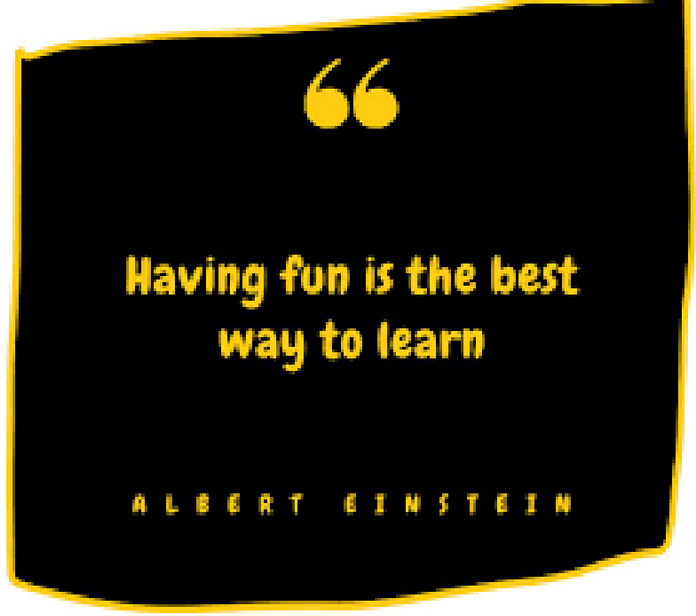
Classes, Objects, and Composition

Colorado State University

Department of Computer Science

Slides Originally Created by Marcia Moraes(Marcia.Moraes@colostate.edu)
and Albert Lionelle (albert.lionelle@colostate.edu)

Announcements



TODO Reminders:
Readings are due **before** lecture

- Reading 8 (zybooks) – you should have already done that ☺
- Lab 05
- Reading 9 (zyBooks) – you should have already done that ☺
- Lab 06
- Reading 10 (zybooks)
- Keep practicing your RPAs in a spaced and mixed manner ☺

