جامعة الجلالة
GALALA UNIVERSITY

Course: Object oriented

Grade: Firth year

Assistant Lecturer : Yasmin Alsakar at Faculty of Computer & Information Sciences - Mansoura University.

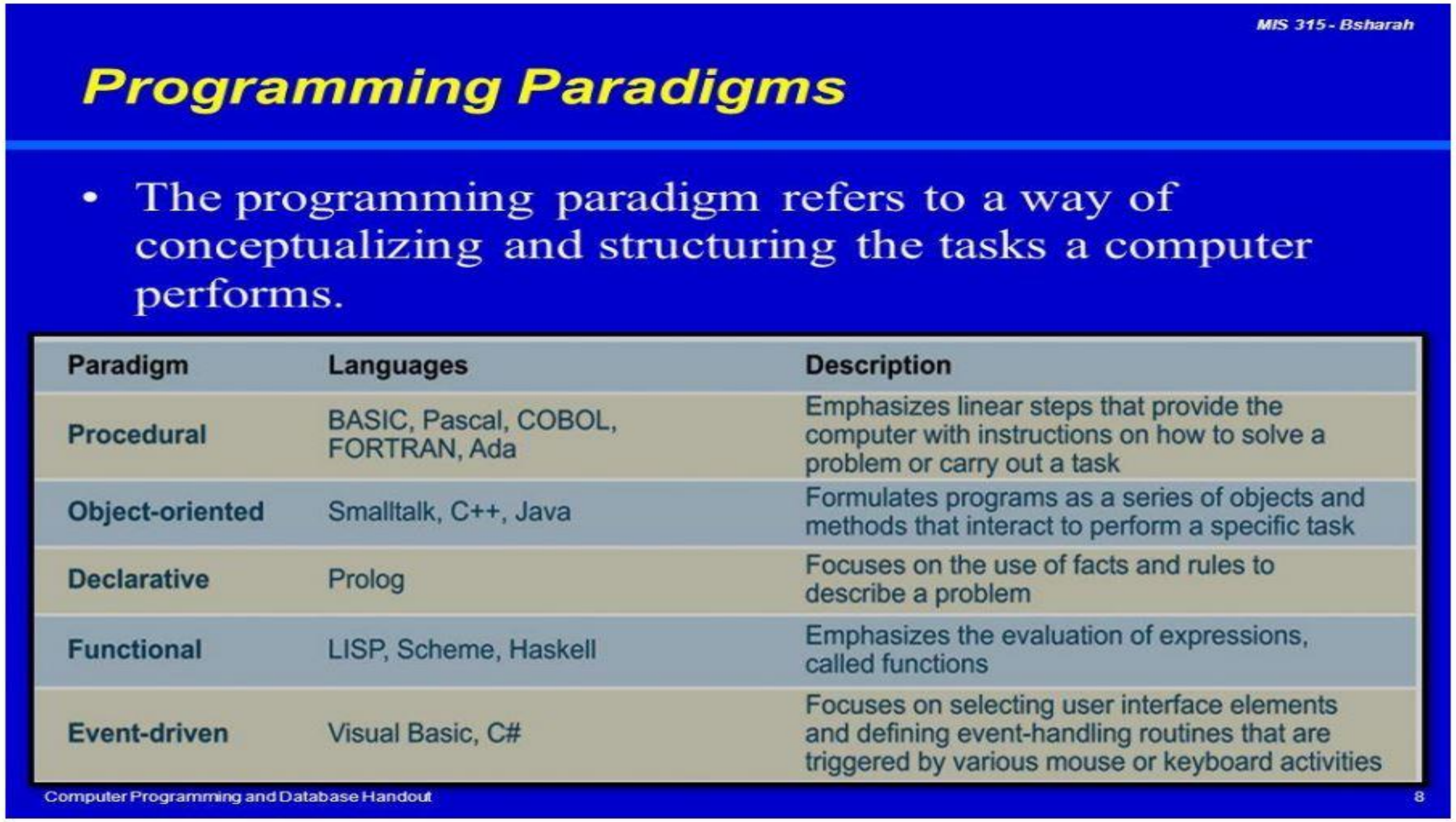
OUTLINE

- What Is Object oriented programming?
- Example of Programming Paradigms
- Object Is comprised Of ?
- What is Class ? Why we need It ?
- Access Modifiers
- Data Hiding
- Create a Class
- Create an Object

What Is Object oriented programming?

- Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects"
- A programming paradigm : is a style of programming, a way of thinking about software construction.
- A programming paradigm does not refer to a specific language but rather to a way to build a program or a methodology to apply.
- Some languages make it easy to write in some paradigms but not others.
- Some Programming Languages allow the programmer to apply more than one Paradigm.

Example of Programming Paradigms



MIS 315 - Bsharah

Programming Paradigms

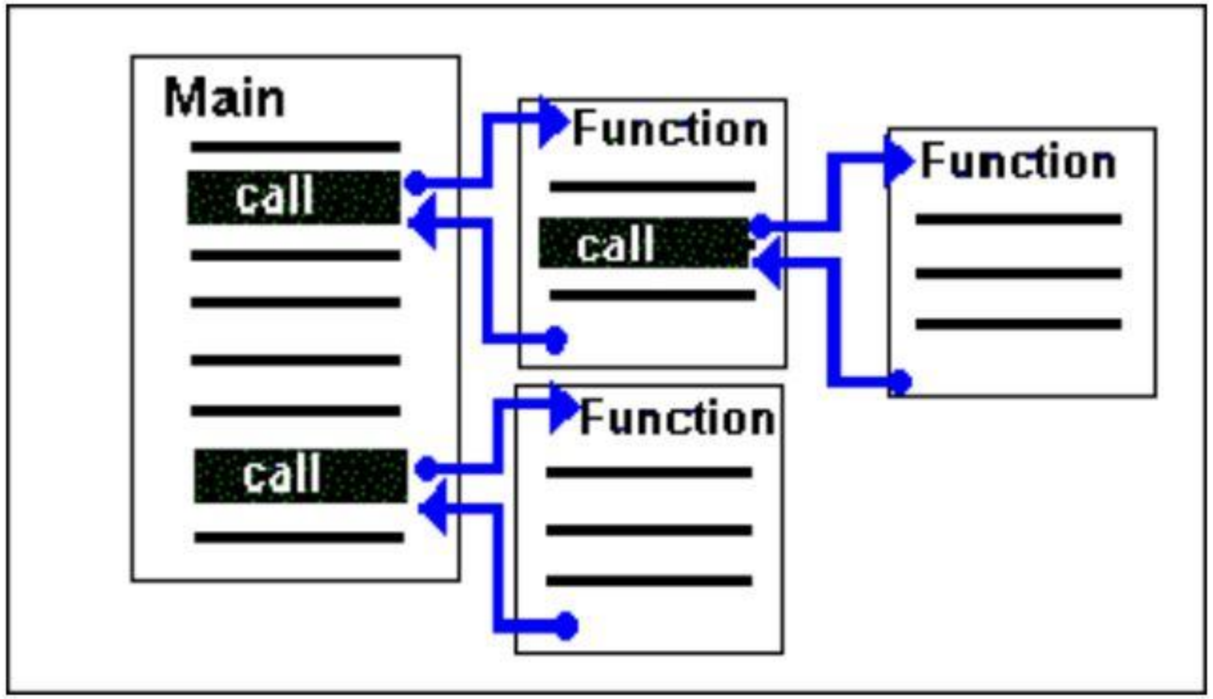
- The programming paradigm refers to a way of conceptualizing and structuring the tasks a computer performs.

Paradigm	Languages	Description
Procedural	BASIC, Pascal, COBOL, FORTRAN, Ada	Emphasizes linear steps that provide the computer with instructions on how to solve a problem or carry out a task
Object-oriented	Smalltalk, C++, Java	Formulates programs as a series of objects and methods that interact to perform a specific task
Declarative	Prolog	Focuses on the use of facts and rules to describe a problem
Functional	LISP, Scheme, Haskell	Emphasizes the evaluation of expressions, called functions
Event-driven	Visual Basic, C#	Focuses on selecting user interface elements and defining event-handling routines that are triggered by various mouse or keyboard activities

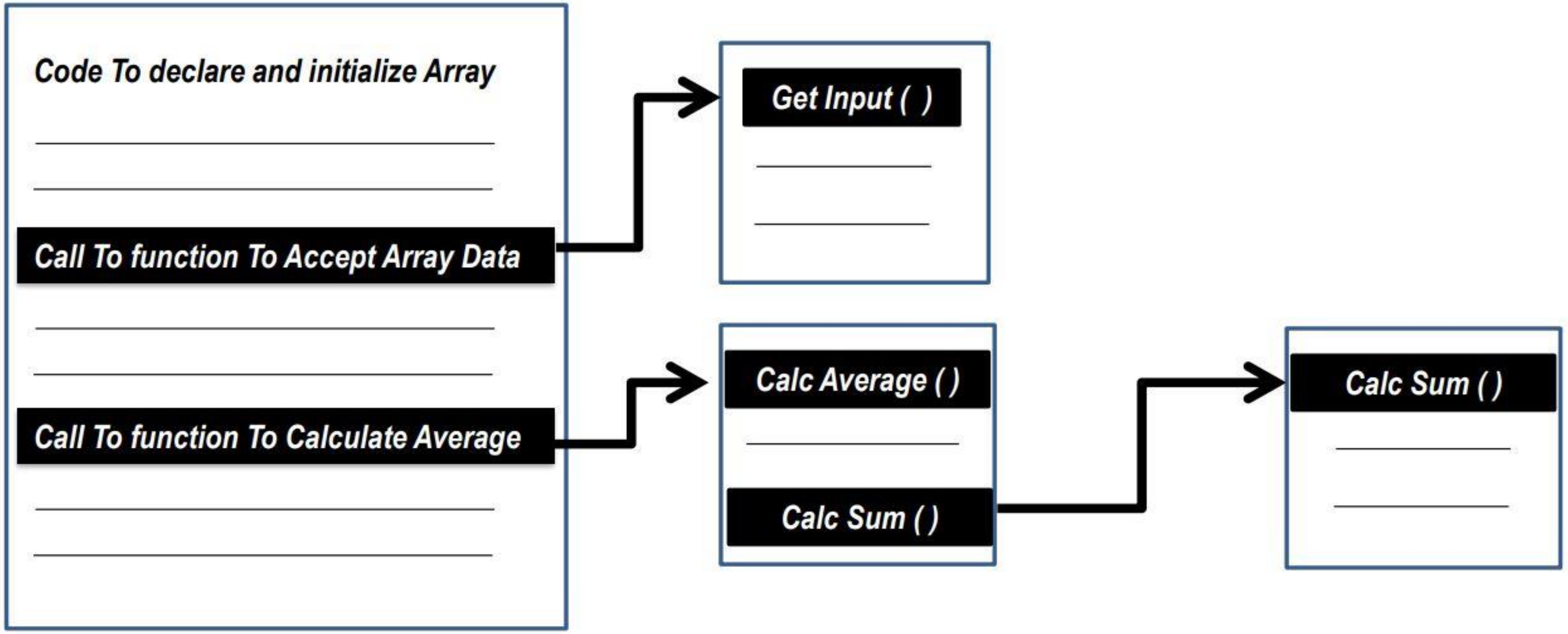
Computer Programming and Database Handout 8

Example of Previous Programming Paradigm

- Procedural Programming: (PP), also known as inline programming takes a top-down approach. It is about writing a list of instructions to tell the computer what to do step by step. It relies on procedures or routines.



