PROGRAMMING

Lecture 7

Sushil Paudel


EVERY GROUP PROJECT
DOES 99% OF THE WORK
HAS NO IDEA WHAT'S GOING ON THE WHOLE TIME
SAYS HE'S GOING TO HELP BUT HE'S NOT
DISAPPEAR AT THE VERY BEGINNING AND DOESN'T SHOW UP AGAIN TIL THE VERY END

PREVIOUS TOPIC

- Inheritance
- Type of inheritance
- Method overriding

TODAY'S TOPIC

- Encapsulation
- Array
- Class Diagram

ENCAPSULATION

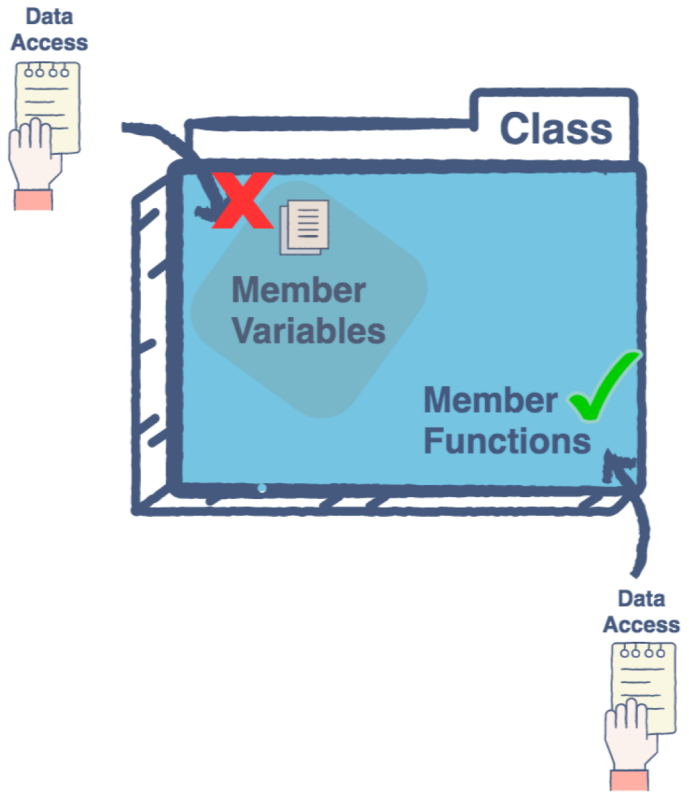
- **Encapsulation** is one of the four fundamental OOP concepts. The other three are inheritance, polymorphism, and abstraction.
- Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit.
- In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class.
- Therefore, it is also known as **data hiding**.