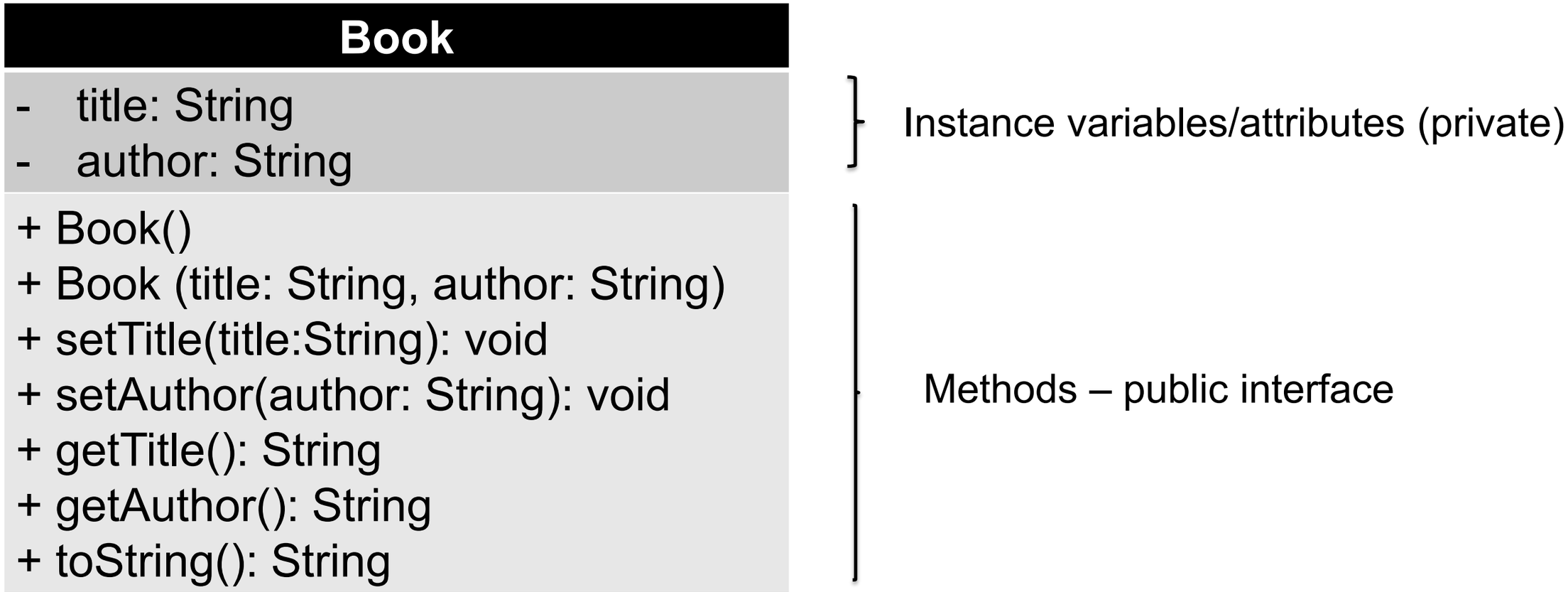
Class

- Describes a set of objects with the same behavior.
  - String class describes the behavior of all strings
    - Specifies how a string stores its characters
    - Which methods can be used (behaviors)
    - How the methods are implemented

Class

- Has a public interface
  - Collection of methods through which the objects of the class can be manipulated
- Stores its data in instance variables (attributes)
  - Data required for executing the methods
  - Instance variables should always be private
  - private instance variables can only be accessed by the methods of its own class

Class Book


Book
- title: String
- author: String
+ Book()
+ Book (title: String, author: String)
+ setTitle(title:String): void
+ setAuthor(author: String): void
+ getTitle(): String
+ getAuthor(): String
+ toString(): String
Instance variables/attributes (private)
Methods – public interface

Class Book

Book
- title: String - author: String
+ Book() + Book (title: String, author: String)
+ setTitle(title:String): void + setAuthor(author: String): void + getTitle(): String + getAuthor(): String + toString(): String

Overloaded constructor

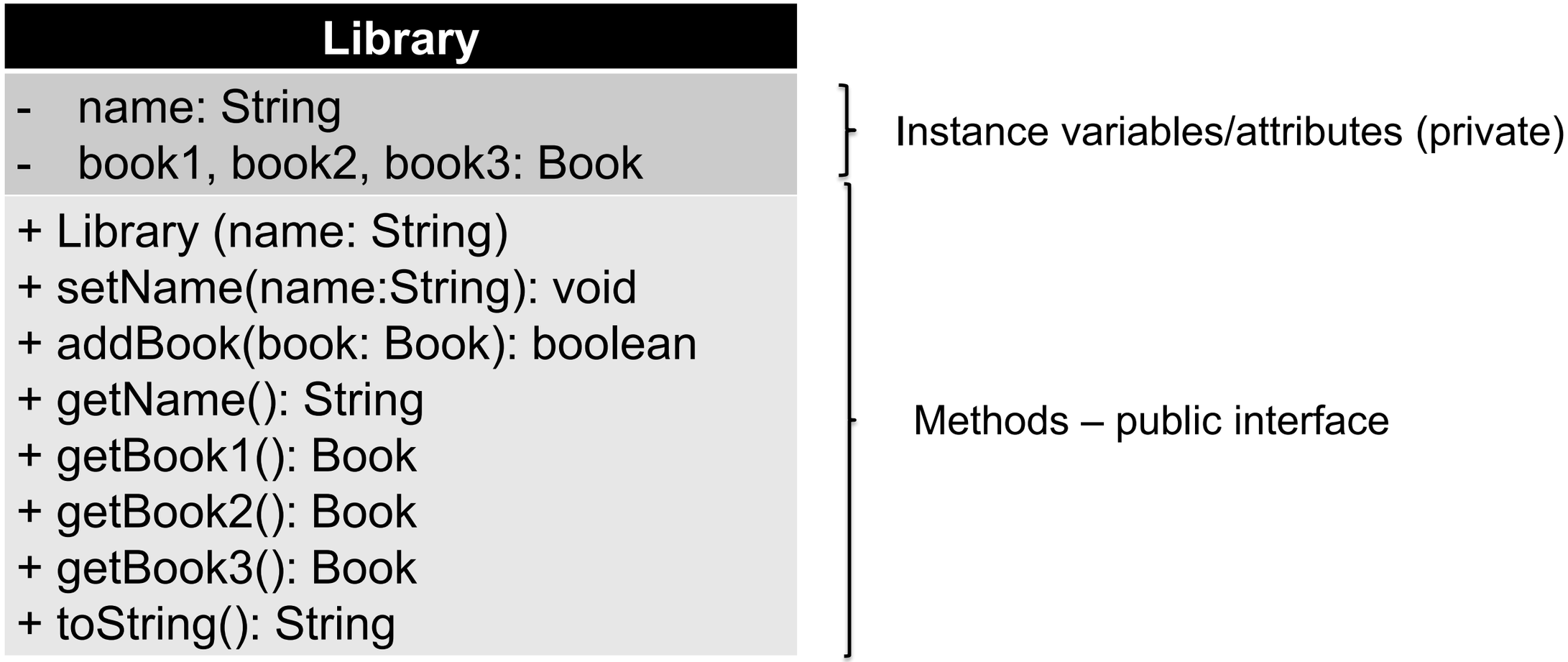
- Same name of method with different parameters
- Each method has a different implementation