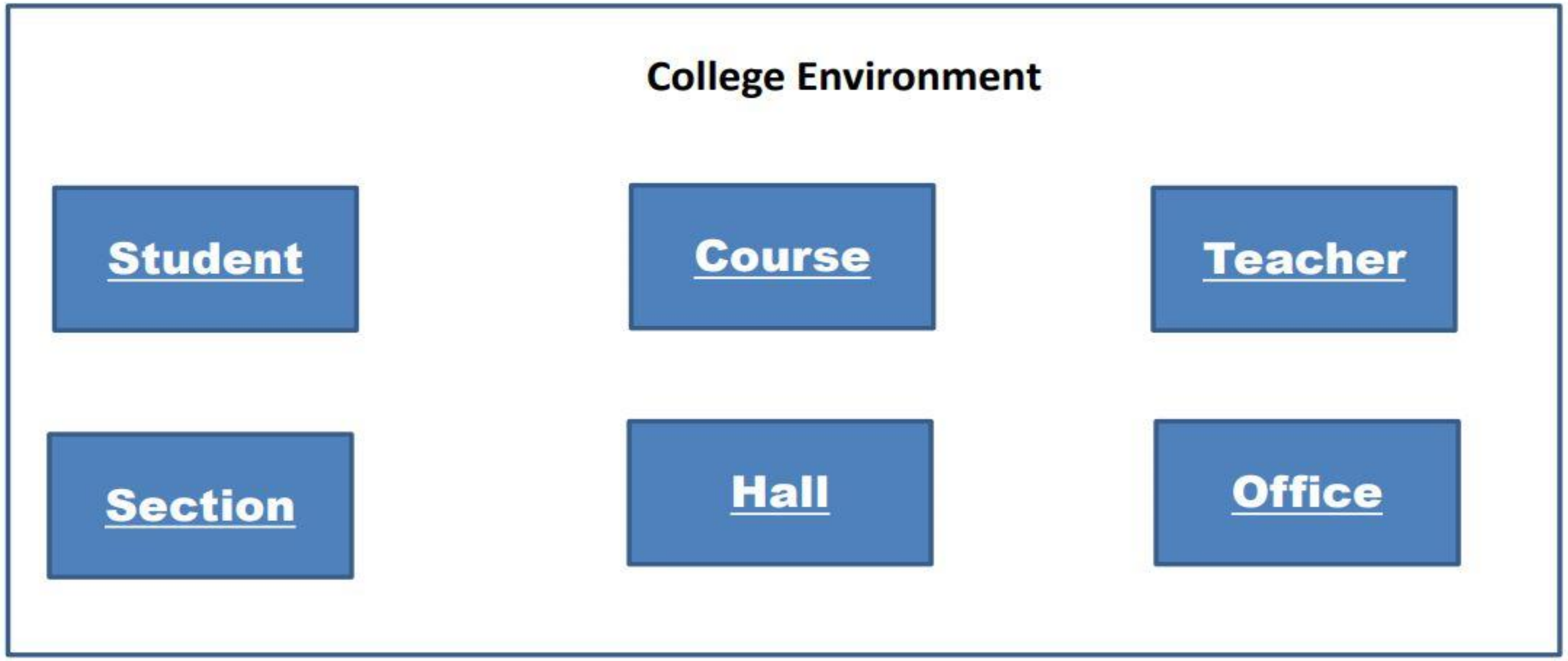
Procedural Programming Example : Program to Calculate Average of Array Items


Code To declare and initialize Array
Call To function To Accept Array Data
Call To function To Calculate Average
Get Input ( )
Calc Average ( )
Calc Sum ( )
Calc Sum ( )

- Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects“
- Object : is a thing (Tangible – Intangible)

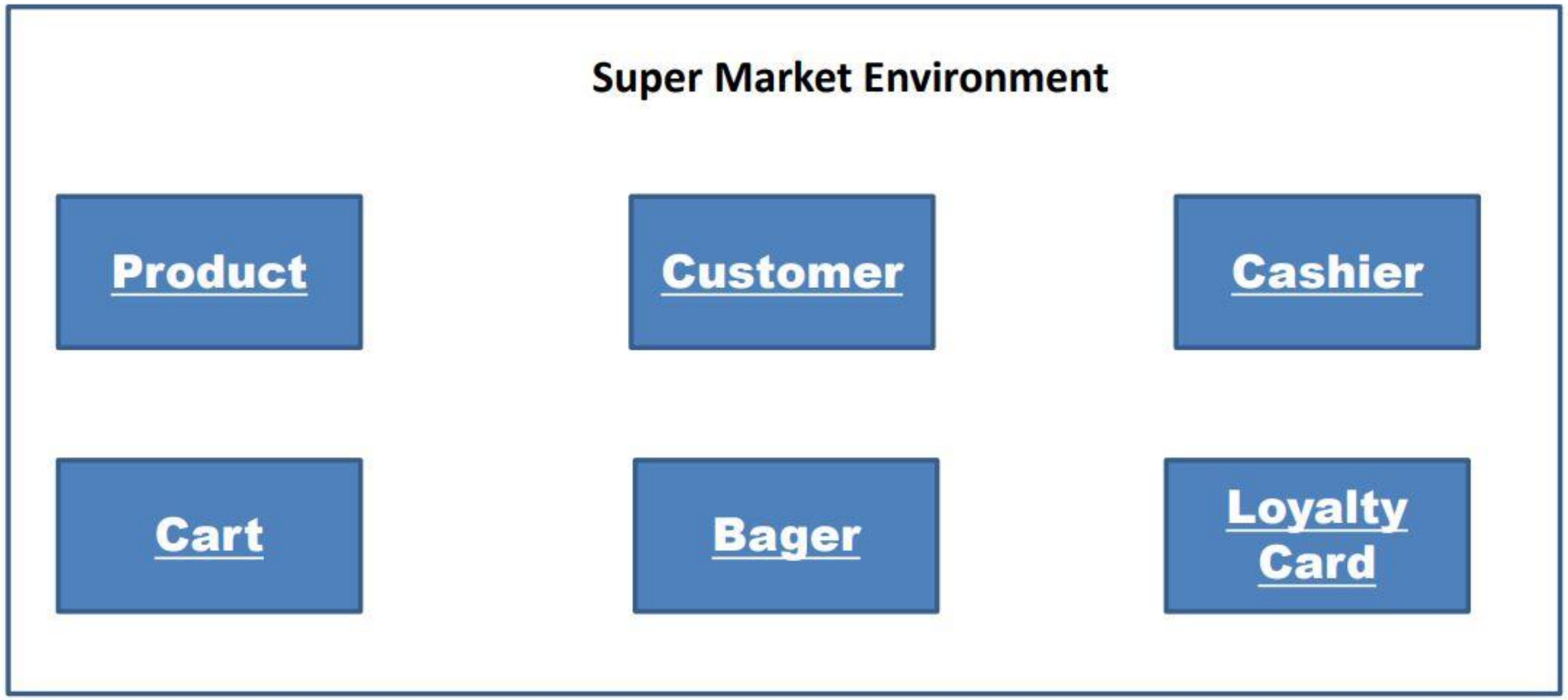
Example (1) for oop:

Objects in College Management Program

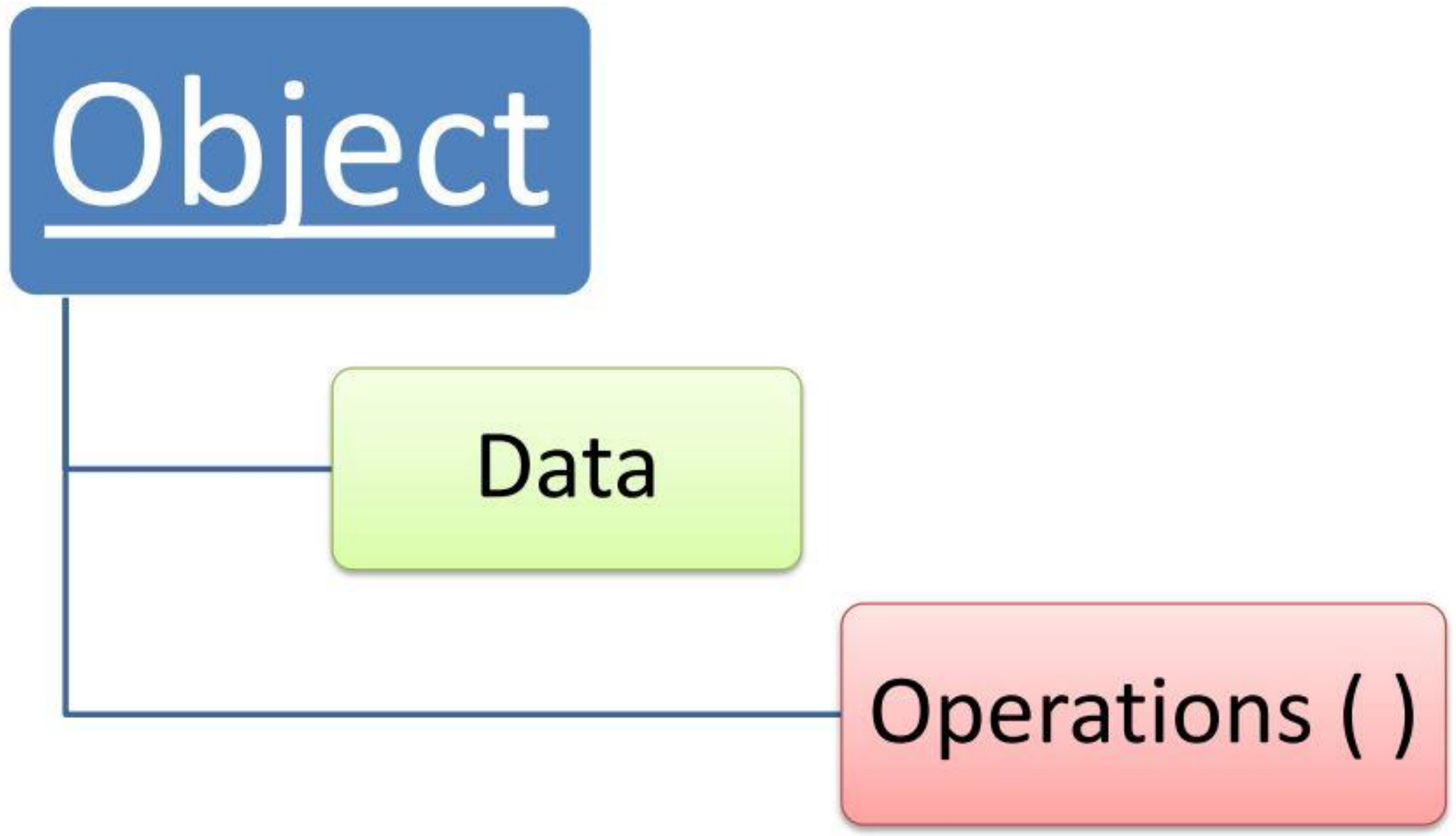


Example (2) for oop:

