



# **OBJECT ORIENTED PROGRAMMING USING JAVA**

**Lab 3**

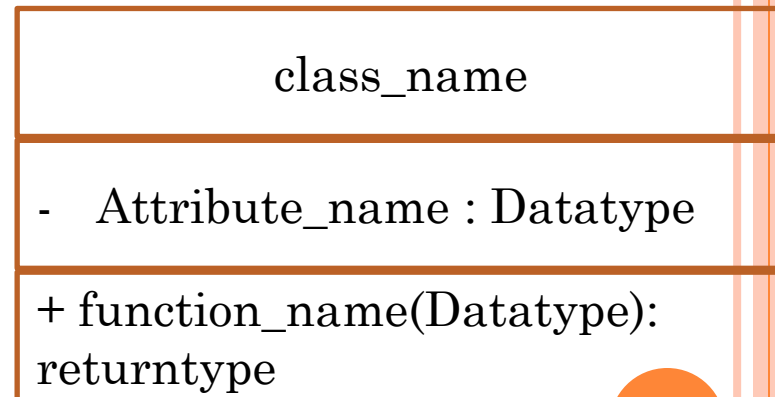
**1**

# CONTENT

- **UML – Class Diagram**
  - ❖ Practice on class diagram
- **Hands-on 1**
- **Access and fields Modifiers**
  - ❖ No modifier, Private, Public, Protected
  - ❖ Static, Final
- **Hands-on 2**
- **Hands-on 3**

# UML – CLASS DIAGRAM

- The UML Class diagram is a graphical notation used to construct and visualize object-oriented systems.
- A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's:
  - Classes
  - Attributes
  - Operations (or methods),
  - The relationships among objects.

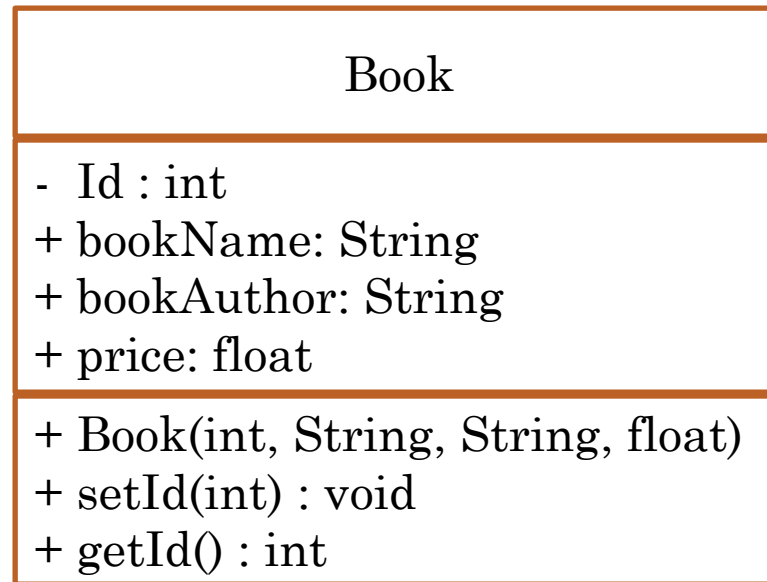


# PRACTICE ON CLASS DIAGRAM

- Draw a class diagram for book class where the book has:
  - Book id as an integer number.
  - Book name as a string
  - Author name as a string
  - The price of the book as a float number.
  - Parametrized constructor for filling all the class attributes.
  - Setter and getter functions for Id attribute.
- All the attributes is public except the id of the book.
- The constructor and methods can be accessed outside the class.

