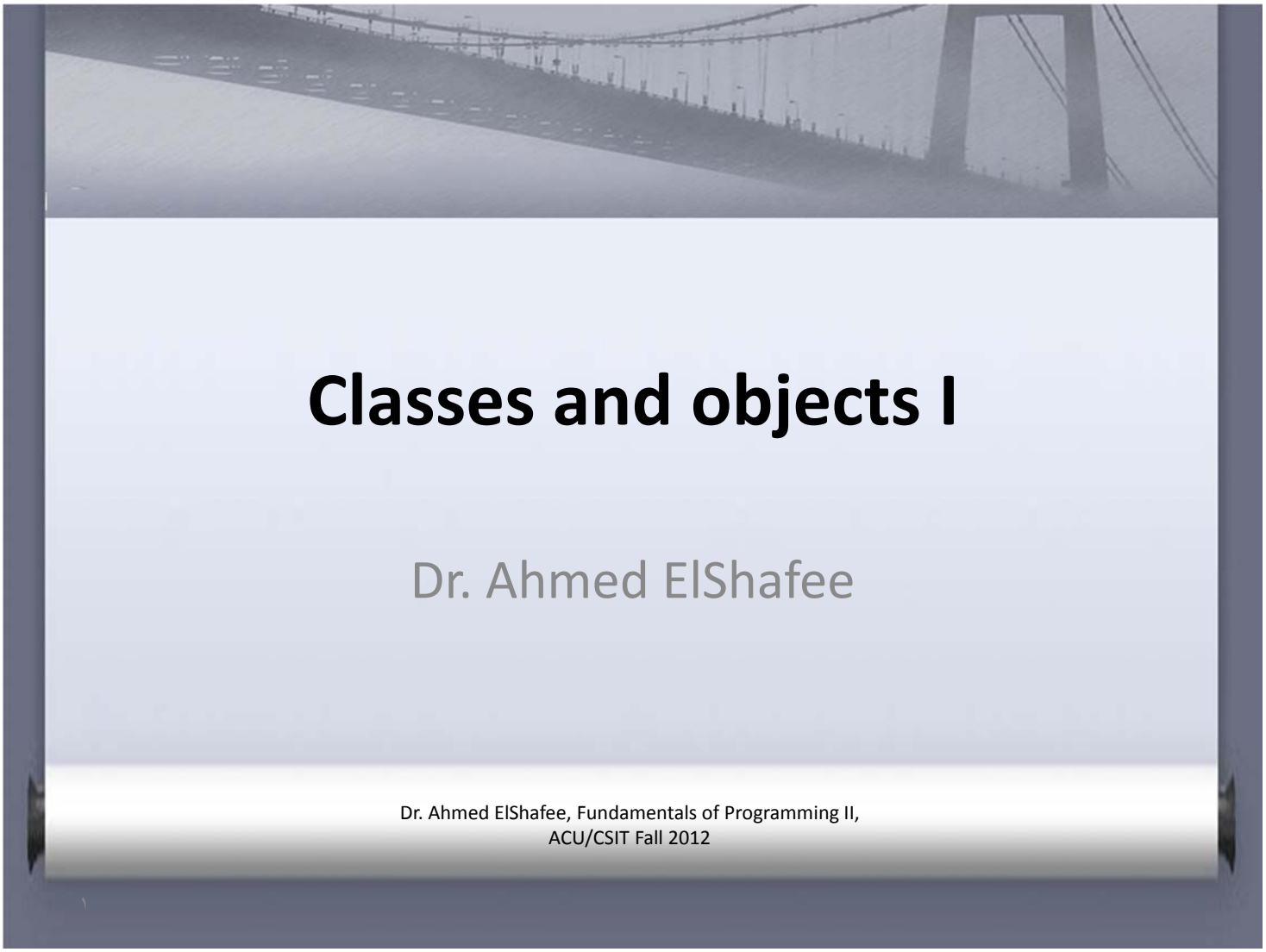


Classes and objects I

Dr. Ahmed ElShafee

Agenda

1. *
2. *
3. *

Creating a Class

- A class is a technique of using one or a group of variables to be used as a foundation for a more detailed variable.
- To create a class, you start with the **class** keyword followed by a name and its body delimited by curly brackets.