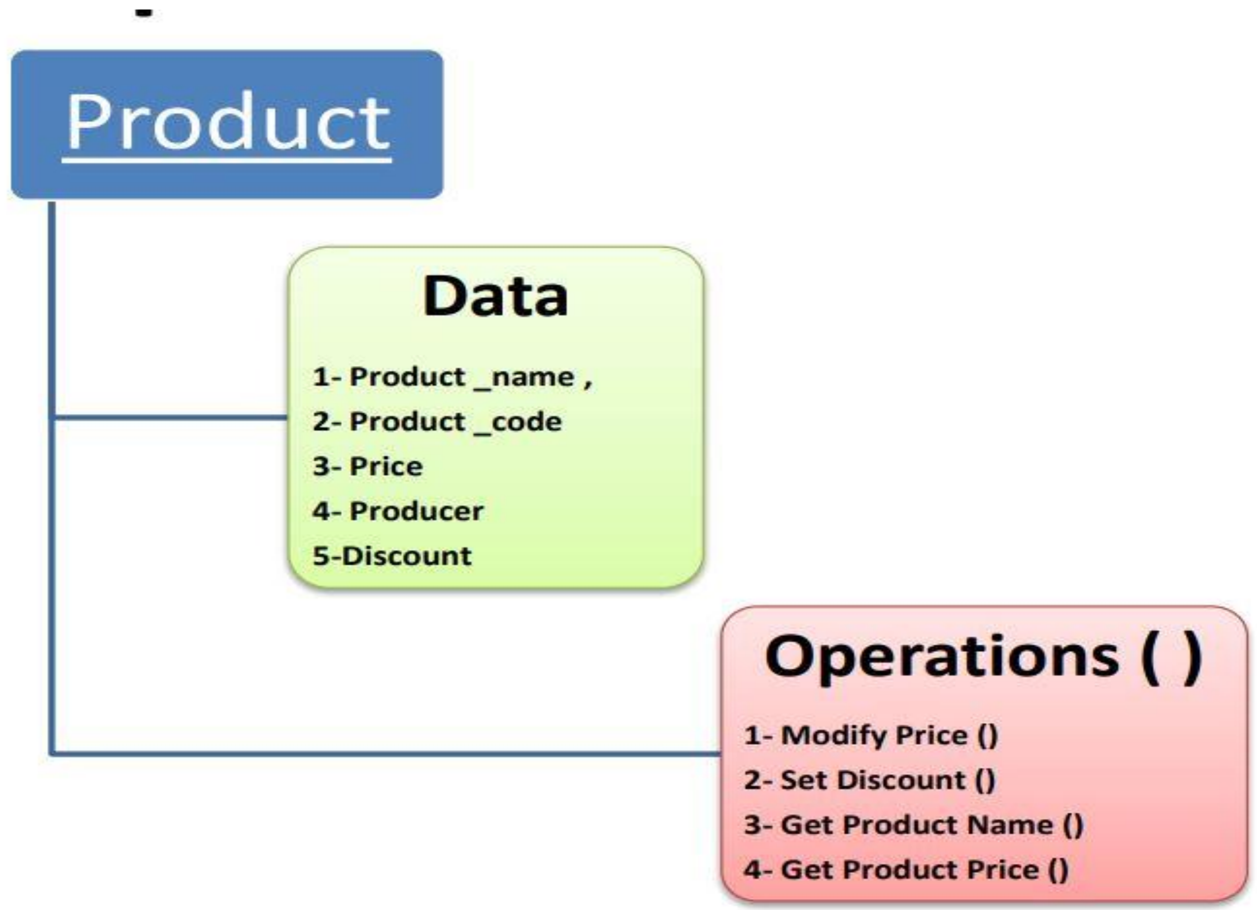
Objects in Super market Program

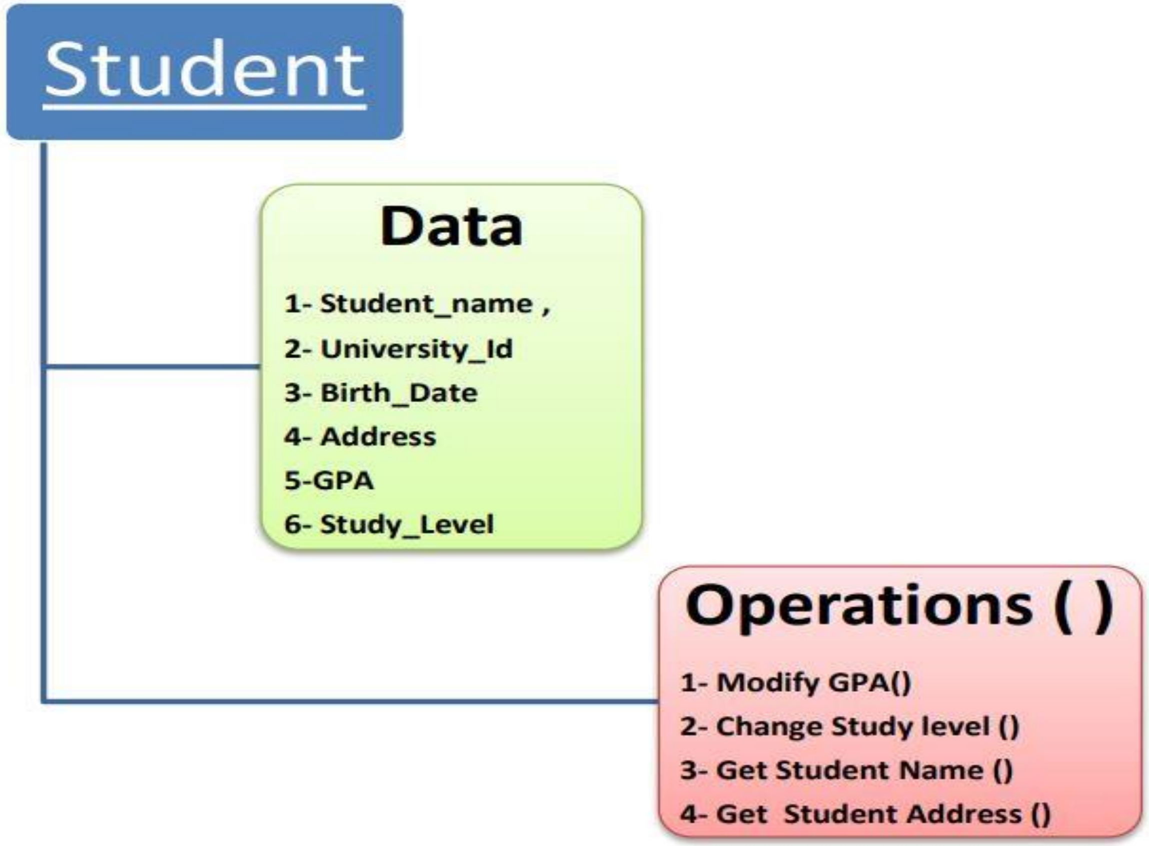


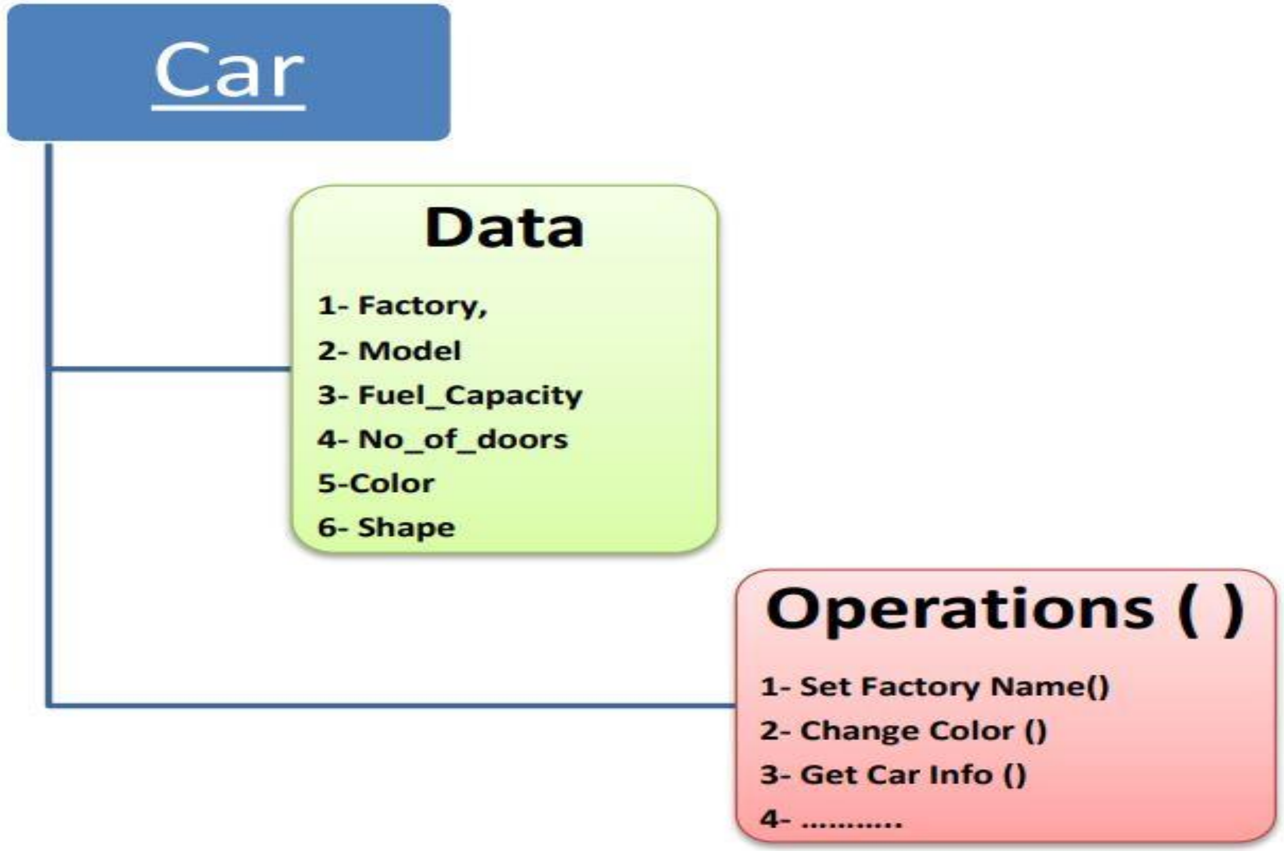
Object Is comprised Of ?



