Java Programming

Classes

Class Definition

Arash Habibi Lashkari

Ph.D. Candidate of UTM University

Kuala Lumpur, Malaysia

INTRODUCTION

- OO programmers usually define their own classes apart from using classes which are already provided to them.
- The following aspects of defining classes will be described:
  - Writing Java class definitions
  - Defining object attributes
  - Defining object methods

JAVA CLASS DEFINITION

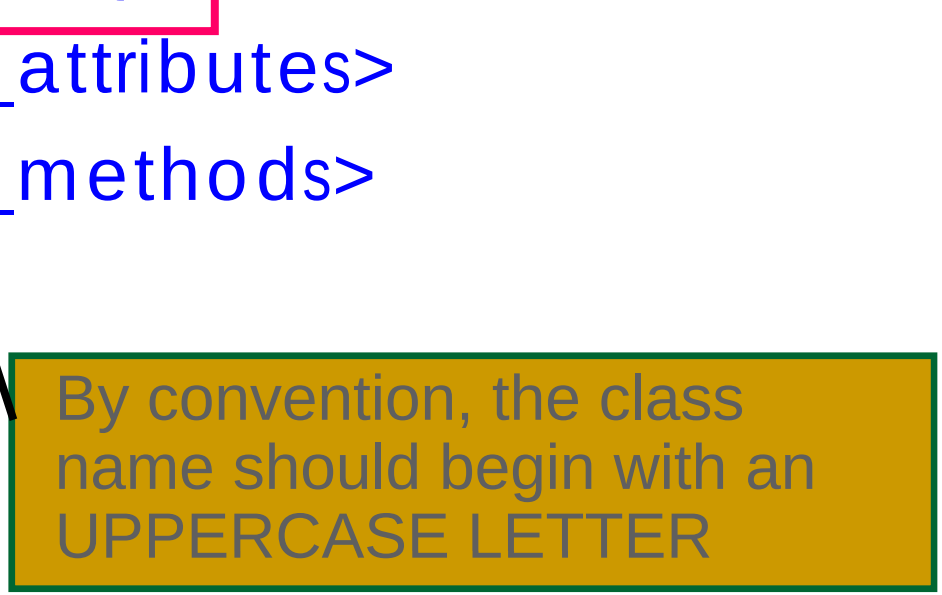
- A simple form of Java class definition:

```
class <class_name> {
  <list_of_object_attributes>
  <list_of_object_methods>
}
```

JAVA CLASS DEFINITION

- A simple form of Java class definition:

```
class <class_name> {
  <list_of_object_attributes>
  <list_of_object_methods>
}
```



JAVA CLASS DEFINITION

Example:

A Java class definition for a **`Subject`** class:

```
class Subject {
    private String name;
    public String getName() {
        return name;
    }
}
```

JAVA CLASS DEFINITION

Example:

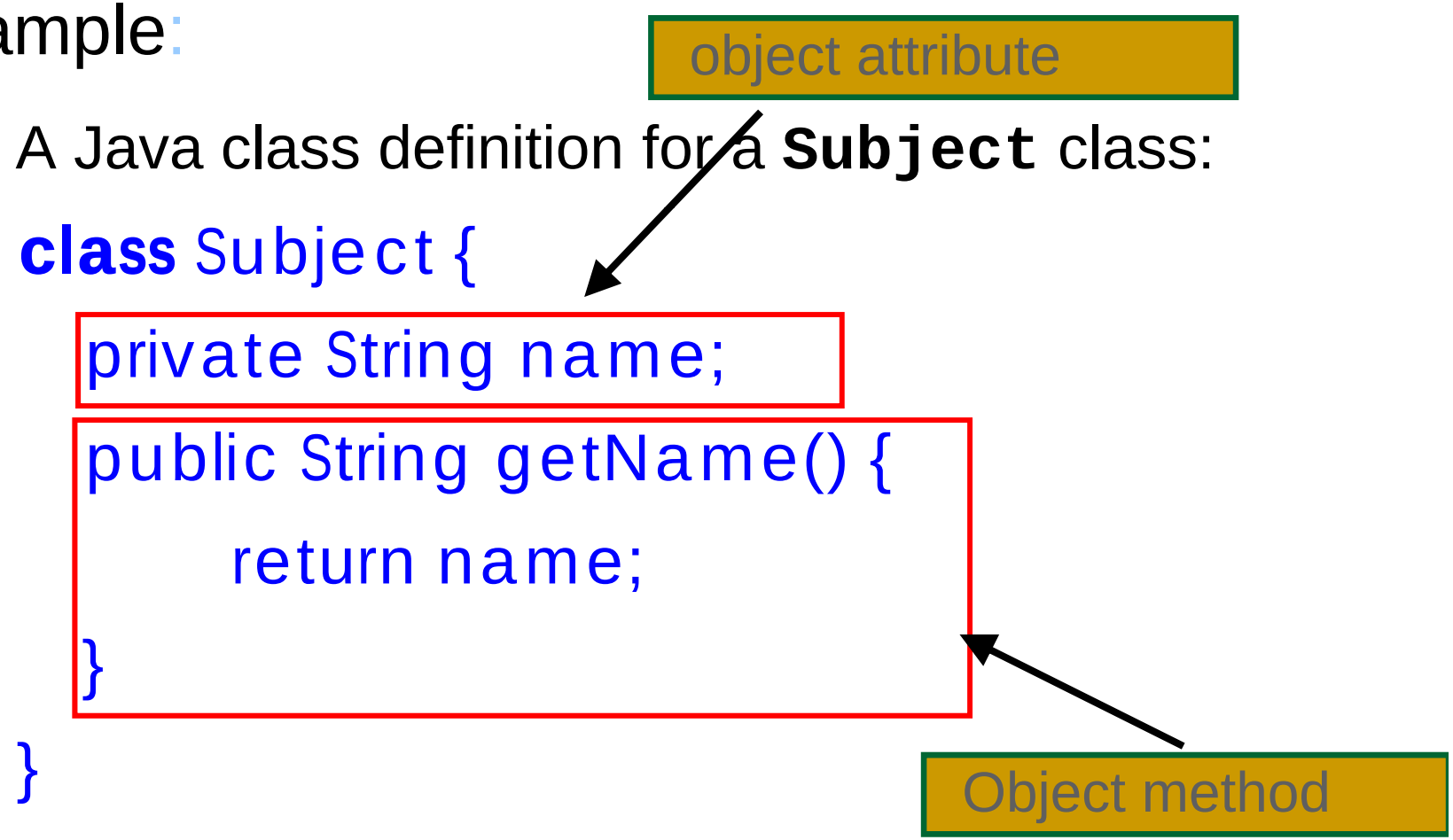
A Java class definition for a **`Subject`** class:

```
class Subject {
   private String name;
    public String getName() {
          return name;
    }
}
```

JAVA CLASS DEFINITION

Example:

A Java class definition for a **`Subject`** class:



DEFINING OBJECT ATTRIBUTES

- Object attributes are defined as instance variables in Java. Each object will have its own instance variable for each attribute.
- By convention, attribute names should begin with lowercase letters.
- A good object-oriented design practice is that object attributes should be hidden.

DEFINING OBJECT ATTRIBUTES

- To define an object attribute in a class definition:

```
private <attribute_type>
        <attribute_name>;
```

- This defines an instance variable with the specified name and type.
- The **`private`** keyword makes the variable privately accessible; in other words, the variable is hidden.

DEFINING OBJECT ATTRIBUTES

Example:

Suppose we wish to define Course objects with the following attributes:

- name of the course
- code of the course
- number of students who registered for the course
- name of the lecturer who handles the course

DEFINING OBJECT ATTRIBUTES

Class definition for the Course class:

```
class Course {
    private String name;
    private String code;
    private int numStudents;
    private String lecturer;
}
```

DEFINING OBJECT ATTRIBUTES

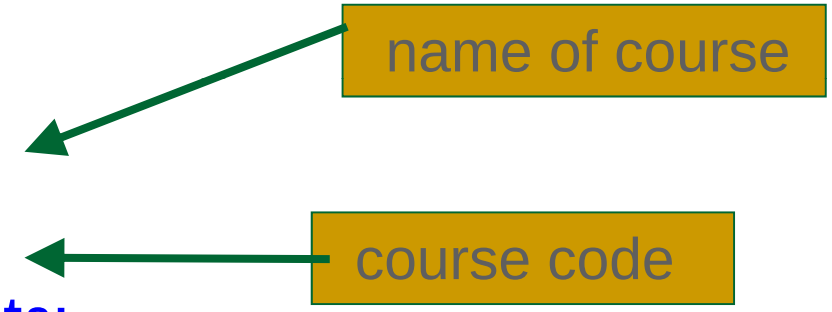
Class definition for the Course class:

```
class Course {
    private String name;
    private String code;
    private int numStudents;
    private String lecturer;
}
```