# PRACTICE ON CLASS DIAGRAM CON...

## ○ The Solution

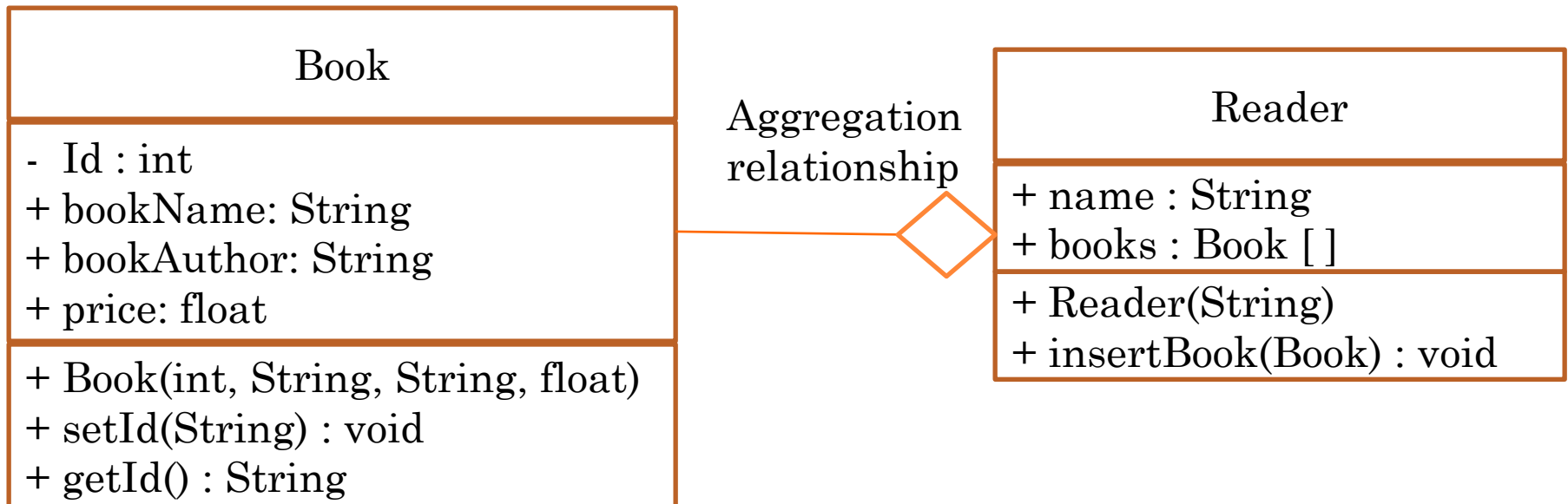


## PRACTICE ON CLASS DIAGRAM CON...

- Update the pervious class diagram by adding a new class which called reader.
- Reader class has:
  - Reader name as a string.
  - Array of books.
  - Parametrized constructor with one parameter for reader name.
  - InsertBook function which takes an object of book as a parameter and adds it into the array.
- All attributes and methods can be accessed outside the class.

# PRACTICE ON CLASS DIAGRAM CON...

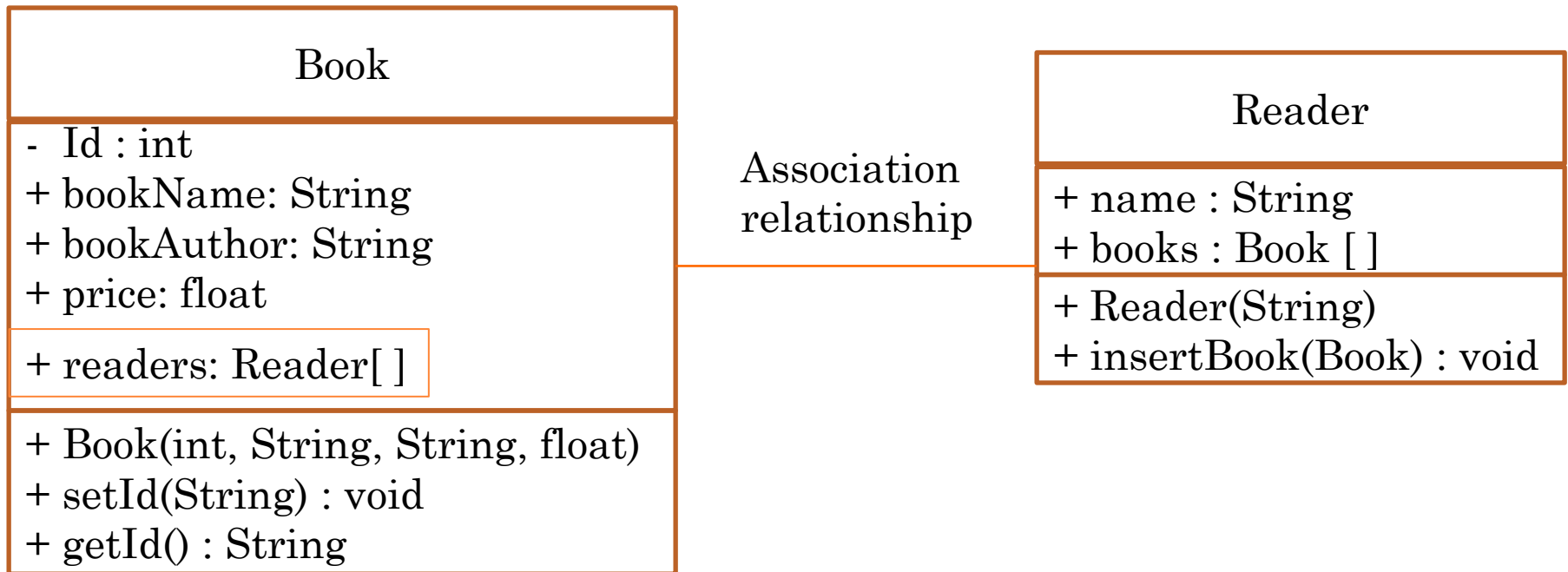
- The Solution:



- Aggregation relationship: One object has or owns another object.
- Books is part of Reader.

# PRACTICE ON CLASS DIAGRAM CON...

- The Solution:



- Bidirectional Association: Two objects might store each other in fields.



# HANDS-ON 1

- Create class point that contains two integers x and y.
- x and y couldn't be accessed outside the class.
- Create class rectangle that contains two points from class point.
- Create a parameterized constructor which has 2 points as a parameters.
- Implement calculateArea method inside rectangle class to calculate the area, then return its value.
- In the main class create object of rectangle, then call calculateArea function.
- Draw UML class diagram for these two classes.

A rectangle is defined by two points

(X1,Y1)



(X2,Y2)

$$\text{Area} = |X1 - X2| * |Y1 - Y2|$$

# SOLUTION

```
class Point{
    private int x , y;
    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
    public int getX()
    { return x;}
    public int getY()
    { return y;}
}
```

```
class Rectangle{
    Point p1;
    Point p2;

    public Rectangle(Point p1, Point p2){
        this.p1 = p1;
        this.p2 = p2;
    }

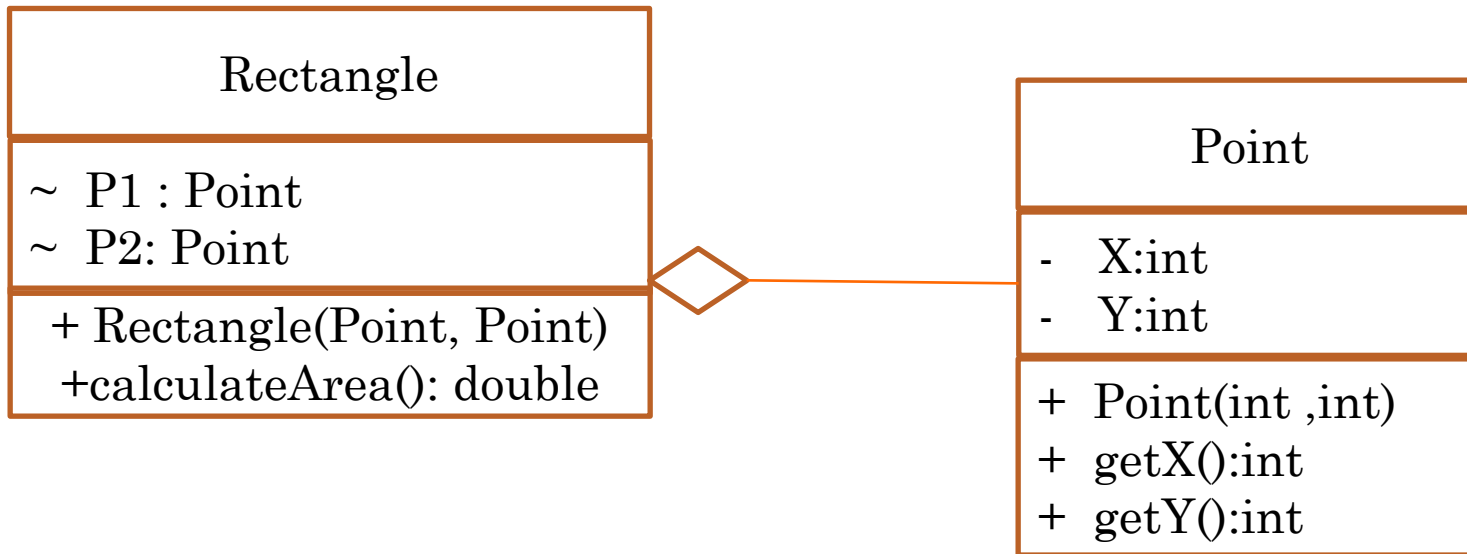
    public double calcArea()
    {
        double area =
            Math.abs(p1.getX() - p2.getX()) *
            Math.abs(p1.getY() - p2.getY());

        return area;
    }
}
```