We can have overload of any method, constructors or any other method defined in the class.

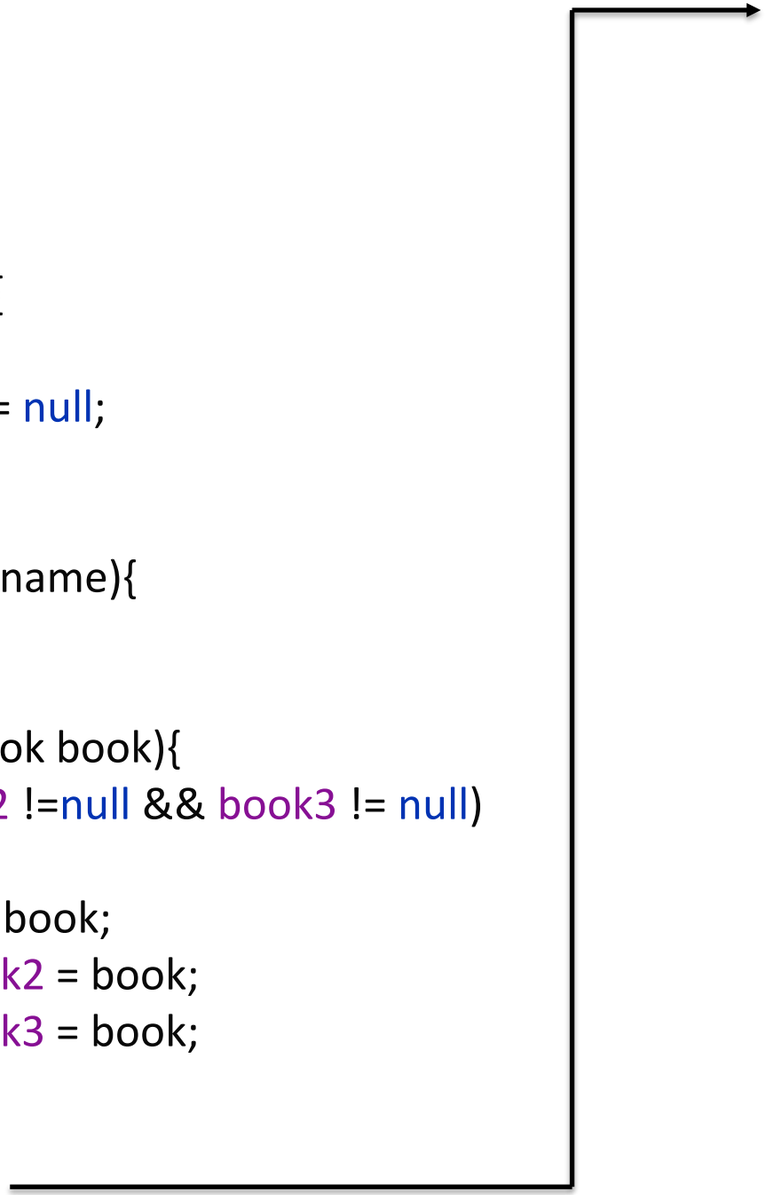
```
public class Book {
    // Instance variables --> attributes
    private String title;
    private String author;
    // Constructor of this class
    public Book(){
        setTitle("");
        setAuthor("");
    }
    public Book(String title, String author)
    {
        setTitle(title);
        setAuthor(author);
    }
    //mutator methods
    public void setTitle(String title){
        this.title = title;
    }
    public void setAuthor(String author){
        this.author = author;
    }
    //accessor methods
    public String getTitle(){
        return title;
    }
    public String getAuthor(){
        return author;
    }
    //toString method
    public String toString(){
        return "Title: " + title + " Author: " + author;
    }
}
```

