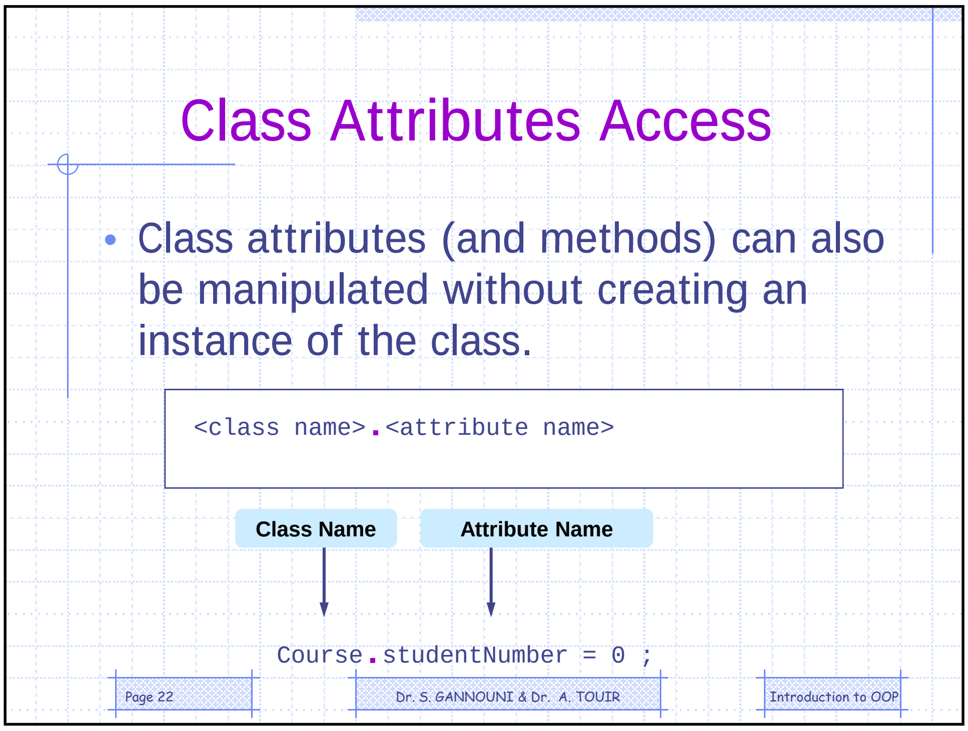
Class Attributes Declaration

- The class attributes (and methods) are declared as instance attribute but with the *static* modifier in addition.

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

Modifiers Data Type Name

```
public static     int       studentNumber ;
```



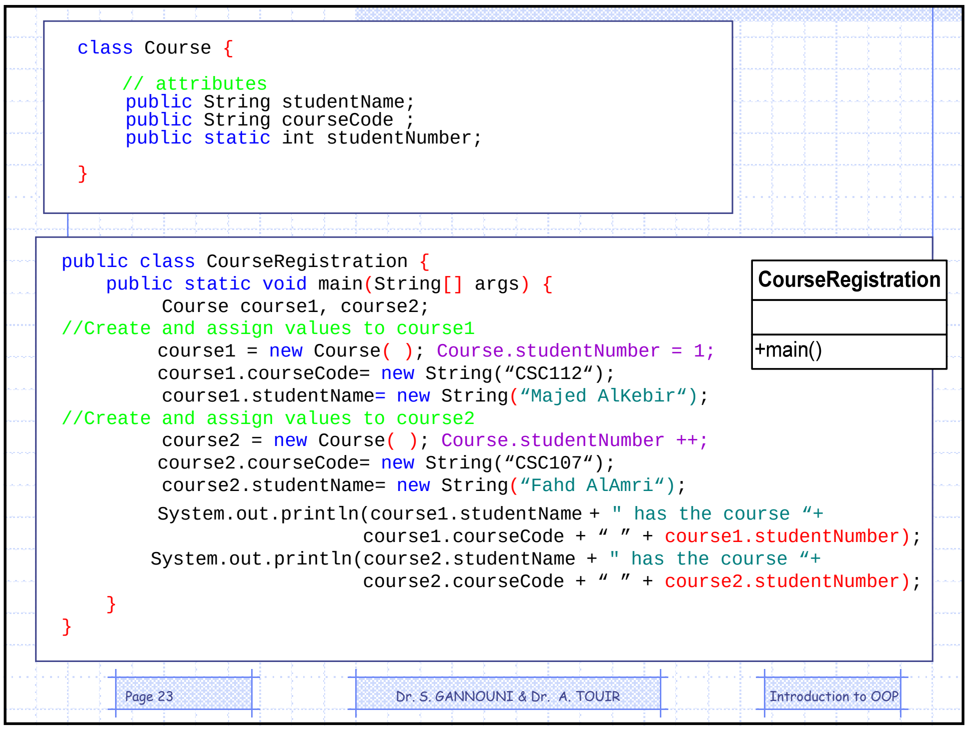
Class Attributes Access

- Class attributes (and methods) can also be manipulated without creating an instance of the class.

```
<class name>.<attribute name>
```

Class Name Attribute Name

```
Course.studentNumber = 0 ;
```



```
class Course {

     // attributes
     public String studentName;
     public String courseCode ;
     public static int studentNumber;

}
```

```
public class CourseRegistration {
    public static void main(String[] args) {
         Course course1, course2;
//Create and assign values to course1
         course1 = new Course( ); Course.studentNumber = 1;
         course1.courseCode= new String(“CSC112“);
         course1.studentName= new String(“Majed AlKebir“);
//Create and assign values to course2
         course2 = new Course( ); Course.studentNumber ++;
         course2.courseCode= new String(“CSC107“);
         course2.studentName= new String(“Fahd AlAmri“);
         System.out.println(course1.studentName + " has the course “+
                            course1.courseCode + “ ” + course1.studentNumber);
        System.out.println(course2.studentName + " has the course “+
                            course2.courseCode + “ ” + course2.studentNumber);
    }
}
```

CourseRegistration
+main()

Page 23 Dr. S. GANNOUNI & Dr. A. TOUIR Introduction to OOP