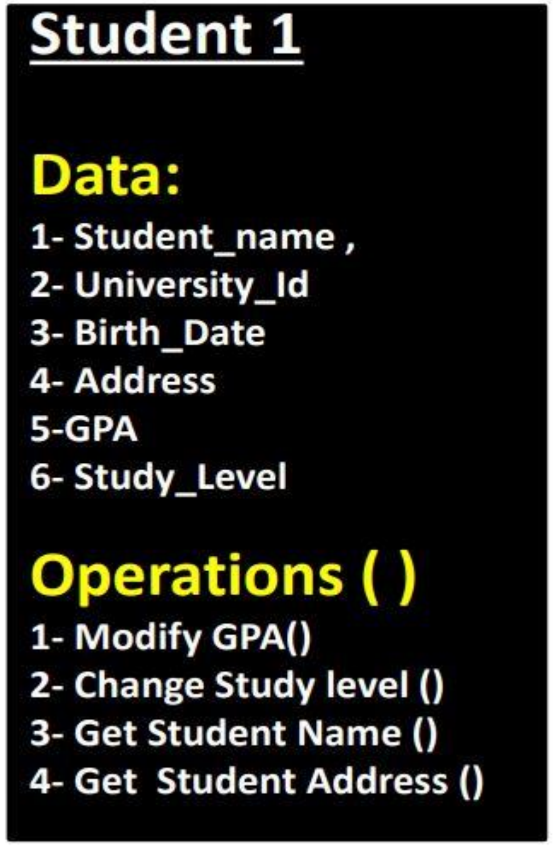
- For example:


Product
Data
1- Product _name ,
2- Product _code
3- Price
4- Producer
5-Discount
Operations ( )
1- Modify Price ()
2- Set Discount ()
3- Get Product Name ()
4- Get Product Price ()


Student
Data
1- Student_name ,
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level
Operations ( )
1- Modify GPA()
2- Change Study level ()
3- Get Student Name ()
4- Get Student Address ()


Car
Data
1- Factory,
2- Model
3- Fuel_Capacity
4- No_of_doors
5-Color
6- Shape
Operations ( )
1- Set Factory Name()
2- Change Color ()
3- Get Car Info ()
4- ....

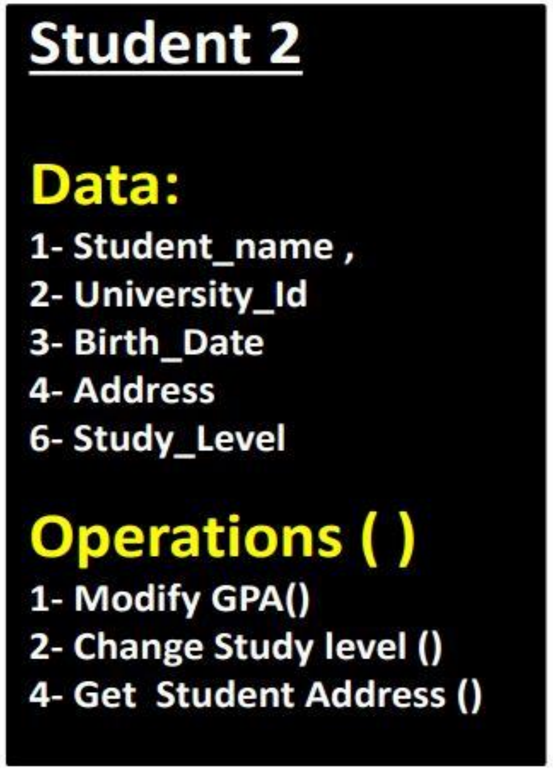
What is Class ? Why we need It ?



Student 1

Data:
1- Student_name ,
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level

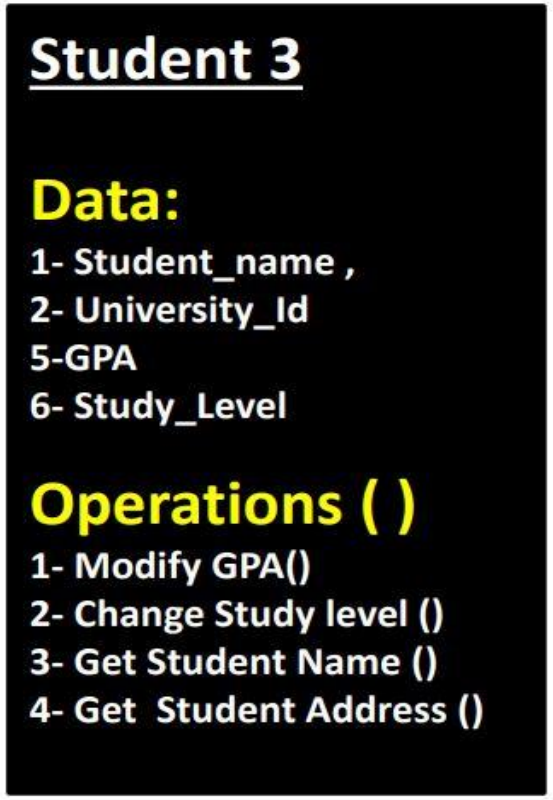
Operations ( )
1- Modify GPA()
2- Change Study level ()
3- Get Student Name ()
4- Get Student Address ()



Student 2

Data:
1- Student_name ,
2- University_Id
3- Birth_Date
4- Address
6- Study_Level

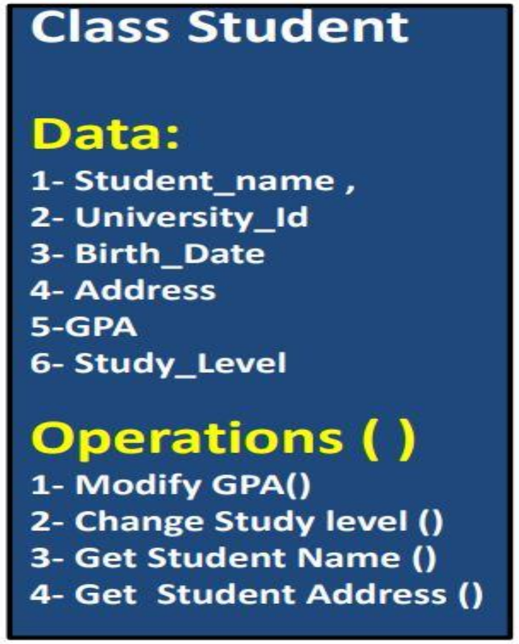
Operations ( )
1- Modify GPA()
2- Change Study level ()
4- Get Student Address ()



Student 3

Data:
1- Student_name ,
2- University_Id
5-GPA
6- Study_Level

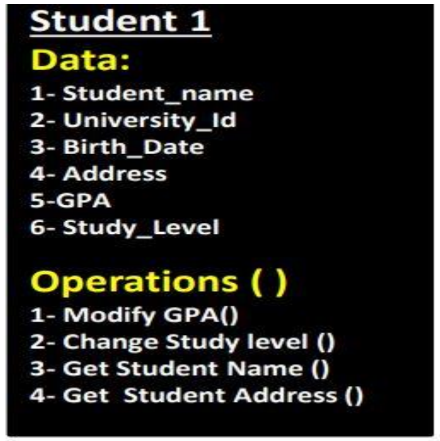
Operations ( )
1- Modify GPA()
2- Change Study level ()
3- Get Student Name ()
4- Get Student Address ()



Class Student

Data:
1- Student_name ,
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level

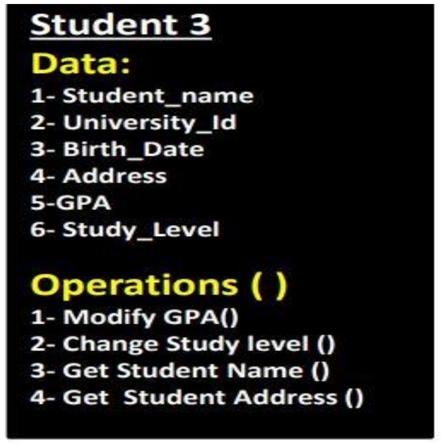
Operations ( )
1- Modify GPA()
2- Change Study level ()
3- Get Student Name ()
4- Get Student Address ()



Student 1

Data:
1- Student_name
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level

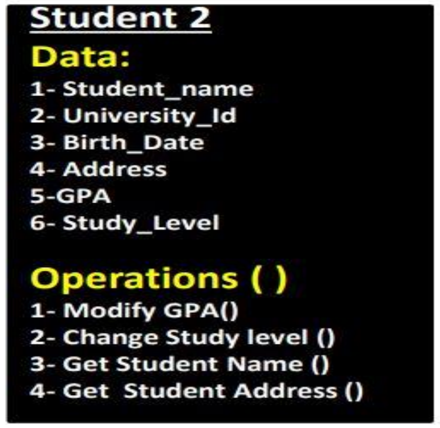
Operations ( )
1- Modify GPA()
2- Change Study level ()
3- Get Student Name ()
4- Get Student Address ()



Student 3

Data:
1- Student_name
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level

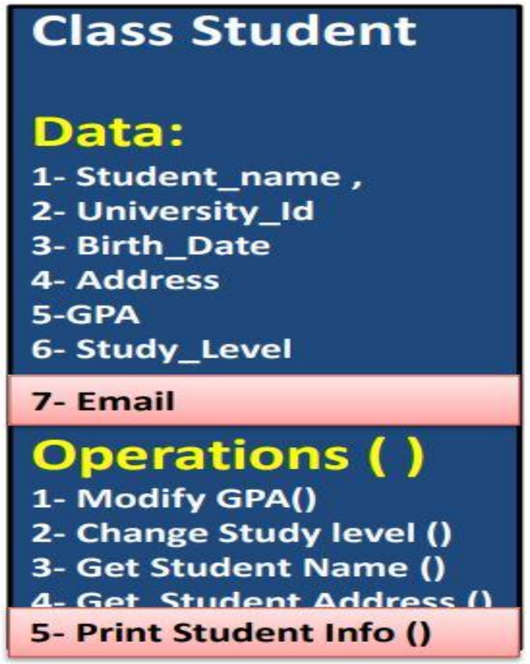
Operations ( )
1- Modify GPA()
2- Change Study level ()
3- Get Student Name ()
4- Get Student Address ()



Student 2

Data:
1- Student_name
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level

Operations ( )
1- Modify GPA()
2- Change Study level ()
3- Get Student Name ()
4- Get Student Address ()



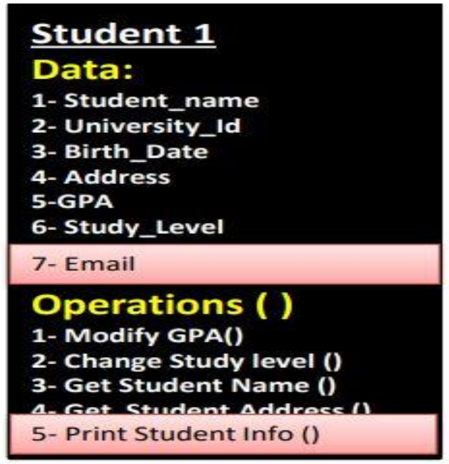
Class Student

Data:

1- Student_name ,
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level
7- Email

Operations ( )

1- Modify GPA()
2- Change Study level ()
3- Get Student Name ()
4- Get Student Address ()
5- Print Student Info ()