Composition

- One class has as an instance variable another class

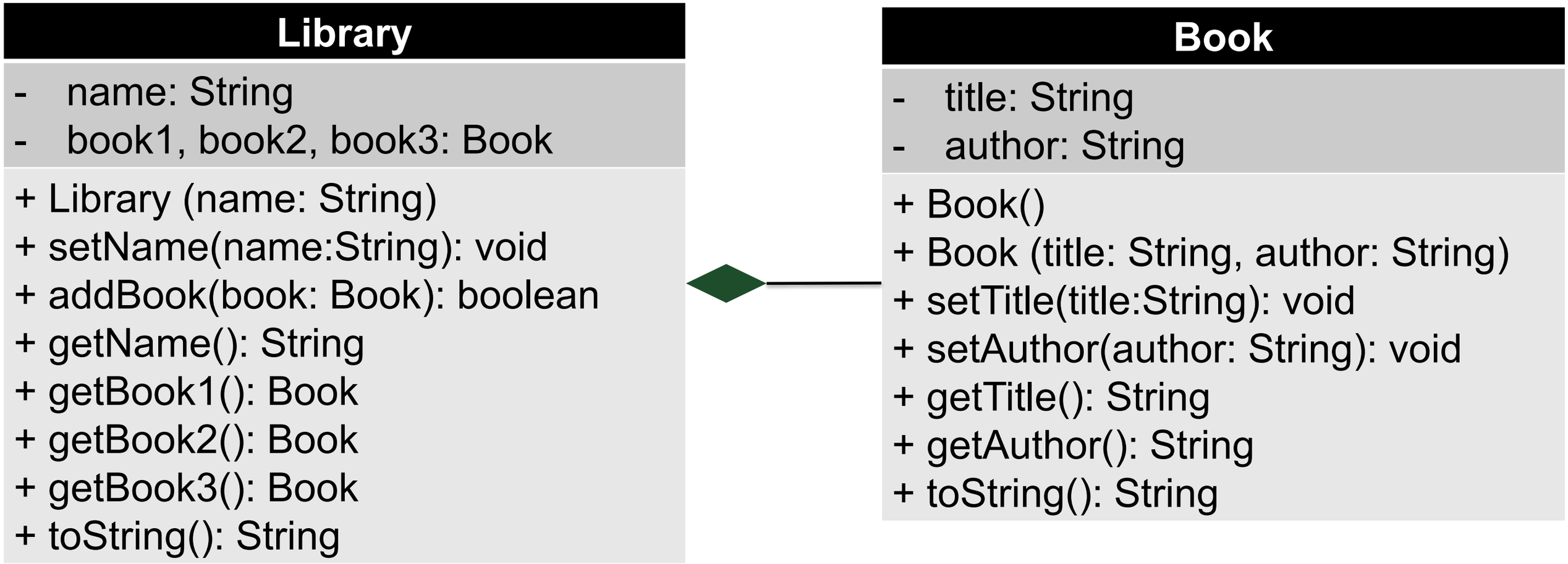


```
public class Library {
   private String name;
   private Book book1;
   private Book book2;
   private Book book3;
   public Library(String name){
      setName(name);
      book1 = null;        book2 = null;
      book3 = null;
   }
   public void setName(String name){
      this.name = name;
   }
   public boolean addBook(Book book){
      if(book1 != null && book2 !=null && book3 != null)
         return false;
      if(book1 == null) book1 = book;
      else if(book2 == null) book2 = book;
      else if(book3 == null) book3 = book;
      return true;
   }
```