DEFINING OBJECT ATTRIBUTES

Class definition for the Course class:

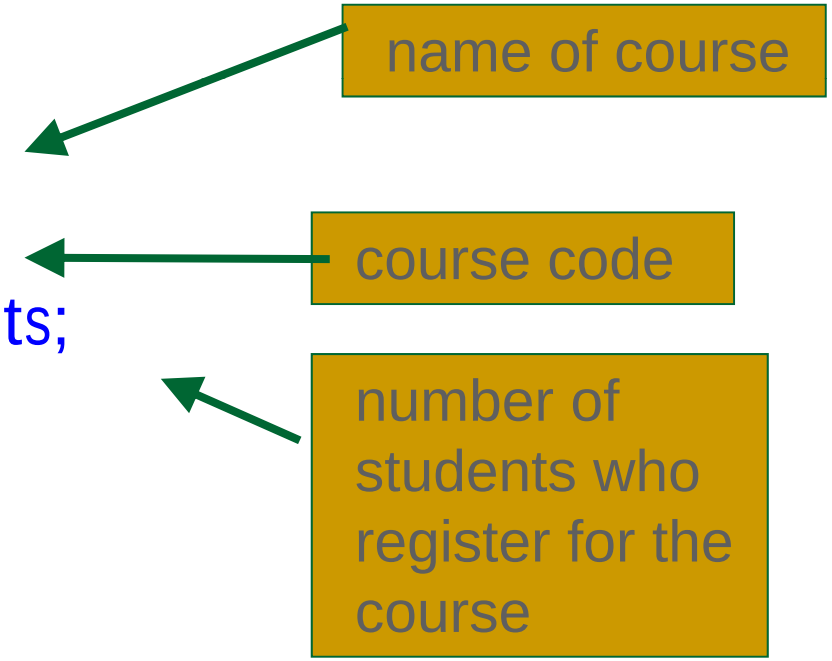
```
class Course {
    private String name;
    private String code;
    private int numStudents;
    private String lecturer;
}
```



DEFINING OBJECT ATTRIBUTES

Class definition for the Course class:

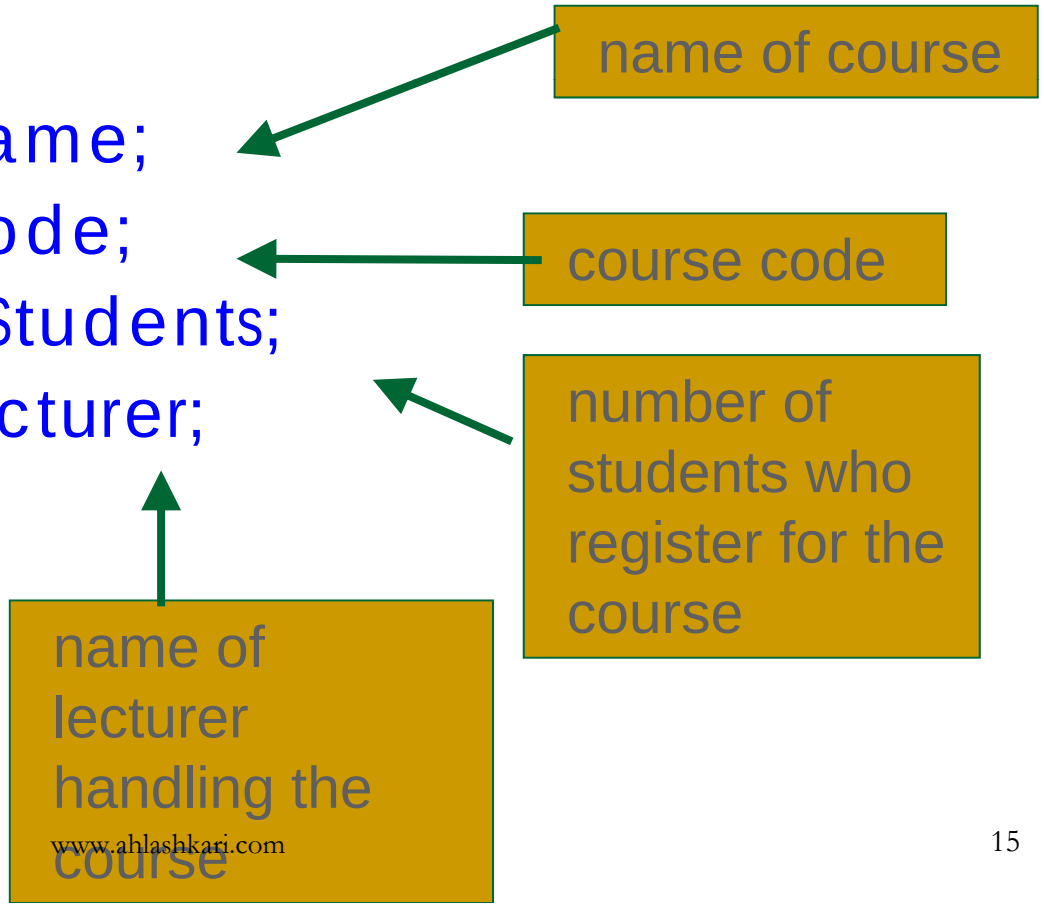
```
class Course {
    private String name;
    private String code;
    private int numStudents;
    private String lecturer;
}
```