## SOLUTION CON...

```
package oop_lab2;
public class uml_practice {
    public static void main(String[] args) {
        Point p1 = new Point(3,2);
        Point p2 = new Point(10,20);
        Rectangle rec = new Rectangle(p1, p2);
        double area = rec.calcArea();
        System.out.println(area);
    }
}
```

## SOLUTION CON...



Aggregation Relation

# ACCESS MODIFIERS

- The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class.
- We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

Access Modifier	within class	within package	outside package by subclass only	outside package
<b>Private</b>	Y	N	N	N
<b>Default</b>	Y	Y	N	N
<b>Protected</b>	Y	Y	Y	N
<b>Public</b>	Y	Y	Y	Y

## FIELD MODIFIERS CON...

### ○ **Static Field (Class Variable):**

- Sometimes, you want to have variables that are common to all objects. This is accomplished with the static modifier.
- Fields that have the **static** modifier in their declaration are called *static fields* or *class variables*.

### ○ **Final Field:**


- A final field cannot have its value changed, once assigned.

# ACCESS MODIFIERS CON... FIND THE MISTAKE AND TRY TO SOLVE IT

```
package main_Package;
class Citizen {
    private String SSN;
    protected String name;
    public int ID;
    public Citizen(String name, int ID)
    {
        this.name = name;
        this.ID = ID;
    }
    public Citizen(String name,int ID,
String SSN) {
        this.name = name;
        this.ID = ID;
        this.SSN = SSN;
    }
}
```

```
package main_Package;

public class MainClass {
    public static void main(String[] args)
    {
        Citizen s;
        s.ID =10;
        System.out.print(s.ID);
    }
}
```



## 2 Solutions:

1. You must initialize variable s.

## Example

```
Citizen s = new Citizen("Nada", 1);
```

# ACCESS MODIFIERS CON... FIND THE MISTAKE AND TRY TO SOLVE IT

```
package new_Package;
public class Citizen {
    String SSN;
    protected String name;
    protected int ID;
    public static int
    citizenNumber =0;
    public Citizen(String name, int ID,
    String SSN) {
        this.name = name;
        this.ID = ID;
        this.SSN = SSN;
        citizenNumber++;
    }
    public void setID( int id)
    { this.ID = id; }
}
```

```
package main_Package;
import new_Package.Citizen;

public class MainClass {
    public static void main(String[] args) {
        Citizen s[ ] =new Citizen[3];
        s[0].setID(15);
    }
}
```



Citizen s[0] isn't initialized.