ENCAPSULATION

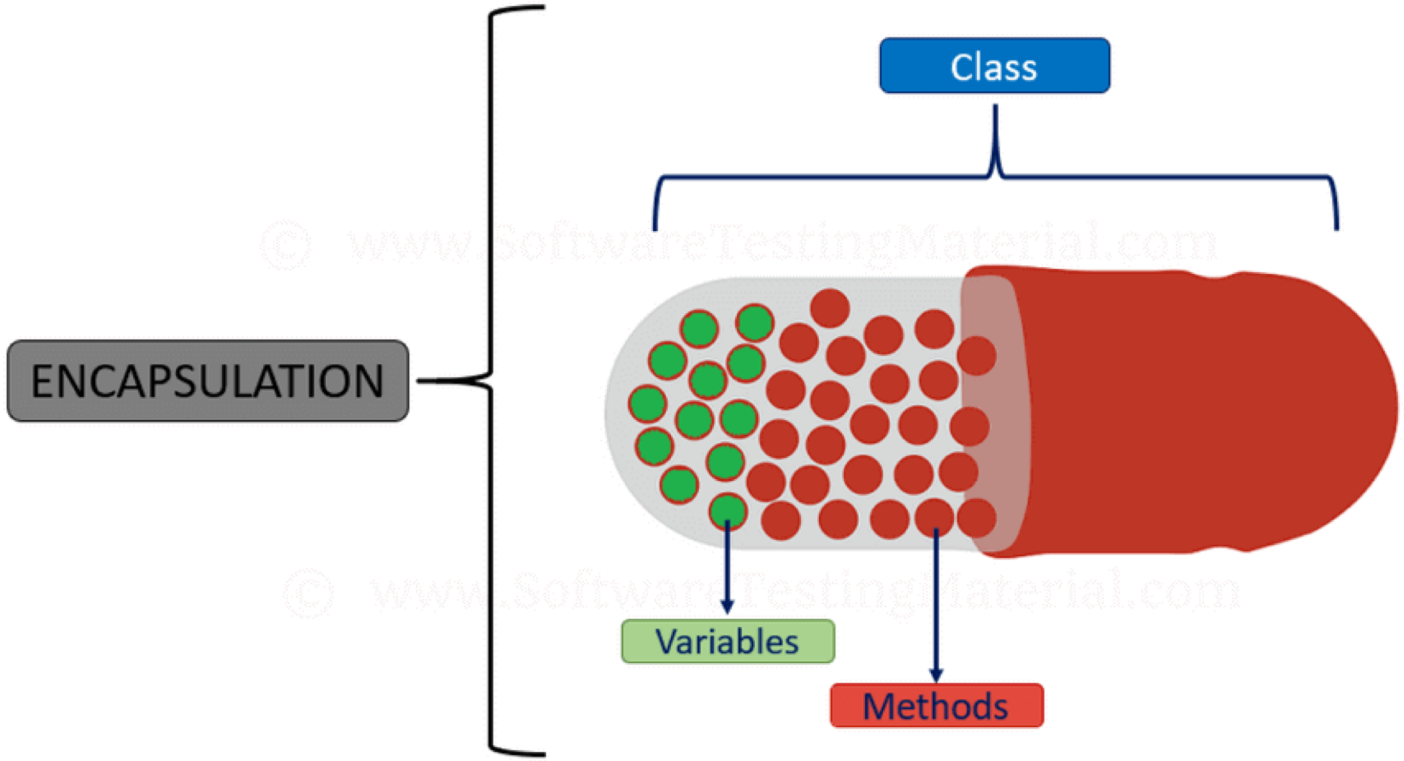
- **Encapsulation** is one of the four fundamental OOP concepts. The other three are inheritance, polymorphism, and abstraction.
- Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit.
- In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class.
- Therefore, it is also known as **data hiding**.

HOW TO ACHIEVE ENCAPSULATION IN JAVA

- Declare the variables of a class as private.
- Provide public setter(mutator) and getter (accessor) methods to modify and view the variables values.



ENCAPSULATION



EXAMPLE

```
public class Person {
  private String name; // private = restricted access

  // Getter
  public String getName() {
    return name;
  }

  // Setter
  public void setName(String name) {
    this.name = name;
  }
}
```

EXAMPLE

```
public class MyClass {

  public static void main(String[] args) {

    Person myObj = new Person();
    myObj.name = "John";  // error
    System.out.println(myObj.name); // error
  }
}
```

EXAMPLE

```
public class MyClass {
  public static void main(String[] args) {
    Person myObj = new Person();
    myObj.setName("John"); // Set the value of the name variable to "John"
    System.out.println(myObj.getName());
  }
}
```

OUTPUT

John

NAMING CONVENTION

Rule	Example

Properties are private.	`private int age;`
	`private boolean happy;`
	`private int numberOfChildren;`

Getter methods begin with *is* if the property is boolean.	`public boolean isHappy() {`
	`  return happy;`
	`}`

Setter methods begin with *set*.	`public void setAge(int newAge) {`
	`  age = newAge;`
	`}`

The method name must have a prefix of *set*/*get*/*is*, followed by the first letter of the property in uppercase, followed by the rest of the property name	`public int getNumberOfChildren() {`
	`  return numberOfChildren;`
	`}`

EXAMPLE

```
public class EncapsulationExample {
   private String name;
   private int id;

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public int getId() {
      return id;
   }

   public void setId(int id) {
      this.id=id;
   }
}
```

ARRAY

- Suppose you need to store 100 numbers we would create the store the data as below:
  - int number1 = 11;
  - int number2 = 22;

  ….