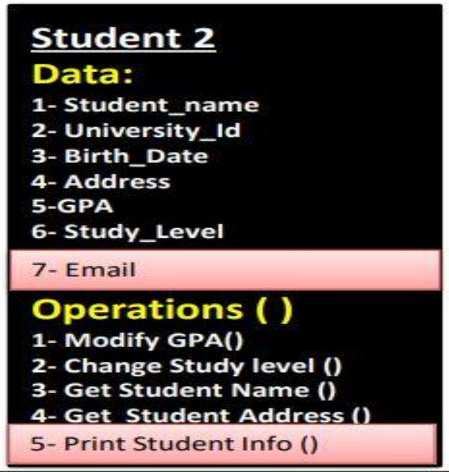
Student 1

Data:

1- Student_name
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level
7- Email

Operations ( )

1- Modify GPA()
2- Change Study level ()
3- Get Student Name ()
4- Get Student Address ()
5- Print Student Info ()



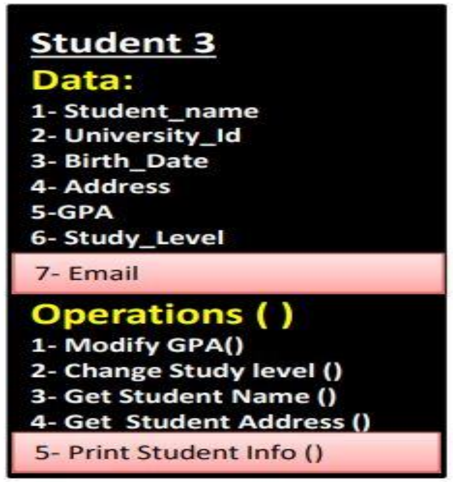
Student 2

Data:

1- Student_name
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level
7- Email

Operations ( )

1- Modify GPA()
2- Change Study level ()
3- Get Student Name ()
4- Get Student Address ()
5- Print Student Info ()



Student 3

Data:

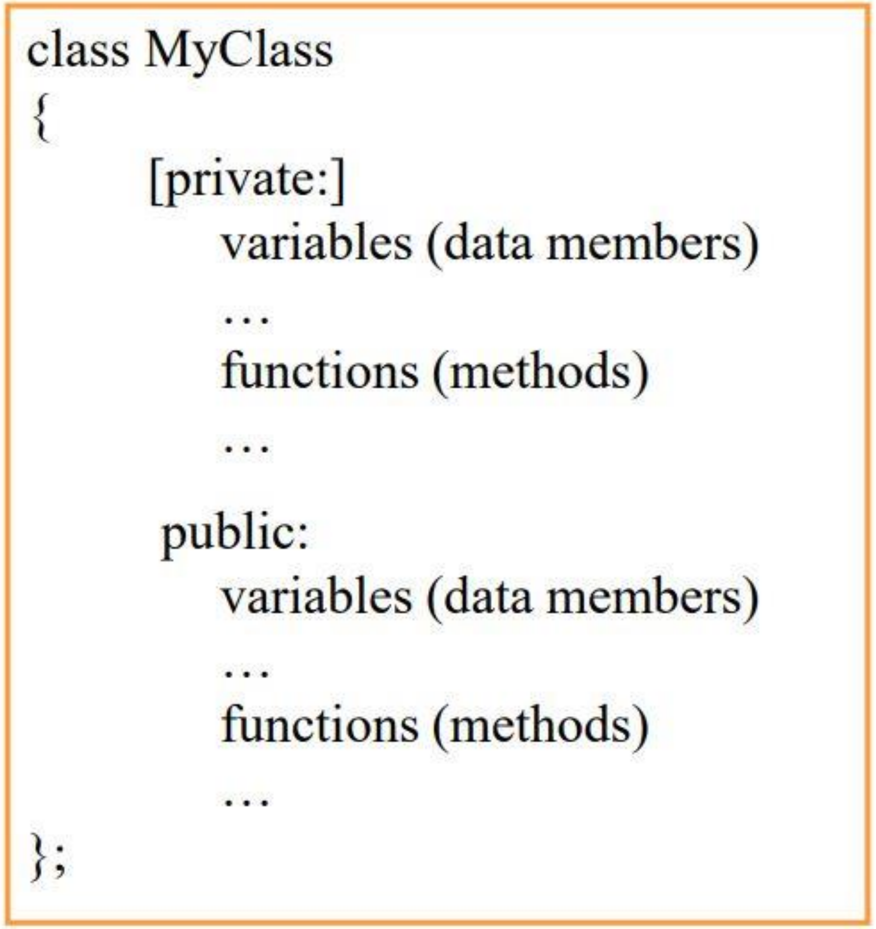
1- Student_name
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level
7- Email

Operations ( )

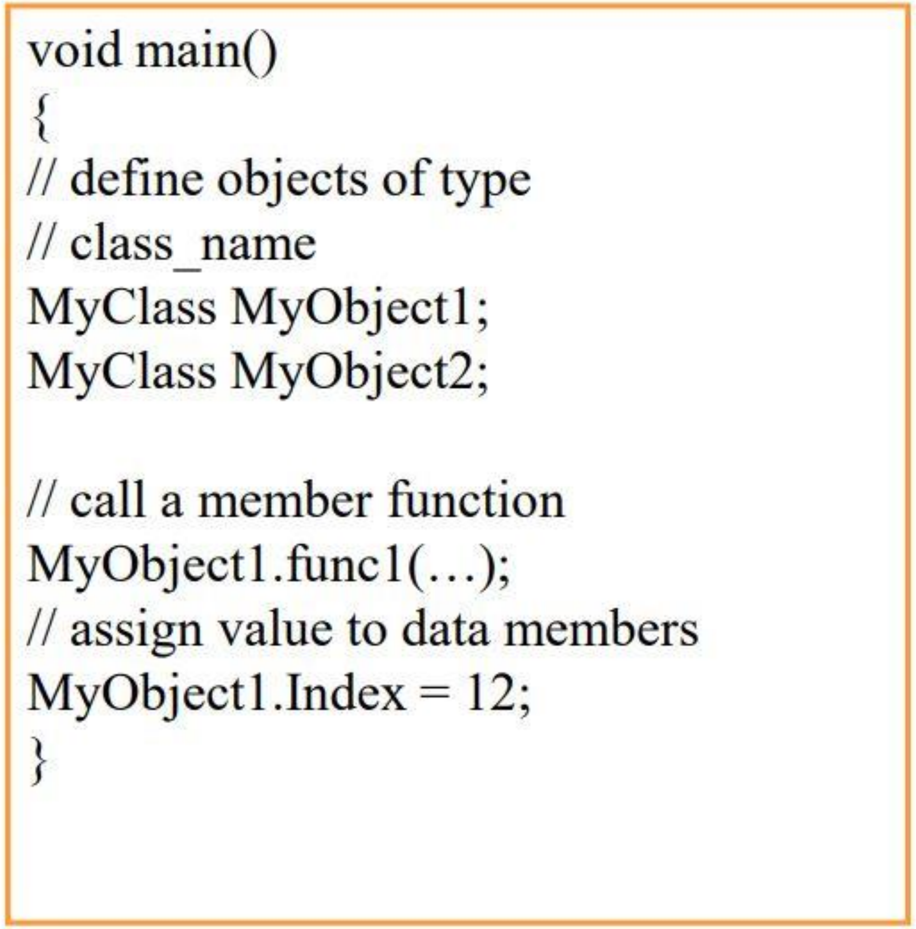
1- Modify GPA()
2- Change Study level ()
3- Get Student Name ()
4- Get Student Address ()
5- Print Student Info ()

- Classes: Where Objects Come From
  - A class is code that describes a particular type of object. It specifies the data that an object can hold (the object's fields), and the actions that an object can perform (the object's methods).
  - You can think of a class as a code "blueprint" that can be used to create a particular type of object.
  - When a program is running, it can use the class to create, in memory, as many objects of a specific type as needed.
  - Each object that is created from a class is called an **instance** of the class.

A class is defined (declared) and used as follows:

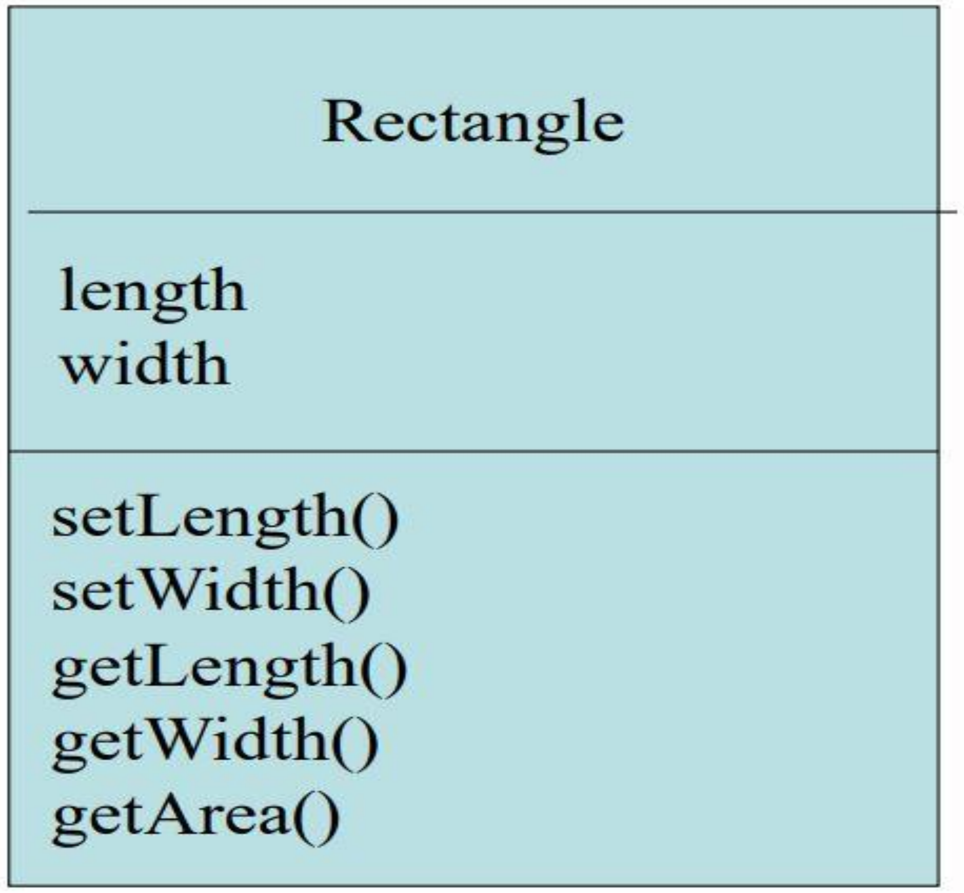


```
class MyClass
{
        [private:]
            variables (data members)
            …
            functions (methods)
            …
         public:
            variables (data members)
            …
            functions (methods)
            …
};
```



```
void main()
{
// define objects of type
// class_name
MyClass MyObject1;
MyClass MyObject2;

// call a member function
MyObject1.func1(…);
// assign value to data members
MyObject1.Index = 12;
}
```