2 Solutions:

```
s[0] = new Citizen("Nada", 1, "123");
s[0].setID(15);
```



# ACCESS MODIFIERS CON... FIND THE MISTAKE AND TRY TO SOLVE IT

```
package new_Package;
class Citizen {
    private String SSN;
    protected String name;
    public int ID;
    public Citizen(String name, int ID)
    {
        this.name = name;
        this.ID = ID;
    }
    public Citizen(String name,int ID,
String SSN) {
        this.name = name;
        this.ID = ID;
        this.SSN = SSN;
    }
}
```

```
package main_Package;

public class MainClass {
    public static void main(String[] args)
    {
        Citizen s =new Citizen("Nada",1);
    }
}
```



Citizen class has no modifier.  
It couldn't be accessed from another package.

## 2 Solutions:

1. Put both classes in the same package.
2. Declare citizen class as a public class. Then import it mainClass file.

# ACCESS MODIFIERS CON... FIND THE MISTAKE AND TRY TO SOLVE IT

```
package main_Package;
class Citizen {
    private String SSN;
    protected String name;
    public int ID;

    public Citizen(String name,int ID,
String SSN) {
        this.name = name;
        this.ID = ID;
        this.SSN = SSN; }
}
```

```
package main_Package;
public class MainClass {
    public static void main(String[] args)
    {
        Citizen s;
        s =new Citizen("Nada",1,
"012345678");

        System.out.print(s.SSN);
    }
}
```



SSN is declared as a private attribute.


Solution:

1. Create getSSN function in citizen class to return the value of ssn and use it in the mainClass.

# ACCESS MODIFIERS CON... FIND THE MISTAKE AND TRY TO SOLVE IT

```
package new_Package;
public class Citizen {
    private String SSN;
    protected String name;
    public int ID;
    public Citizen(String name,
int ID) {
    this.name = name;
    this.ID = ID;
}
public Citizen(String
name,int ID, String SSN) {
    this.name = name;
    this.ID = ID;
    this.SSN = SSN;
}
}
```

```
package main_Package;
import new_Package.Citizen;
import java.util.Scanner;
public class MainClass {
    public static void main(String[] args) {
        Citizen s =new Citizen("Nada",1);
        Scanner x = new Scanner(System.in);
        String n = x.next();
        if(n.equals(s.name)){
            System.out.println(s.ID);
        }
    }
}
```



Name attribute is protected  
So, it couldn't be accessed  
directly by calling it

## 2 Solutions:

1. Put both classes in the same package.
2. Create getName function to be able to retrieve the value of the name.

# ACCESS MODIFIERS CON... FIND THE MISTAKE AND TRY TO SOLVE IT

```
package main_Package;
public class Citizen {
    private String SSN;
    protected String name;
    public int ID;
    public static final int
    citizenNumber =0;
    public Citizen(String name, int ID,
    String SSN) {
        this.name = name;
        this.ID = ID;
        this.SSN = SSN;
        citizenNumber++;
    }
}
```



```
package main_Package;
import java.util.Scanner;
public class MainClass {
    public static void main(String[] args) {
        Citizen s =new Citizen("Nada",1, "123");
        Scanner x = new Scanner(System.in);
        String n = x.next();
        if(n.equals(s.name)){
            System.out.println(s.ID);
        }
    }
}
```

CitizenNumber is declared as final field. So, you couldn't change its value.

## 2 Solutions:

1. Remove final keyword
2. Remove citizenNumber++ from the constructor.

# ACCESS MODIFIERS CON... FIND THE MISTAKE AND TRY TO SOLVE IT

```
package main_Package;
public class Citizen {
    private String SSN;
    protected String name;
    public int ID;
    public static int
    citizenNumber =0;
    public Citizen(String name, int ID,
    String SSN) {
        this.name = name;
        this.ID = ID;
        this.SSN = SSN;
        citizenNumber++;
    }
}
```

```
package main_Package;
import java.util.Scanner;
public class MainClass {
    public static void main(String[] arg
    Citizen s =new Citizen("Nada",1);
    Scanner x = new Scanner(System.in);
    String n = x.next();
    if(n.equals(s.name)){
        System.out.println(s.ID);
    }
}
```



- You have only one parametrized constructor with three parameters.
- In the main, you use a parametrized constructor with 2 parameters.