  - int number100 = 1000;

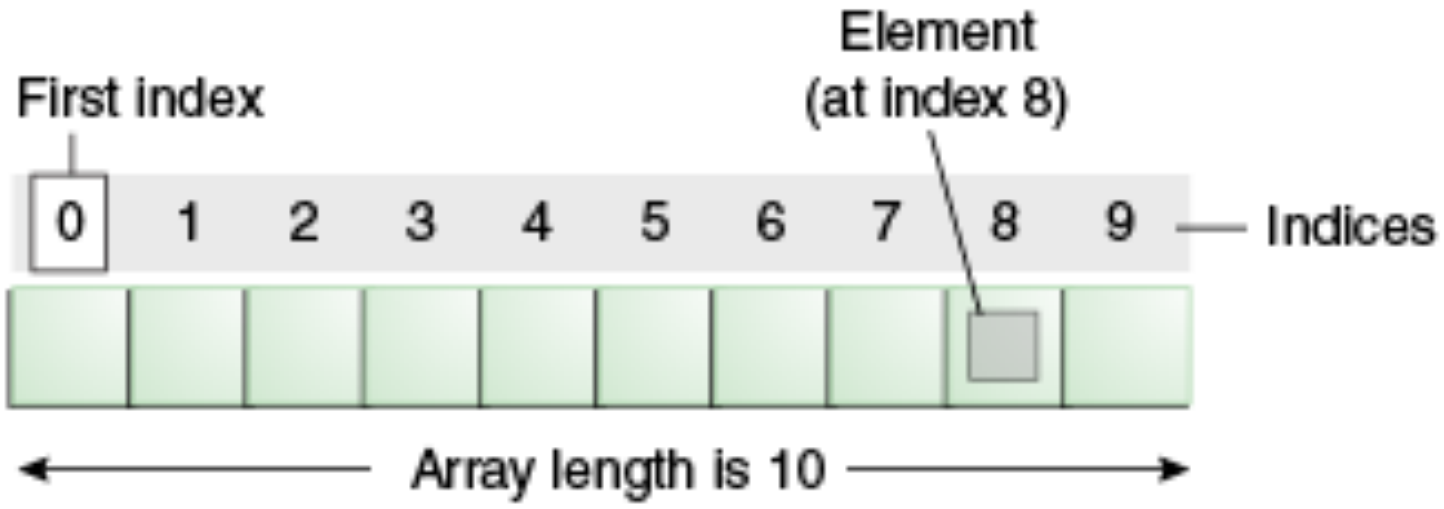
- What if you need to store thousands of data? Would you create 1000 variables?
- What if you need to store millions of records?

ARRAY

- Instead of declaring individual variables like number0, number1, ..., and number100, you declare one array variable such as **numbers** and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables.
- In this way, we don't have to create separate variables for separate data.
- Only one variable is sufficient to store the data.
- This is possible because of the concept called Array.

ARRAY

- Java provides a data structure, the **array**, which stores a fixed-size sequential collection of elements of the same type.
- Array in java is index based, first element of the array is stored at 0 index and nth element is stored at n-1 index.



ADVANTAGES & DISADVANTAGES OF ARRAY

Advantage of Java Array

- **Code Optimization:** It makes the code optimized, we can retrieve or sort the data easily.
- **Random access:** We can get any data located at any index position.

Disadvantage of Java Array

- **Size Limit:** We can store only fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in java.