DEFINING OBJECT ATTRIBUTES

Class definition for the Course class:

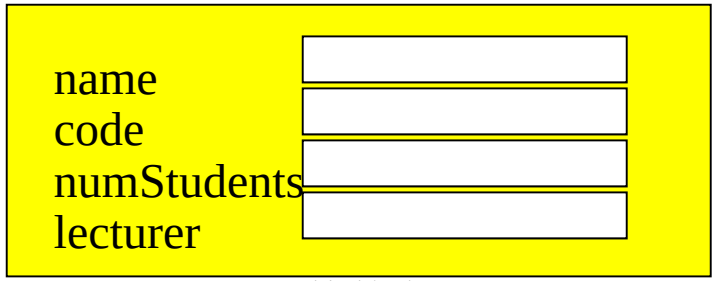
```
class Course {
    private String name;
    private String code;
    private int numStudents;
    private String lecturer;
}
```



DEFINING OBJECT ATTRIBUTES

- Each Course object created will be allocated some memory space for each instance variable declared in the Course class.

new Course( )



DEFINING OBJECT METHODS

Syntax for defining object methods:

```
public <return_type>
  <method_name>(<parameters>)
{
    <method_body>
}
```

DEFINING OBJECT METHODS

- The **public** keyword means that the method can be “called” from the outside of the class definition.
- This means that the method will be part of the **class’s interface**.
- By convention, method names should begin with **lowercase letters.**

WRITING THE METHOD BODY

- Objects exhibit their behaviour when they execute their methods.
- The behaviour of an object can cause
  - its state to change
  - messages to be sent to other objects (including to itself)
  - data to be returned to its client

WRITING THE METHOD BODY

- An object usually needs certain data when executing a method.
- For example, a Rectangle object requires the width and height when executing a method to calculate its area.

WRITING THE METHOD BODY

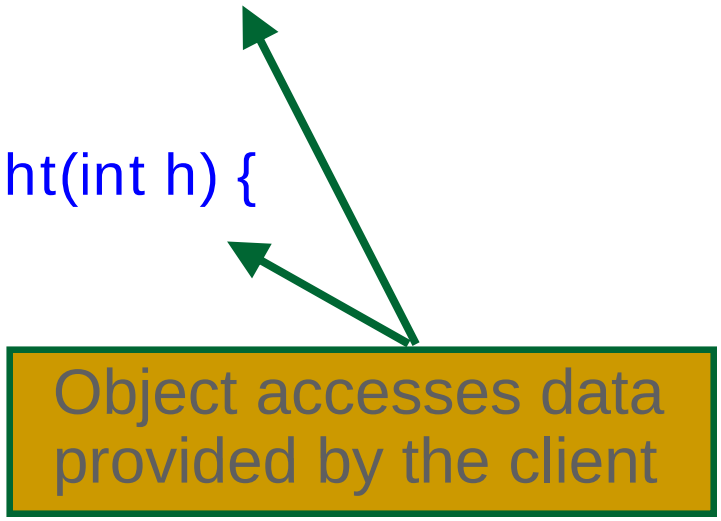
- Several ways an object obtains data:
  - access its own attributes
  - data provided by the client (parameters of the method)
  - request data from other objects by sending messages to them