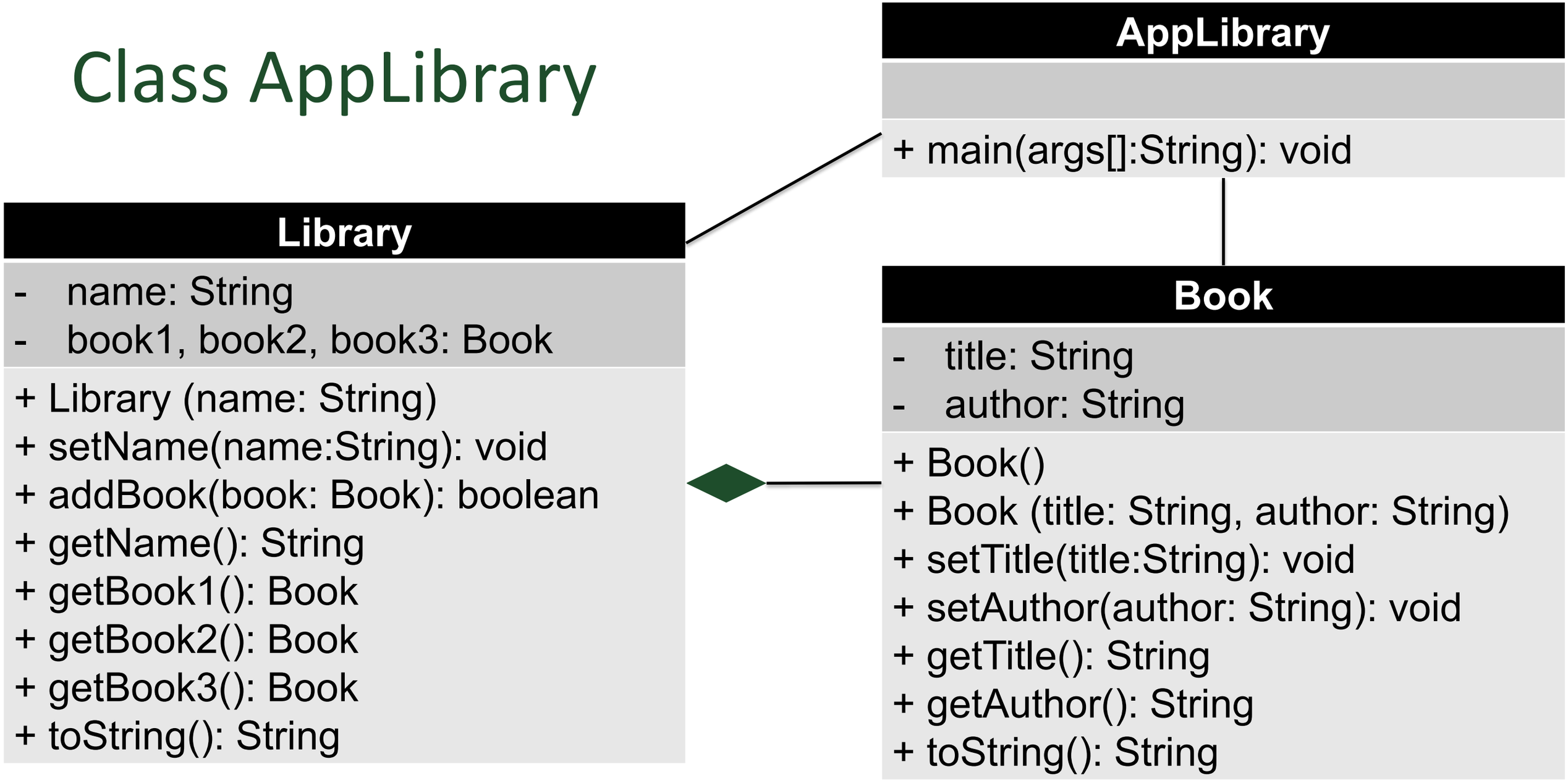


```
   public String getName(){
      return name;
   }
   public Book getBook1(){
      return book1;
   }
   public Book getBook2(){
      return book2;
   }
   public Book getBook3(){
      return book3;
   }
   public String toString(){
      String msg = "";
      if(book1 != null) msg += book1 + "\n";
      if(book2 != null) msg += book2 + "\n";
      if(book3 != null) msg += book3 + "\n";
      if(msg.equals("")) msg = "No books in the
library!";
      return msg;
   }
}
```