## Solution:

1. Create an additional constructor which has 2 parameters .

# ACCESS MODIFIERS CON... FIND THE MISTAKE AND TRY TO SOLVE IT

```
package main_Package;
public class Citizen {
    String SSN;
    protected String name;
    protected final int ID;
    public static int
    citizenNumber =0;
    public Citizen(String name, int ID,
    String SSN) {
        this.name = name;
        this.ID = ID;
        this.SSN = SSN;
        citizenNumber++;
    }
    public void setID( int id)
    { this.ID = id; }
}
```

```
package main_Package;
import java.util.Scanner;
public class MainClass {
    public static void main(String[] args) {
        Citizen s =new Citizen("Nada",1, "123");
        Scanner x = new Scanner(System.in);
        String n = x.next();
        int id = x.nextInt();
        if(n.equals(s.name)){
            s.setID(id);
            Sys
        }
    }
}
```



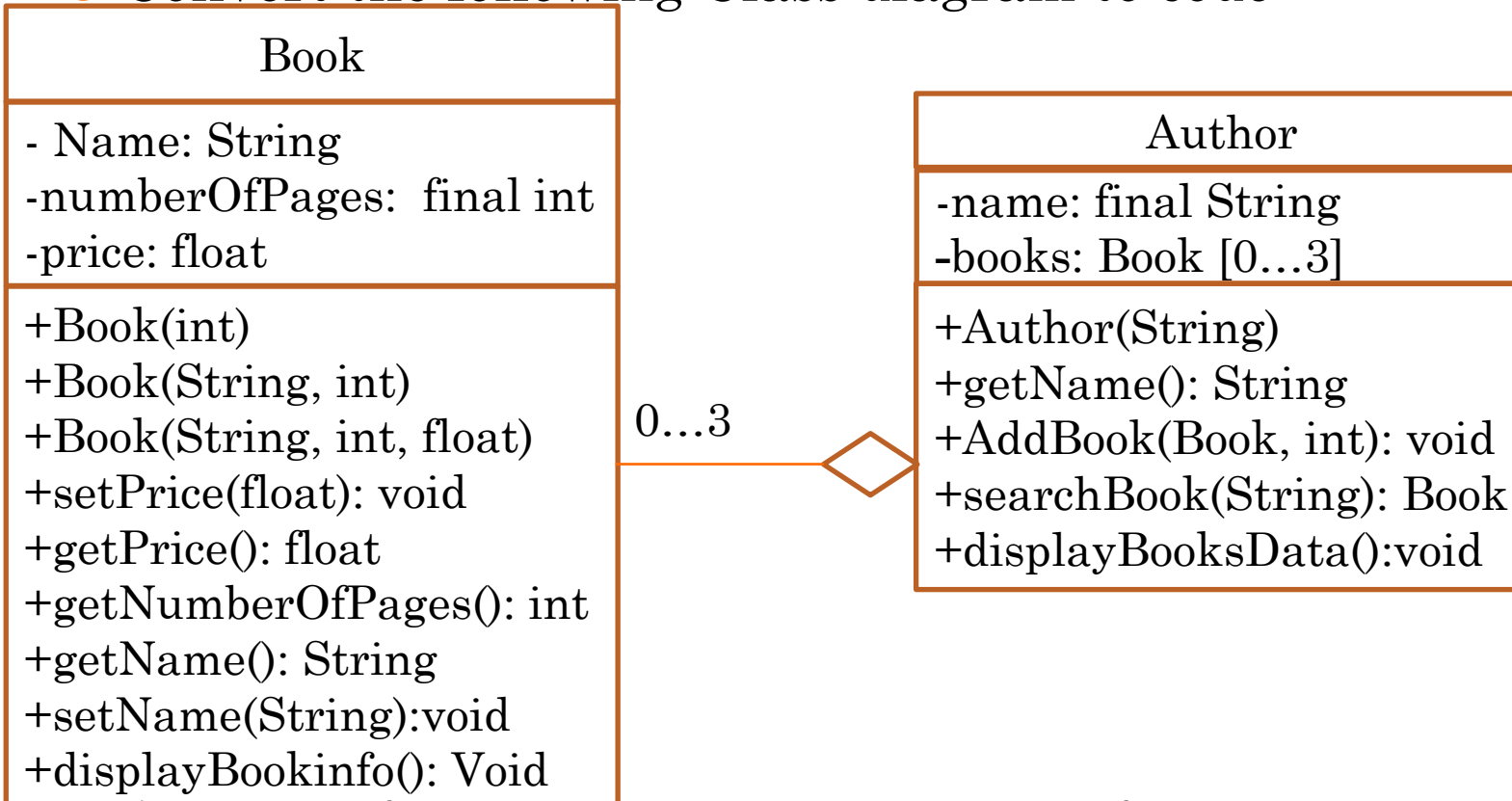
- ID has final modifier. So, you aren't allowed to edit it.