WRITING THE METHOD BODY

```
class Rectangle {

    private int width;
    private int height;

    public void setWidth(int w) {
        width = w;
    }
    public void setHeight(int h) {
        height = h;
    }
```

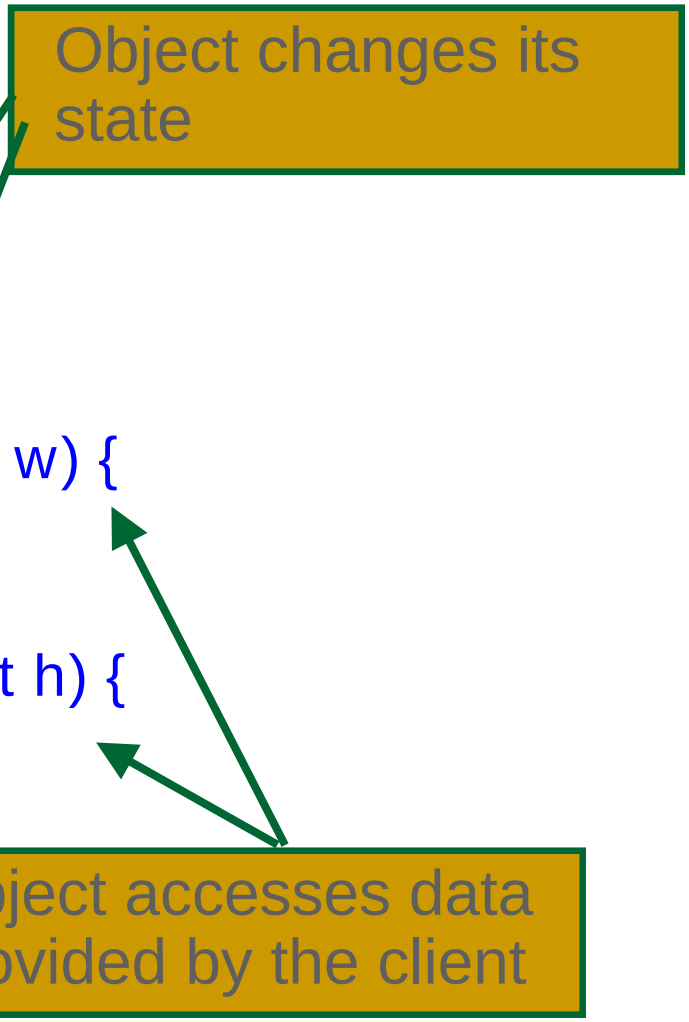
WRITING THE METHOD BODY

```
class Rectangle {

    private int width;
    private int height;

    public void setWidth(int w) {
        width = w;
    }
    public void setHeight(int h) {
        height = h;
    }
```



WRITING THE METHOD BODY

```
class Rectangle {

    private int width;
    private int height;

    public void setWidth(int w) {
        width = w;
    }
    public void setHeight(int h) {
        height = h;
    }
```

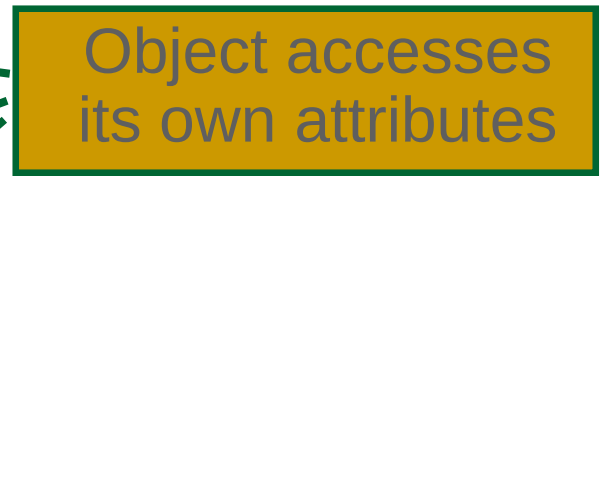


WRITING THE METHOD BODY

```
    public int getWidth() {
       return width;
    }
    public int getHeight() {
       return height;
    }
    public int getArea() {
       return width * height;
    }
    public void display() {
       (new Painter()).displayRectangle(width, height);
    }
}
```