Class Library is composed by instances of Class Book – UML Composition link



Class AppLibrary



Class AppLibrary – Using previous classes as variables inside the main method

```
public class AppLibrary {
    public static void main(String args[]){
        Library lib = new Library("Library");
        System.out.println(lib.toString());
        Book b1 = new Book("Death on the Nile", "Agatha Christie");
        if(lib.addBook(b1)) System.out.println("Book added!");
        else System.out.println("No more space in the library!");
        System.out.println(lib.toString());
    }
}
```

Classes have variables / data

- Classes can have any number of variables and types in them
  - Variables in the class code block
    - May be static
      - Variable value is shared across all classes / the program
    - May be instance (not-static)
      - Variable value is only set for every instance / object uniquely
        - » Length of String only makes sense for unique strings!
  - Variables may have scope
    - Who has access to read them
    - public – everyone in every class can read and write to them
    - private – only methods in that class can read and write to them (suggested!)
  - Can be changed or can't be changed
    - final – defines a constant value – can't be changed

who has access to it - ideally private unless reason

```
pub  class Cake {

    public static final boolean IS_GOOD = true;
    private String name;
    private double cost;

}
```

all variables have type

Access a static variable →
Need to use className.variableName

Cake.IS_GOOD

Analyze this code to answer the question

```
public class Hero {
     public static final String LEAGUE = “HERO”;

     public String powerLookup(int which) {
      final String rtn = LEAGUE + “: Flight”;

      if(which < 0) {
          rtn = LEAGUE + “: Laser Eyes”;
      }

      return rtn;
    }


    public static void main(String[] args) {

       System.out.println(Hero.LEAGUE);
       Hero.LEAGUE = “Villain”;

       Hero ajax = new Hero(); // must build the object
       System.out.println(ajax.powerLookup(-1));

    }
}
```

- Will this code compile?
  - A – Yes
  - B – No
  - C – Not sure

Let's Look At Code

```
public class Hero {
     public static final String LEAGUE = "HERO";

     public String powerLookup(int which) {
      final String rtn = LEAGUE + ": Flight";

      if(which < 0) {
          rtn = LEAGUE + ": Laser Eyes";
      }

      return rtn;
    }


    public static void main(String[] args) {

       System.out.println(Hero.LEAGUE);
       Hero.LEAGUE = "Villain";

       Hero ajax = new Hero(); // must build the object
       System.out.println(ajax.powerLookup(-1));

    }
}
```

Will not compile

Will not compile

- league is accessible by all other classes by saying
  - `Hero.LEAGUE`
  - Since it is static and public, not because it is final. However, other classes can't change it since it is final.
- This code would not compile
  - the compiler would error on the code in the if block
  - it is trying to *reset* the value in **rtn**
  - **rtn** is declared as **final**