TYPES OF ARRAY

- Single Dimensional Array
- Multidimensional Array

SYNTAX

```
dataType[] var; // Declaration

dataType var[]; // Declaration

dataType[] var = new dataType[arraySize]; // instantiation

// Declaration, instantiation and initialization
dataType[] var = {value0, value1, ..., valuek};
```

EXAMPLE

```
public static void main(String[] args) {
        int[] a = new int[5];// declaration and instantiation
        a[0] = 10;// initialization
        a[1] = 20;
        a[2] = 70;
        a[3] = 40;
        a[4] = 50;


        // Loop in array
        for (int i = 0; i < a.length; i++) {// length is the property of array
                System.out.println(a[i]); // printing array a
        }

}
```

OUTPUT

```
10
20
70
40
50
```

EXAMPLE

```
public static void main(String[] args) {


        double[] b = {10.1, 20.2, 30.3};//declaration, instantiation and initialization


        // Loop in array
        for (int i = 0; i < b.length; i++) {
                System.out.println(b[i]); // printing array b
        }
}
```

OUTPUT

```
10.1
20.2
30.3
```

FOR EACH LOOP

```
for(data_type variable : array){
    //body of the loop
}
```

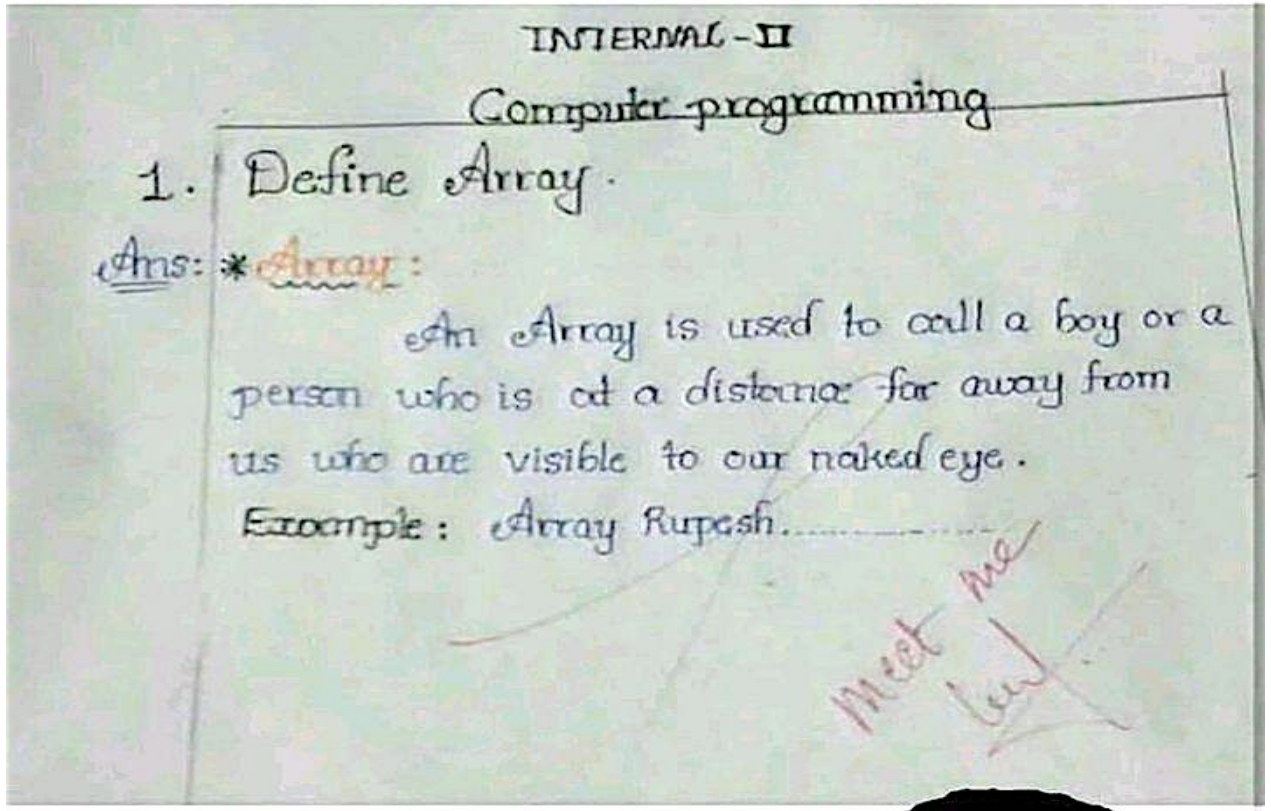
FOR EACH LOOP

```
class ArrayExample{
        public static void main(String args[]){
              int[] arr={1, 2, 3, 4, 5};

              //printing array using for-each loop
               for (int  i : arr)
                     System.out.println(i);
              }
       }
}
```

