Object Oriented Programming using Java

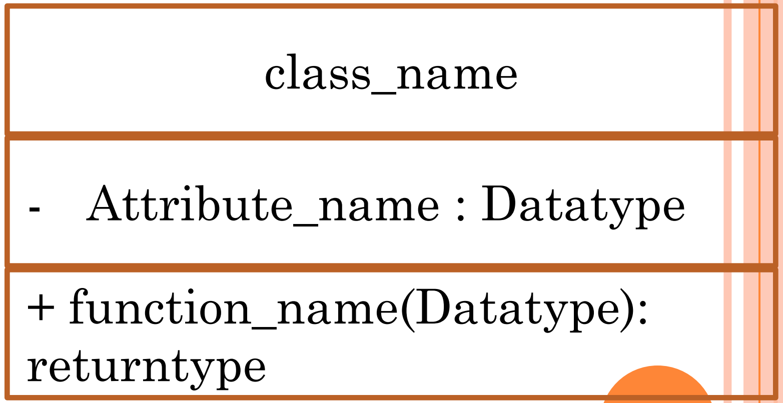
Lab 3

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.

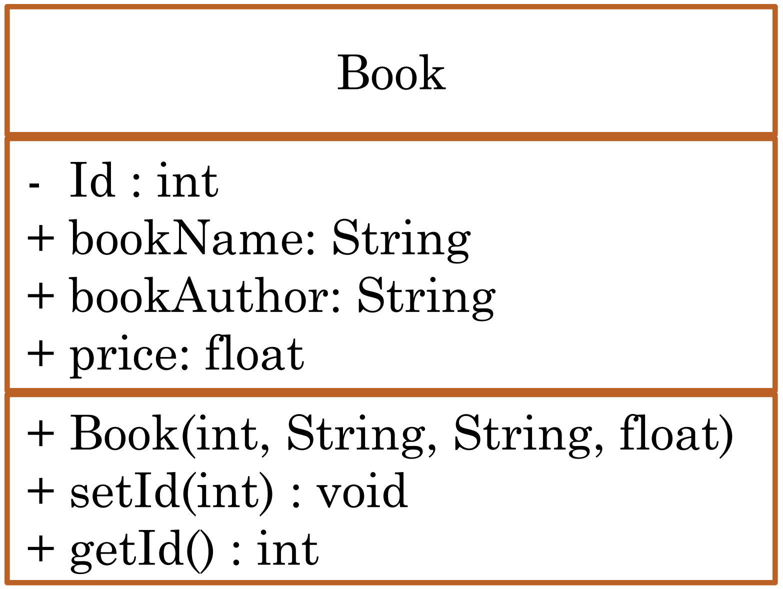

class_name
- Attribute_name : Datatype
+ function_name(Datatype): returntype

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

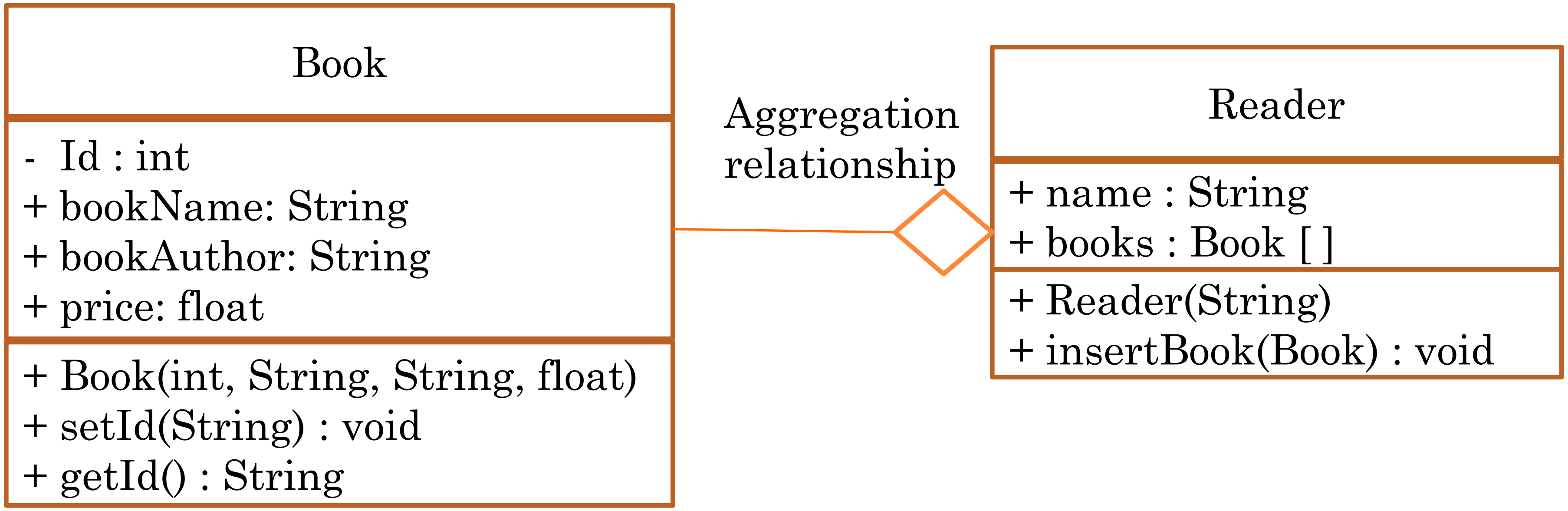


Practice on Class diagram con…

- 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…

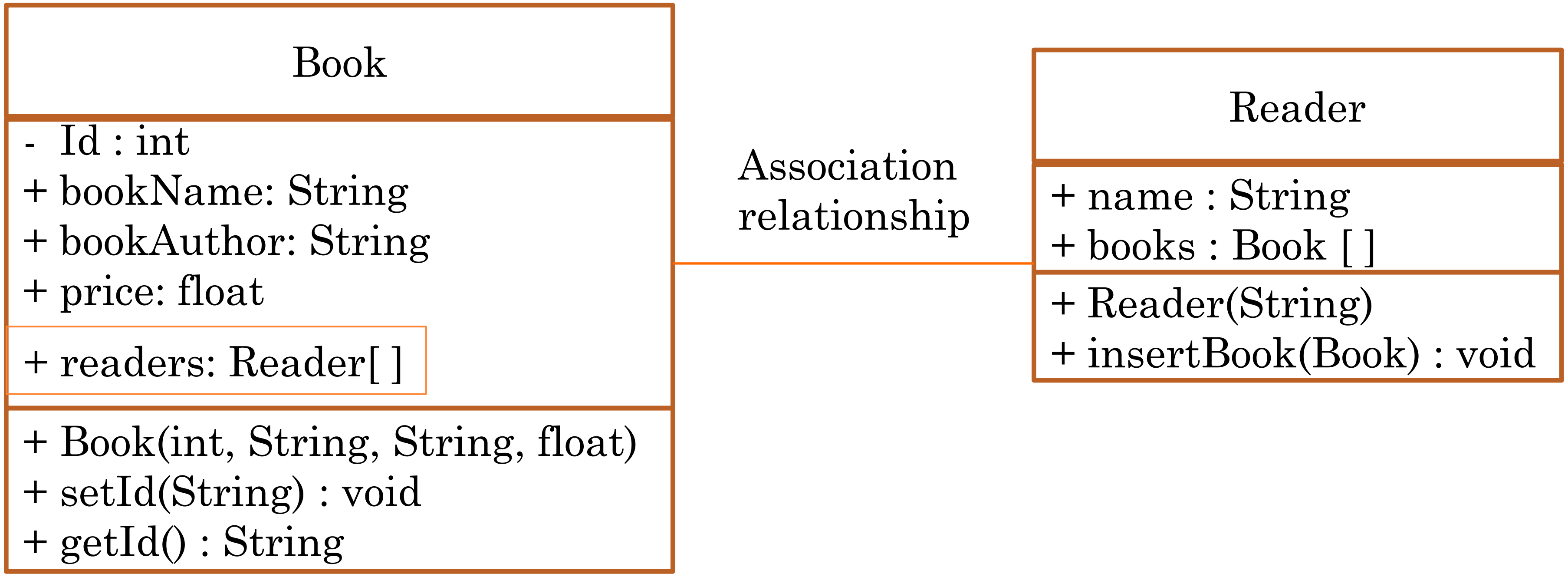
- The Solution:



- 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.

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
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	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.

<u>2 Solutions:</u>
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

<u>2 Solutions:</u>

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.

<u>2 Solutions:</u>

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[] 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);
        }
    }
}
```

- 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.

<u>2 Solutions:</u>

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

QUESTIONS