Example for writing the rectangle class:

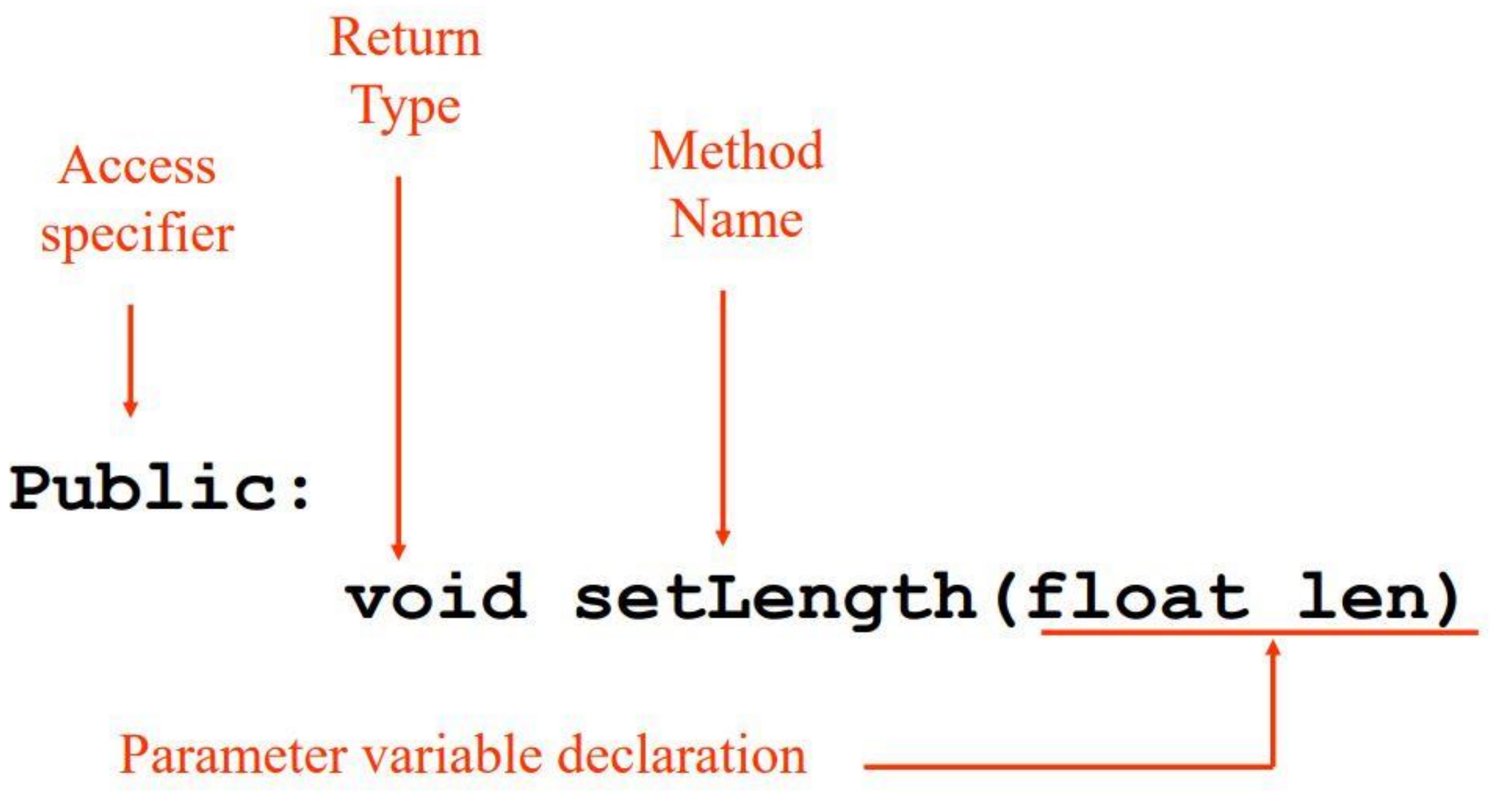


Access Modifiers

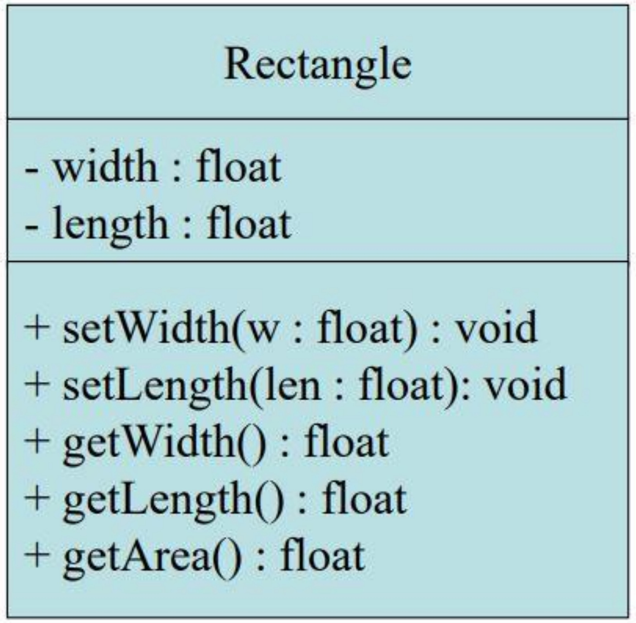
- An access modifier is a C++ keyword that indicates how a field or method can be accessed.
- **public** – When the public access modifier is applied to a class member, the member can be accessed by code inside the class or outside.
- **private** – When the private access modifier is applied to a class member, the member cannot be accessed by code outside the class. The member can be accessed only by methods that are members of the same class.

Data Hiding

- An object hides its internal, private fields from code that is outside the class that the object is an instance of.
- Only the class's methods may directly access and change the object's internal data.
- Code outside the class must use the class's public methods to operate on an object's private fields.
- Data hiding is important because classes are typically used as components in large software systems, involving a team of programmers.
- Data hiding helps enforce the integrity of an object's internal data.

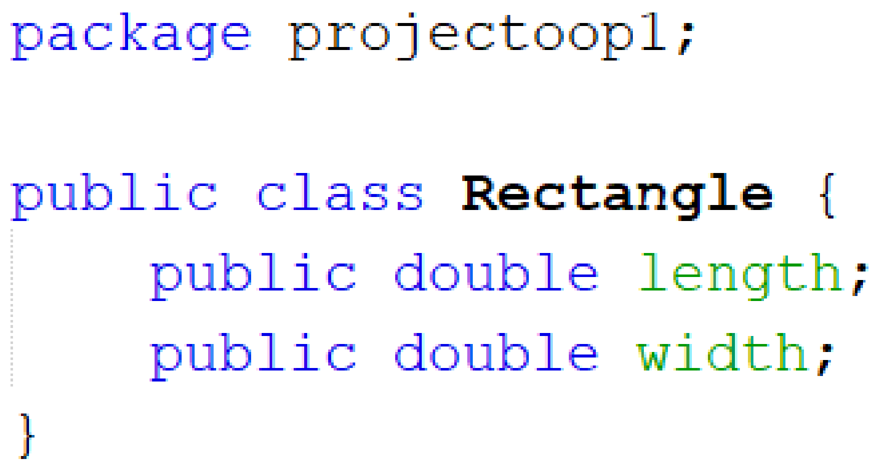


```
Public:
        void setLength(float len)
```



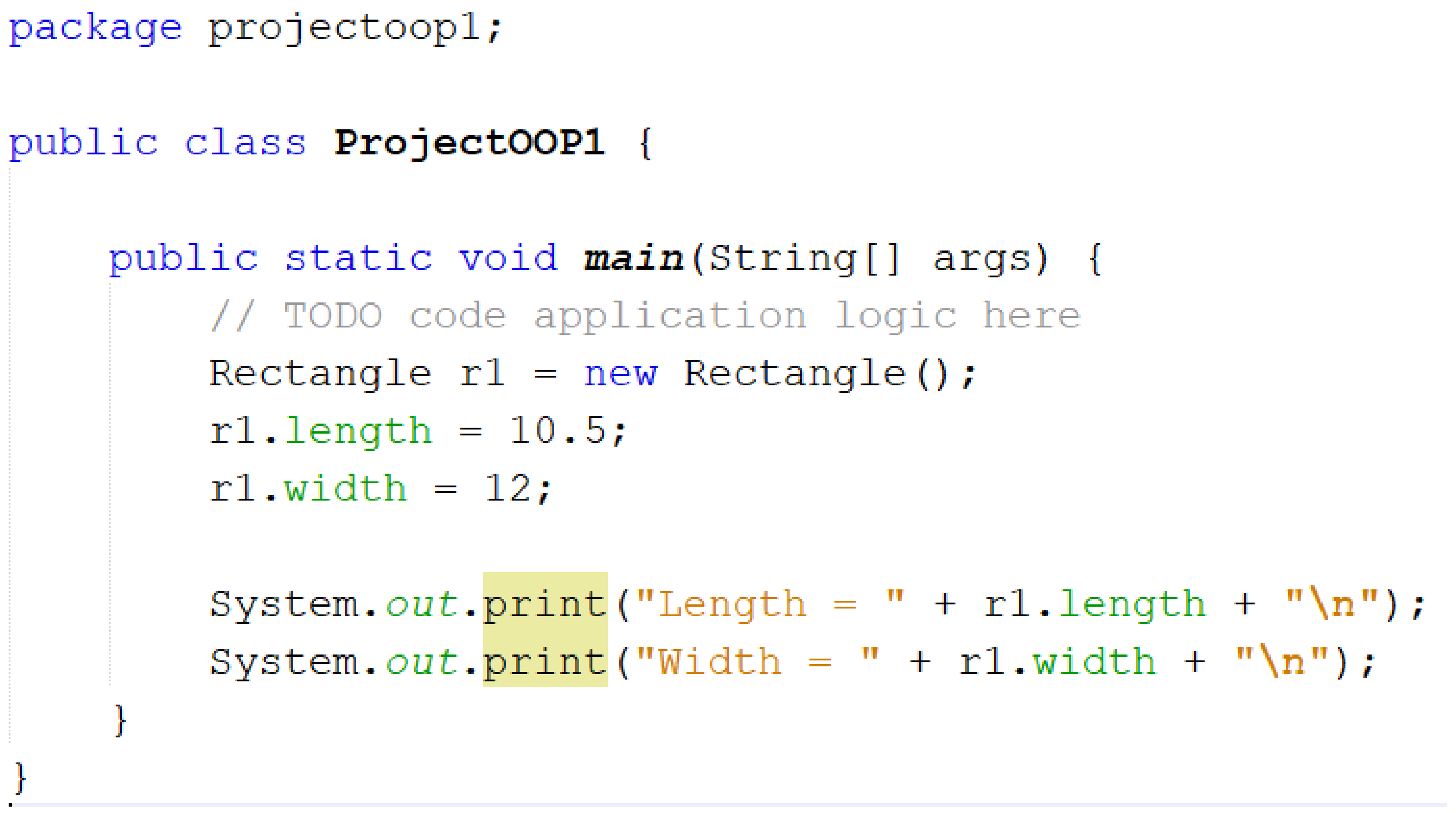
Create a Class

- To create a class, use the class keyword:



```
package projectoop1;

public class Rectangle {
    public double length;
    public double width;
}
```