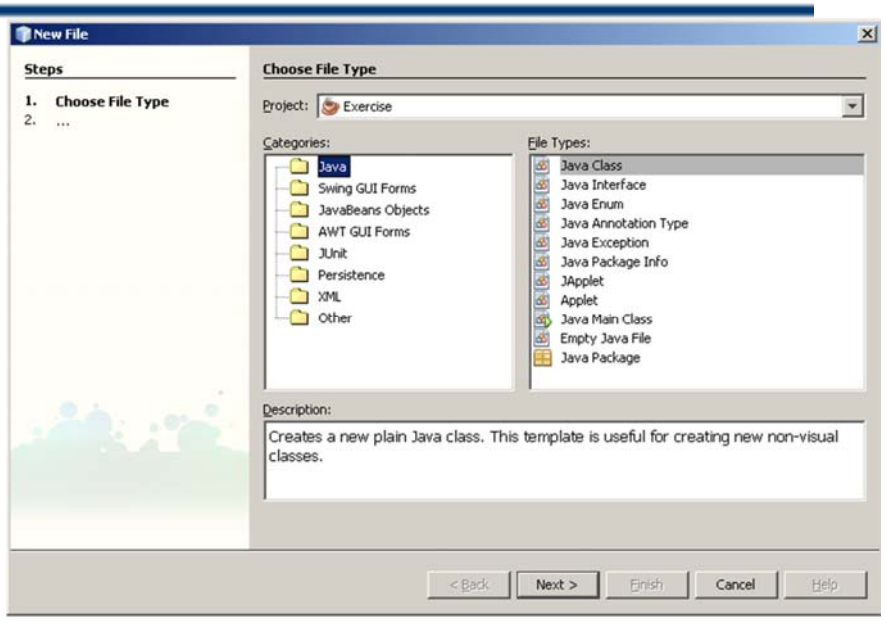
```
class House
{
}
```

- A class is created in a code file.

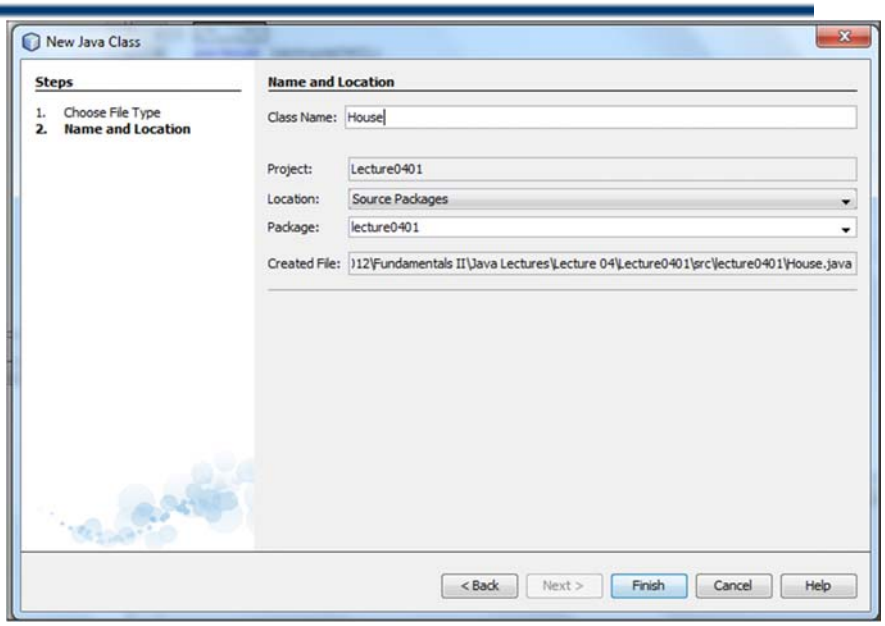
As such, you can include it in the first file of your project.

```
1  /*...*/
5  package lecture0401;
6
7  /**...*/
11 public class Lecture0401 {
12
13     /**...*/
16     public static void main(String[] args) {
17         // TODO code application logic here
18     }
19 }
20 class House
21 {
22
23 }
24
```


- You can also create a class in its own file.
- On the main menu, click File -> New File...
- On the File toolbar, click the New File button
- Type class name "House2"



- A new file contains “House2” class in created




```
1  /*...*/
5  package lecture0401;
6
7  /**...*/
11 public class Hourse2 {
12
13 }
14
```


Declaring a Variable of a Class Type

- Like any normal variable, to use a class in your program, you can first declare a variable for it like the variables we introduced before.
- This is not a value variable, it’s an object variable.
- So you need another line, to allocate memory for it using the **new** operator.

```
1  /*...*/
5  package lecture0401;
6
7  /**...*/
11 public class Lecture0401 {
12
13     /**...*/
16     public static void main(String[] args) {
17         House property;;
18     }
19 }
20 class House1
21 {
22
23 }
```

```
1  /*...*/
5  package lecture0401;
6
7  /**...*/
11 public class Lecture0401 {
12
13     /**...*/
16     public static void main(String[] args) {
17         House1 property1;
18         property1=new House1();
19     }
20 }
21 class House1
22 {
23
24 }
25
```