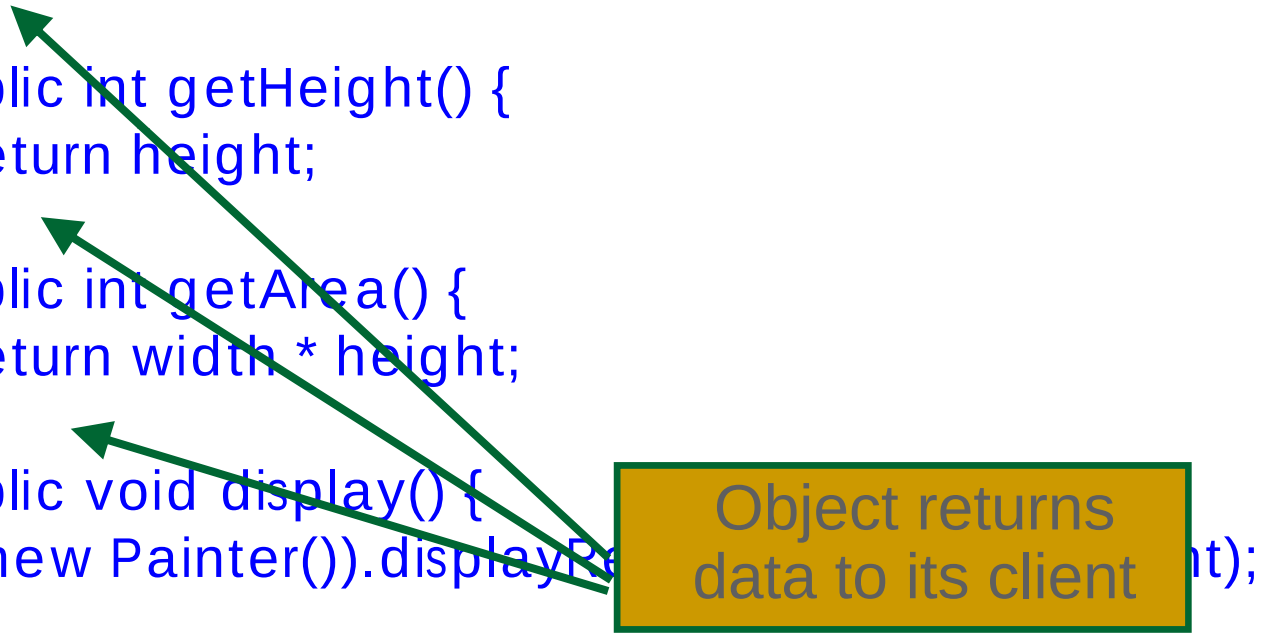
WRITING THE METHOD BODY

```
    public int getWidth() {
        return width;
    }
    public int getHeight() {
        return height;
    }
    public int getArea() {
        return width * height;
    }
    public void display() {
        (new Painter()).displayRectangle(width, height);
    }
}
```



WRITING THE METHOD BODY

```
    public int getWidth() {
       return width;
    }
    public int getHeight() {
       return height;
    }
    public int getArea() {
       return width * height;
    }
    public void display() {
       (new Painter()).displayRe                    nt);
    }
}
```



WRITING THE METHOD BODY

```
    public int getWidth() {
       return width;
    }
    public int getHeight() {
       return height;
    }
    public int getArea() {
       return width * height;
    }
    public void display() {
       (new Painter()).displayRectangle(width, height);
    }
}
```

Object sends message to another object

CLIENT PROGRAM

```
class MyApplication {
    public static void main(String[] args) {
            Rectangle r1 = new Rectangle();
            Rectangle r2 = new Rectangle();
            r1.setWidth(10);  r1.setHeight(20);
            r2.setWidth(5); r2.setHeight(30);

            System.out.println("First rectangle: ");
            System.out.println("Width: "+r1.getWidth());
            System.out.println("Height: "+r1.getHeight());
            System.out.println("Area: "+r1.getArea());
            r2.display();
    }}
```

Questions

THANK YOU

Arash Habibi Lashkari

PHD. Candidate of UTM

Kuala Lumpur, Malaysia

Feb, 2010

THE END