Create an Object

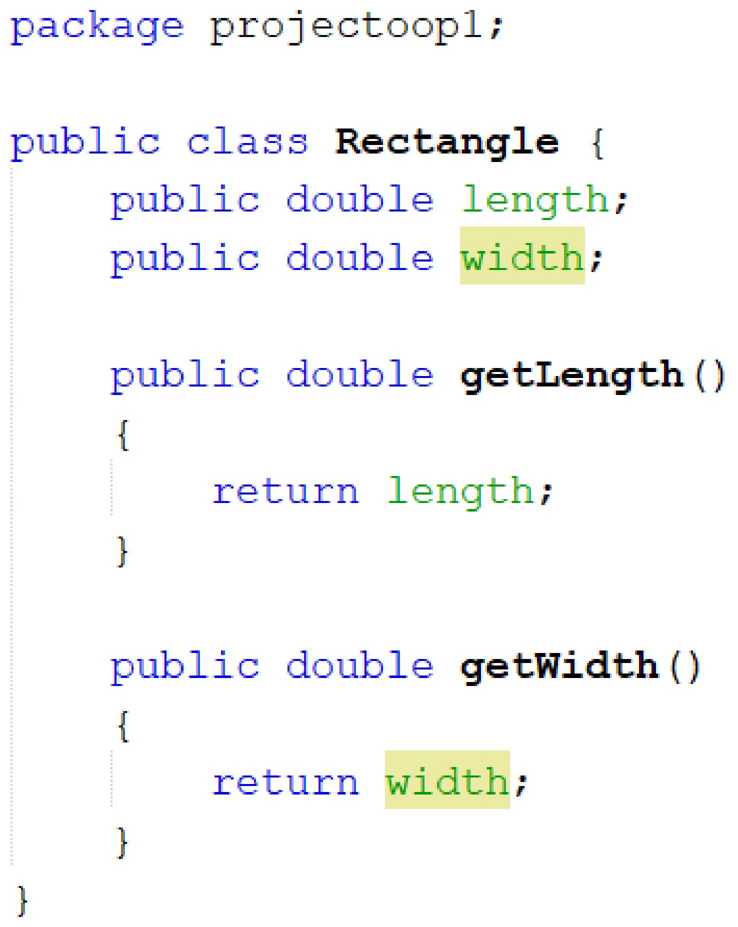


```
package projectoop1;

public class ProjectOOP1 {

    public static void main(String[] args) {
        // TODO code application logic here
        Rectangle r1 = new Rectangle();
        r1.length = 10.5;
        r1.width = 12;

        System.out.print("Length = " + r1.length + "\n");
        System.out.print("Width = " + r1.width + "\n");
    }
}
```

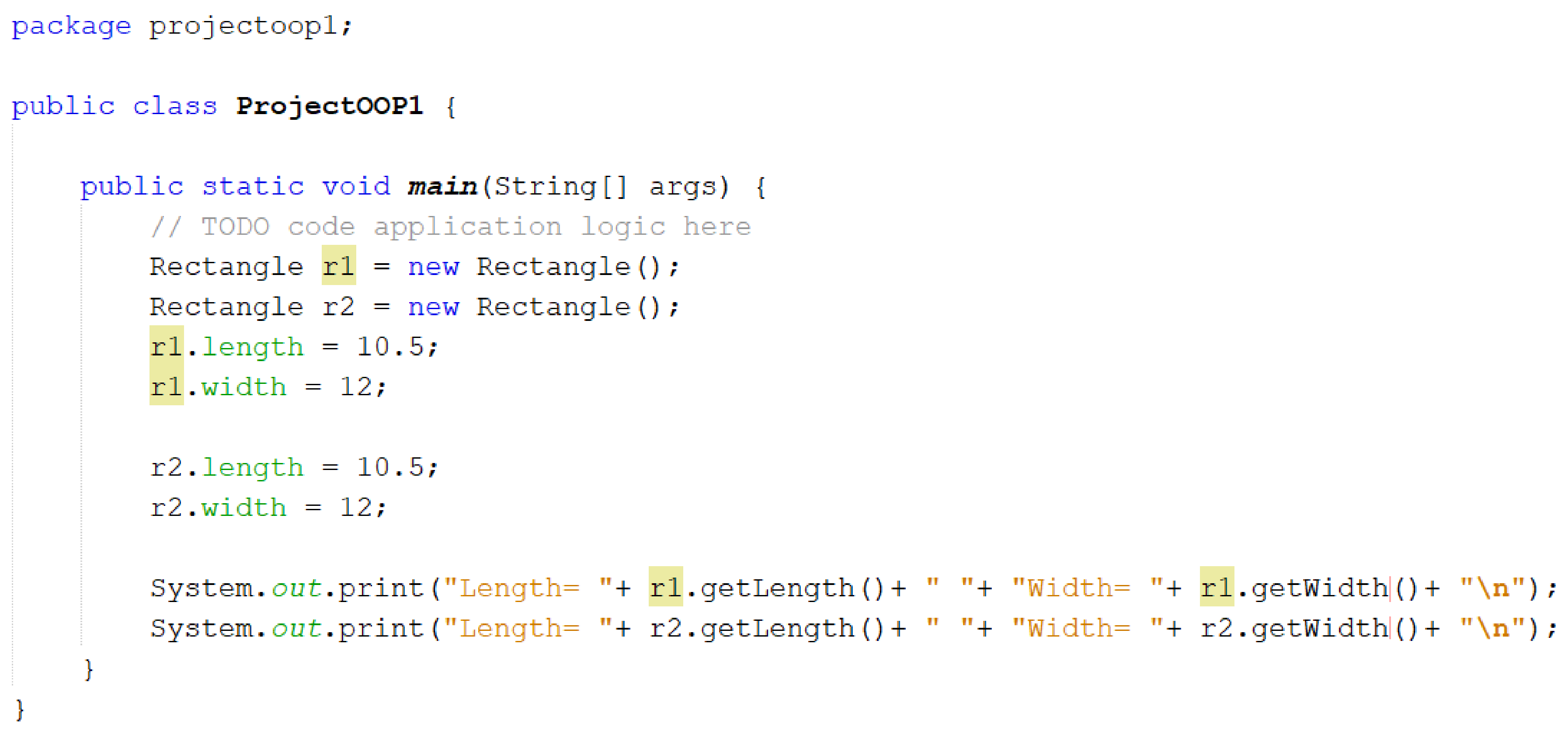
Multiple Objects

```java
package projectoop1;

public class ProjectOOP1 {

    public static void main(String[] args) {
        // TODO code application logic here
        Rectangle r1 = new Rectangle();
        Rectangle r2 = new Rectangle();
        r1.length = 10.5;
        r1.width = 12;

        r2.length = 6.5;
        r2.width = 10;

        System.out.print("Length = " + r1.length + " " + "Width= " + r1.width +  " " +"\n");
        System.out.print("Length = " + r2.length + " " + "Width= " + r2.width +  " " + "\n");
    }
}
```

Class Methods Example(1):

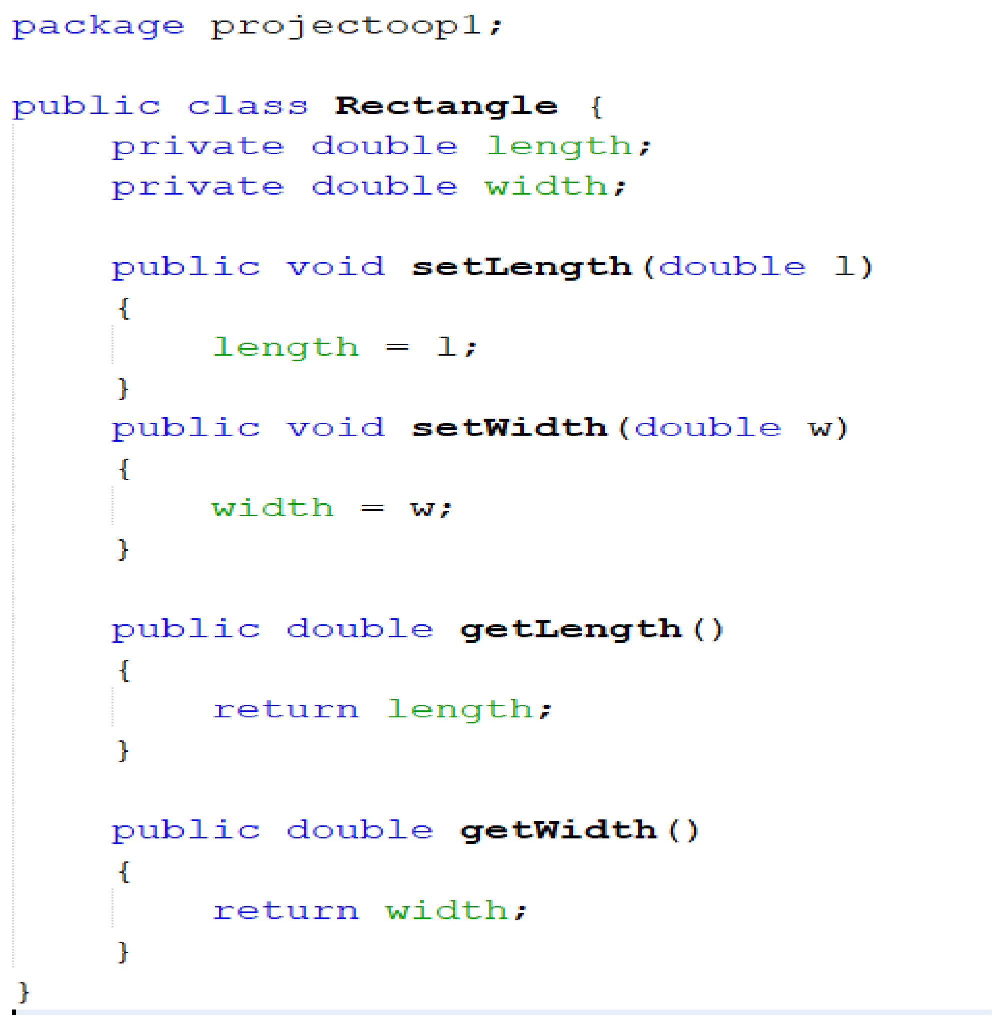


```
package projectoop1;

public class Rectangle {
    public double length;
    public double width;

    public double getLength()
    {
        return length;
    }

    public double getWidth()
    {
        return width;
    }
}
```