- You can do the both steps in a single line

```
1  /*...*/
5  package lecture0401;
6
7  /**...*/
11 public class Lecture0401 {
12
13     /**...*/
16     public static void main(String[] args) {
17         House1 property1;
18         property1=new House1();
19         House1 property2 = new House1();
20     }
21 }
22 class House1
23 {
24
25 }
```


Sharing a Class

- When creating a class, if you want it to be accessible by code in other files (packages), precede the **class** keyword with **public** when creating it.
- If the **class** keyword is preceded by **public**, the class must be created in its own file.

Class' Member Variables

- The section between the curly brackets, { and }, of a class is referred to as its body.
- In the body of a class, you can create a list of the parts that make up the class.
- Each of these parts must be a complete variable with a name and a data type.
- For example, here are the characteristics that make up a house,

```
25  class House {
26
27      long propertyNumber;
28      String propertyType;
29      byte Stories;
30      public int Bedrooms;
31      double MarketValue;
32  }
33
```


Initializing an Object

- After declaring an instance of a class, you can access each of its members and assign it the desired value.

```
    public static void main(String[] args) {
        House property = new House();
        property.propertyNumber = 283795;
        property.propertyType = "Single Family";
        property.Bedrooms = 4;
        property.MarketValue = 652880;
    }
}
```

- Once a member variable has been initialized, you can use the dot operator to access it and retrieve its value:

```
    /**...*/
    public static void main(String[] args) {
        House property = new House();
        property.propertyNumber = 283795;
        property.propertyType = "Single Family";
        property.Bedrooms = 4;
        property.MarketValue = 652880;
        System.out.println("=//= Altair Realty =//=");
        System.out.println("Properties Inventory");;
        System.out.println("Property #: " + property.propertyNumber);
        System.out.println("Property Type: " + property.propertyType);
        System.out.println("Bedrooms: " + property.Bedrooms);
        System.out.println("Market Value: " + property.MarketValue);
    }
}
```


Lecture0401

check Lab manual

The Methods of a Class

- A method is simply a section of code that takes care of a particular detail for the functionality of the class.
- To create a method, you specify its name, which follows the rules we defined for **variables**.

```
20
21    class House {
22
23        long propertyNumber;
24        String propertyType;
25        byte Stories;
26        public int Bedrooms;
27        double MarketValue;
28
29        void Display() {
30        }
31    }
```


- After creating a method, in its body delimited by its curly brackets, you can define the desired behavior.

```
29        void Display() {
30            System.out.println("=//= Altair Realty =//=");
31            System.out.println("Properties Inventory");
32            System.out.println("Property Type: " + propertyType);
33            System.out.println("Bedrooms: " + Bedrooms);
34        }
35    }
```

Lecture0402

Check Lab manual

Lecture0403

Check Lab manual

The Static Members of a Class "Static Fields"

- A variable you have declared of a class is also called an instance of the class.
- In the same way, you can declare various instances of the same class as necessary:
- Each one of these instances gives you access to the members of the class but each instance holds the particular values of the members of its instance.

```
Book book1= new Book();
Book book2= new Book();
```


lecture0404

Check Lab manual

- In your application, you can declare a class member and refer to it regardless of which instance of an object you are using. Such a member variable is called static.
- To declare a member variable of a class as static, type the **static** keyword on its left.
- Whenever you have a static member, in order to refer to it, you must "qualify" it in the class in which you want to call it.
- Qualifying a member means you must specify its class.

Lecture0405

Check Lab manual

Lecture0406

- you don't need to declare a variable of their class in order to access them.

Check lab manual

Lecture0407

- You can also declare member variables of the main class as static.
- If you are referring to a static member variable in the same class in which it was declared, you don't have to qualify it.

Check lab manual

The Static Members of a Class Static Methods

- Like a member variable, a method of a class can be defined as static.
- Consequently, this particular method can access any member of the class regardless of the instance if there are many instances of the class declared.

Lecture0408

Check lab manual

Characteristics of Members of a Class Constants

- You can create a constant variable in a class. To create a constant variable, type the **final** keyword to the left of the variable.
- Once again, when declaring a constant, you must initialize it with an appropriate constant value.