OUTPUT

1
2
3
4
5

ARRAY...


INTERNAL - II
Computer programming
1. Define Array.
Ans: * Array :
An Array is used to call a boy or a person who is at a distance far away from us who are visible to our naked eye.
Example : Array Rupesh.....
meet me here


This is what happens when you bunk classes

CLASS DIAGRAM

- Suppose you have to design a system.
- Before implementing a bunch of classes, you'll want to have a conceptual understanding of the system—that is, what classes do I need? What functionality and information will these classes have? How do they interact with one another? Who can see these classes? And so on.
- That's where class diagrams come in. Class diagrams are a neat way of visualizing the classes in your system *before* you actually start coding them up. They're a static representation of your system structure.

BASIC COMPONENTS OF CLASS DIAGRAM

The standard class diagram is composed of three sections:

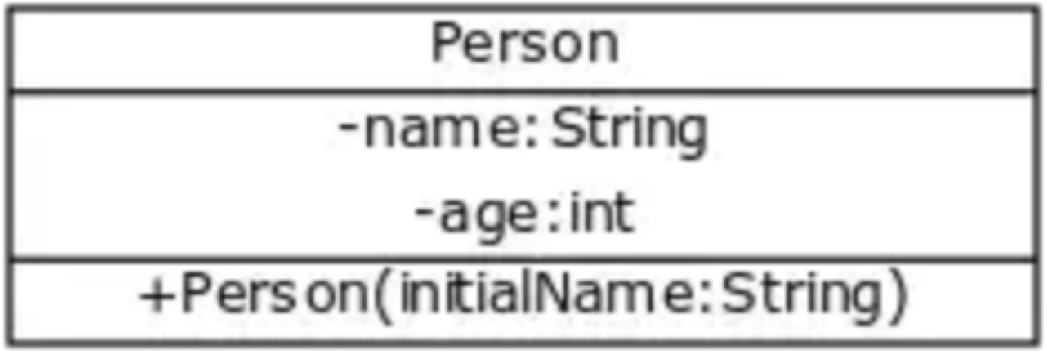
- **Upper section:** Contains the name of the class.
- **Middle section:** Contains the attributes/properties/state of the class.
- **Bottom section:** Includes class operations (methods). Displayed in list format, each operation takes up its own line.

ACCESS MODIFIER

- All classes have different access levels depending on the access modifier (visibility). Here are the access levels with their corresponding symbols:
- Public (+)
- Private (-)
- Protected (#)
- Package (~)
- Static (underlined)