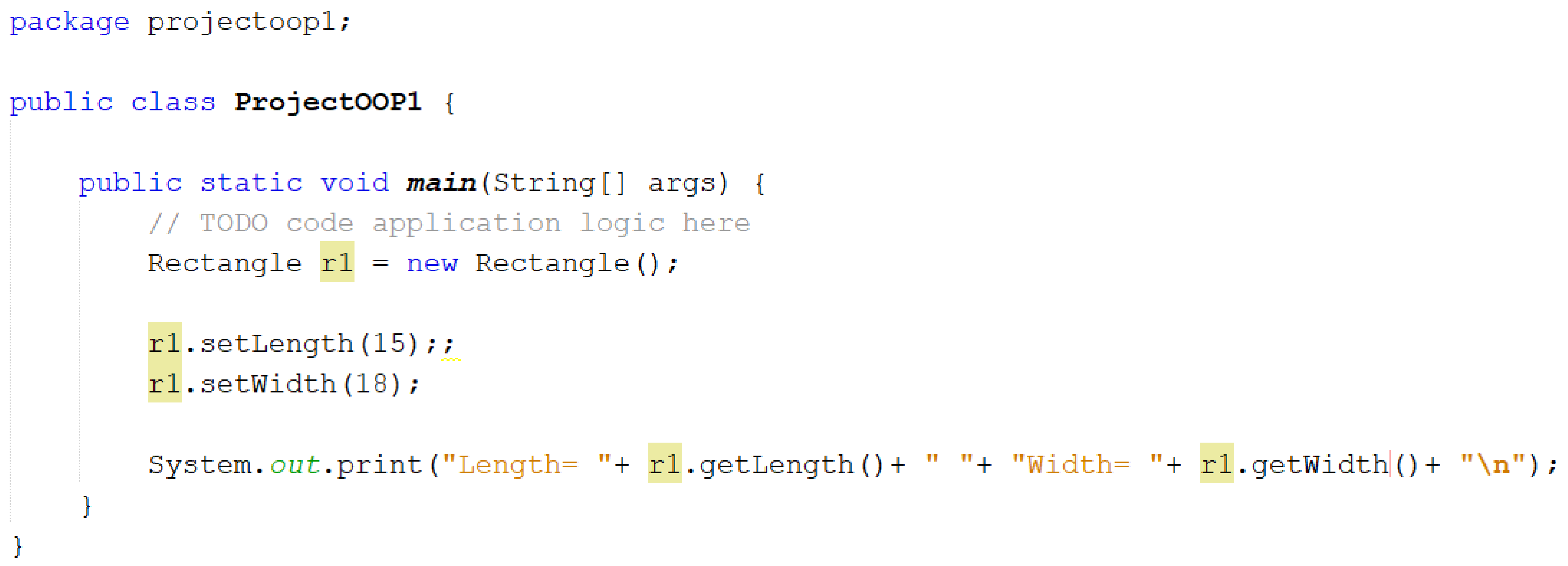


```java
package projectoop1;

public class ProjectOOP1 {

    public static void main(String[] args) {
        // TODO code application logic here
        Rectangle r1 = new Rectangle();
        Rectangle r2 = new Rectangle();
        r1.length = 10.5;
        r1.width = 12;

        r2.length = 10.5;
        r2.width = 12;

        System.out.print("Length= "+ r1.getLength()+ " "+ "Width= "+ r1.getWidth()+ "\n");
        System.out.print("Length= "+ r2.getLength()+ " "+ "Width= "+ r2.getWidth()+ "\n");
    }
}
```

Class Methods Example(2):

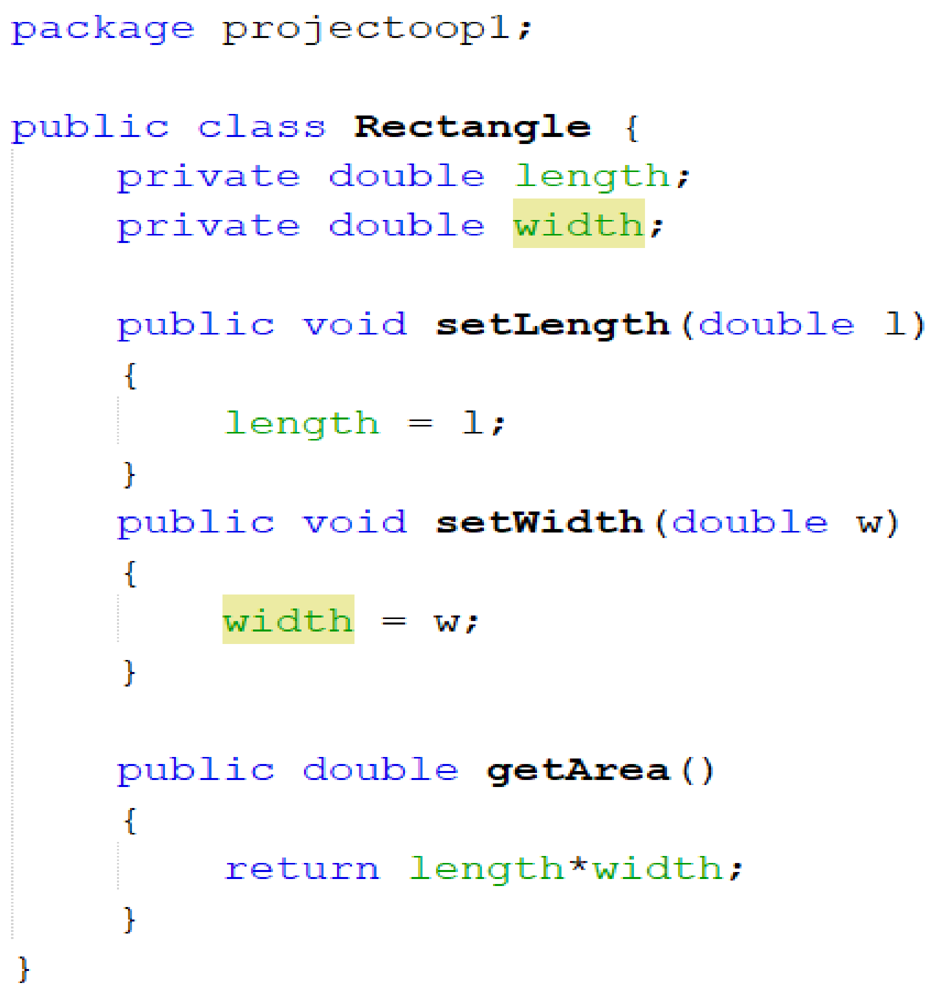


```
package projectoop1;

public class Rectangle {
    private double length;
    private double width;

    public void setLength(double l)
    {
        length = l;
    }
    public void setWidth(double w)
    {
        width = w;
    }

    public double getLength()
    {
        return length;
    }

    public double getWidth()
    {
        return width;
    }
}
```

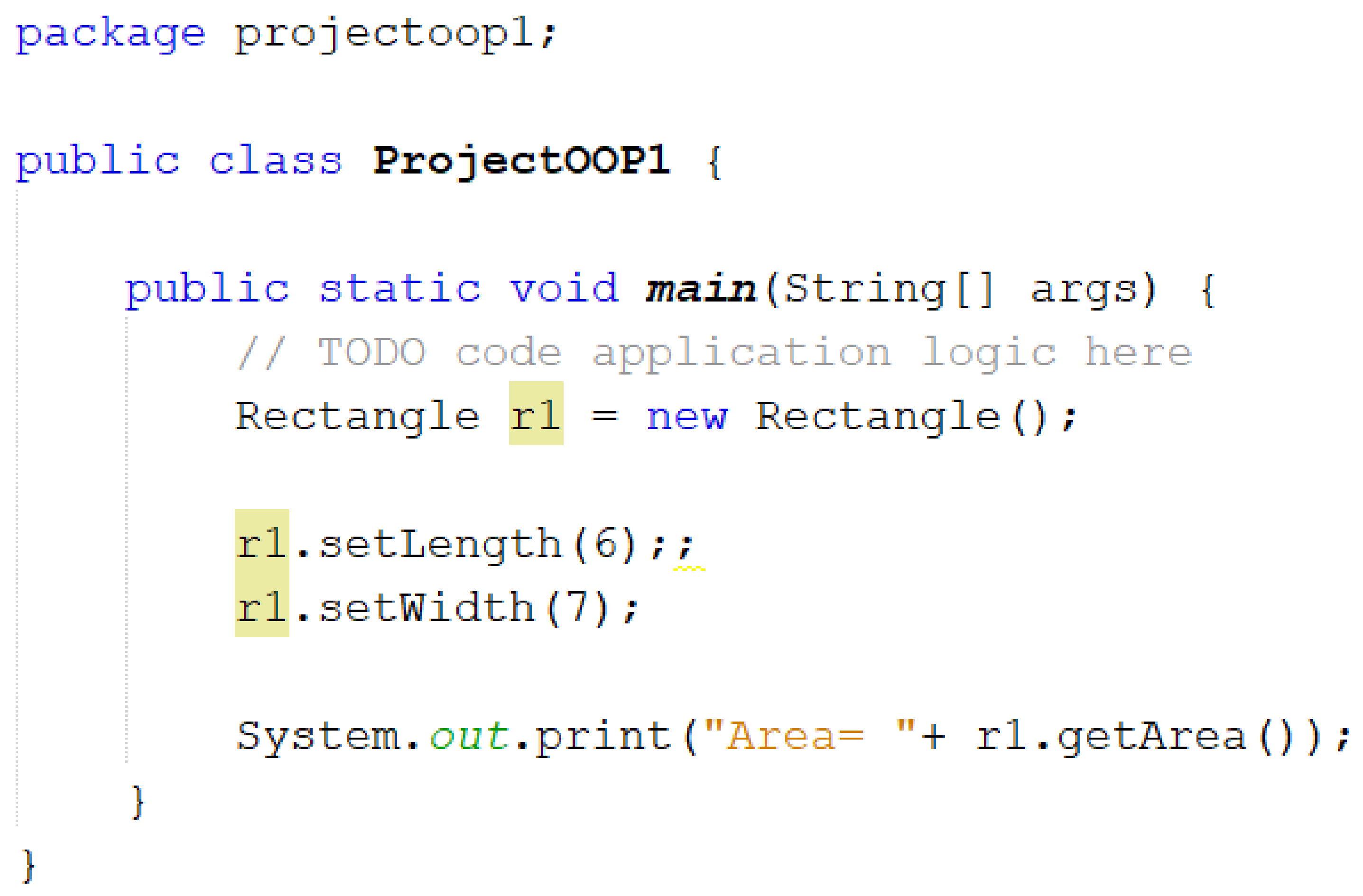


```
package projectoop1;

public class ProjectOOP1 {

    public static void main(String[] args) {
        // TODO code application logic here
        Rectangle r1 = new Rectangle();

        r1.setLength(15);;
        r1.setWidth(18);

        System.out.print("Length= "+ r1.getLength()+ " "+ "Width= "+ r1.getWidth()+ "\n");
    }
}
```

Calculation of area of rectangle



```
package projectoop1;

public class Rectangle {
    private double length;
    private double width;

    public void setLength(double l)
    {
        length = l;
    }
    public void setWidth(double w)
    {
        width = w;
    }

    public double getArea()
    {
        return length*width;
    }
}
```



```
package projectoop1;

public class ProjectOOP1 {

    public static void main(String[] args) {
        // TODO code application logic here
        Rectangle r1 = new Rectangle();

        r1.setLength(6);;
        r1.setWidth(7);

        System.out.print("Area= "+ r1.getArea());
    }
}
```

Thank you