Characteristics of Members of a Class this Instance

- If a class contains member variables and methods, the (non-static) member variables are automatically available to the method(s) of the class, even member variables that are private.
- When accessing a member variable or a method from another method of the class, to indicate that the member you are accessing belongs to the same class, you can precede it with the **this** member and the period operator.
- When using the **this** member variable (in C/C++, it is a pointer), you can access any member of a class within any method of the same class.

There are rules you must observe when using **this**:

- The **this** member can never be declared: it is automatically implied when you create a Class **this** cannot be used in a class A to access a member of class B.
- **this** cannot be used in a **static** method

Lecture0410

Check lab manual

Lecture0411 (Class methods)

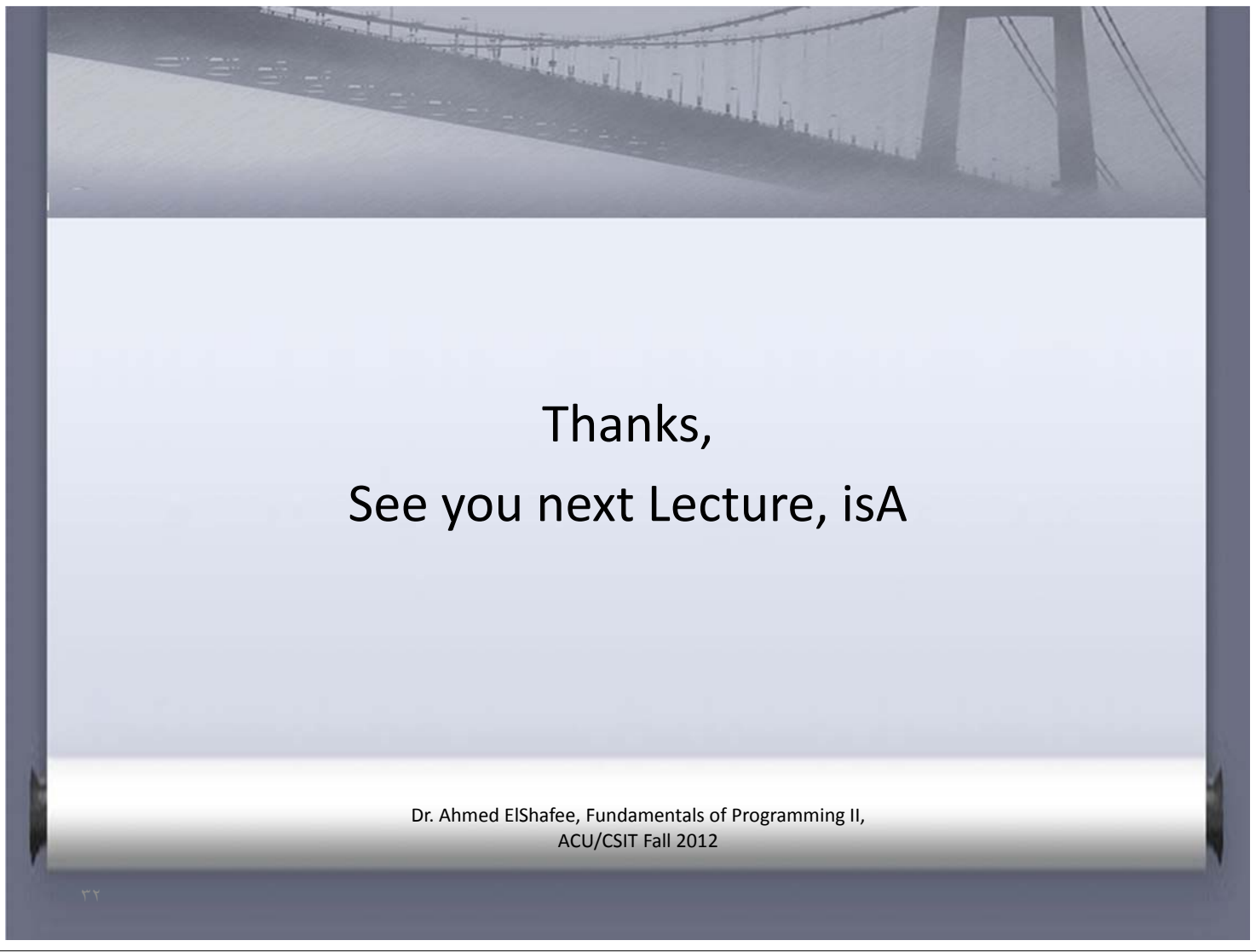
Check lab manual

Lecture0412 (Class methods)

Check lab manual

Lecture0413 (**Methods' Arguments**)

Check lab manual



Thanks,

See you next Lecture, isA