## 2 Solutions:

1. Remove final keyword.
2. Remove setID function from Citizen class.  
Remove s.setID(id) from the main function.

## HANDS-ON 2

- Convert the following Class diagram to code



- AddBook function: it takes an object of book and index to set the book into the array using the index.
- searchBook: it takes a name of book. If the book is exists, then return it else return null.
- displayBooksData: display name, number of pages and price of each book.

# HANDS-ON 2 SOLUTION

```
public class Book {
    private String name;
    private final int numberOfPages;
    private float price;
    public Book(int numberOfPages) {
        this.numberOfPages =
numberOfPages;
    }
    public Book(String name, int
        numberOfPages) {
        this.name = name;
        this.numberOfPages =
numberOfPages;
    }
    public Book(String name, int
        numberOfPages, float price) {
        this.name = name;
        this.numberOfPages =
        numberOfPages;
        this.price = price;
    }
}
```

```
public String getName() {
    return name;
}
public int getNumberOfPages() {
    return numberOfPages;
}
public float getPrice() {
    return price;
}
public void setName(String name) {
    this.name = name;
}

public void setPrice(float price) {
    this.price = price;
}
public void displayBookinfo()
{
    System.out.println("The book name is "+
        name);
    System.out.println("Number of pages "
        +numberOfPages);
    System.out.println("The book price is “
        +price+" $");
}
}
```



# HANDS-ON 2 SOLUTION

```
public class Author {
    private final String name;
    private Book books[] =new Book[3];
    public Author(String name) {
        this.name = name;
        for(int i=0;i<3;i++)
        {
            books[i]=null;
        }
    }

    public void addBook(Book newBook, int index)
    {
        if(index >=0 && index <3)
        {
            books[index]=newBook;
        }
        else
        {
            System.out.println("Please enter an index
between 0 and 3");
        }
    }
}
```

```
public Book searchBook(String name)
{
    Book theBook=null;
    for(int i=0;i<3;i++)
    {
        if(books[i]!=null &&
books[i].getName().equals(name))
        {
            theBook = books[i];
        }
    }
    return theBook;
}

public void displayBooksData()
{
    for(int i=0;i<3;i++)
    {
        if(books[i]!=null)
        {
            books[i].displayBookinfo();
        }
    }
}
}
```

## HANDS-ON 3

- Create main class which has main function:
  - Create object of author.
  - Create 3 objects of book.
    - The first one has only number of pages.
    - The second one has number of pages and name.
    - The third one has number of pages, name and price.
  - Set a value of all missing data in the books.
  - Add the books into the author.
  - Search a book using its name. if you found the book display its info.
  - Display all the author's books.

# HANDS-ON 3 SOLUTION

```
public class MainClass {  
  
    public static void main(String[] args) {  
        Author author1=new Author("Mary");  
        Book book1=new Book(50);  
        book1.setName("ABC");  
        book1.setPrice(50.5f);  
        Book book2=new Book("Hello",60);  
        book2.setPrice(60.5f);  
        Book book3=new Book("New World",60,70.6f);  
        author1.addBook(book1, 0);  
        author1.addBook(book2, 1);  
        author1.addBook(book3, 2);  
        Book b = author1.searchBook("Hello");  
        if(b!=null)  
        {  
            b.displayBookinfo();  
        }  
        author1.displayBooksData();  
    }  
}
```

# QUESTIONS