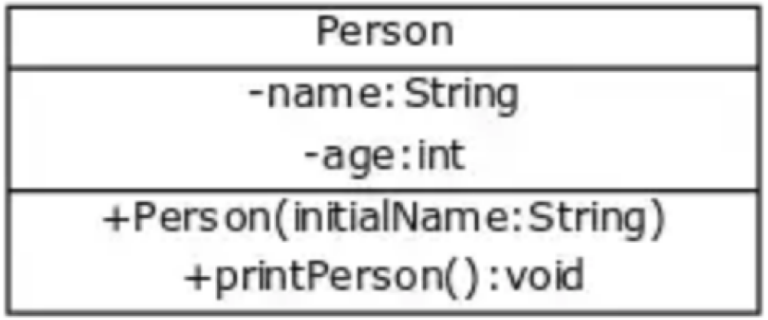
EXAMPLE

```
public class Person {
        private String name;
        private int age;

        public Person(String initialName) {
                this.name = initialName;
                this.age = 0;
        }
 }
```

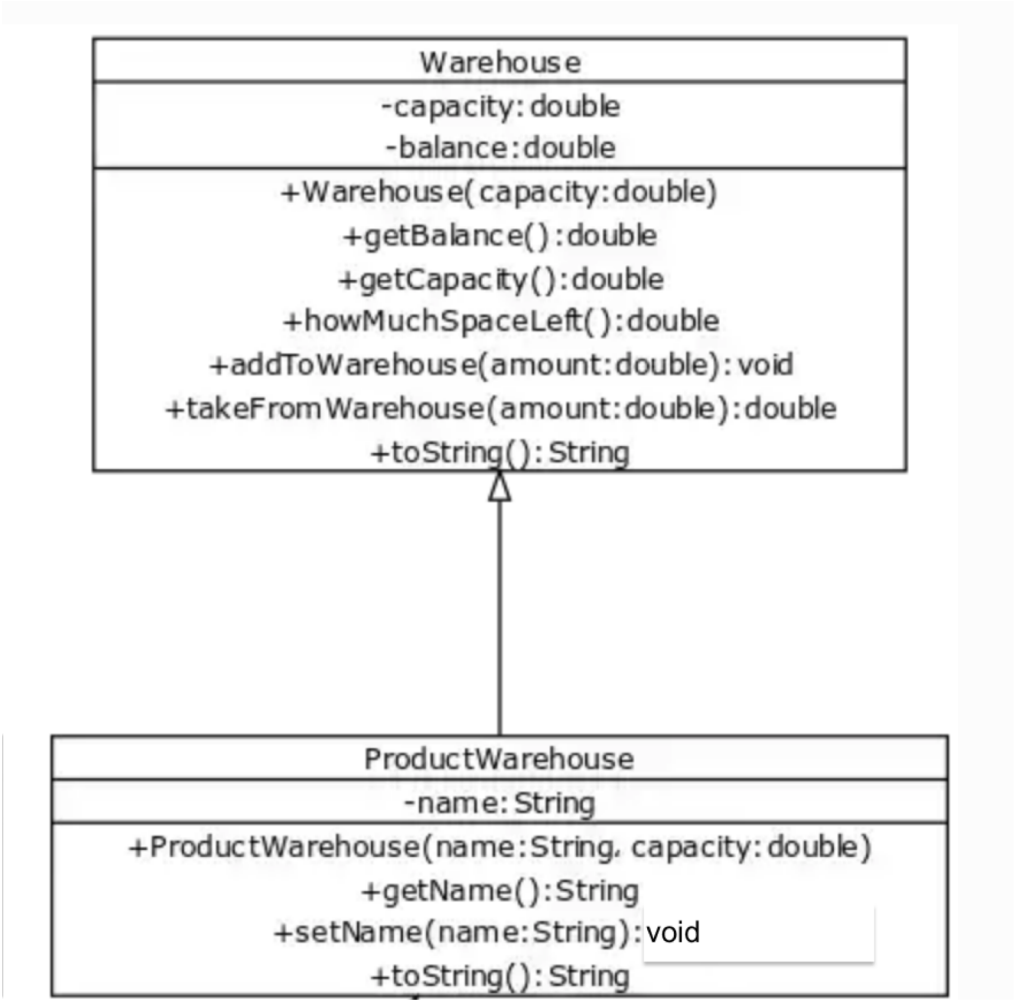


EXAMPLE

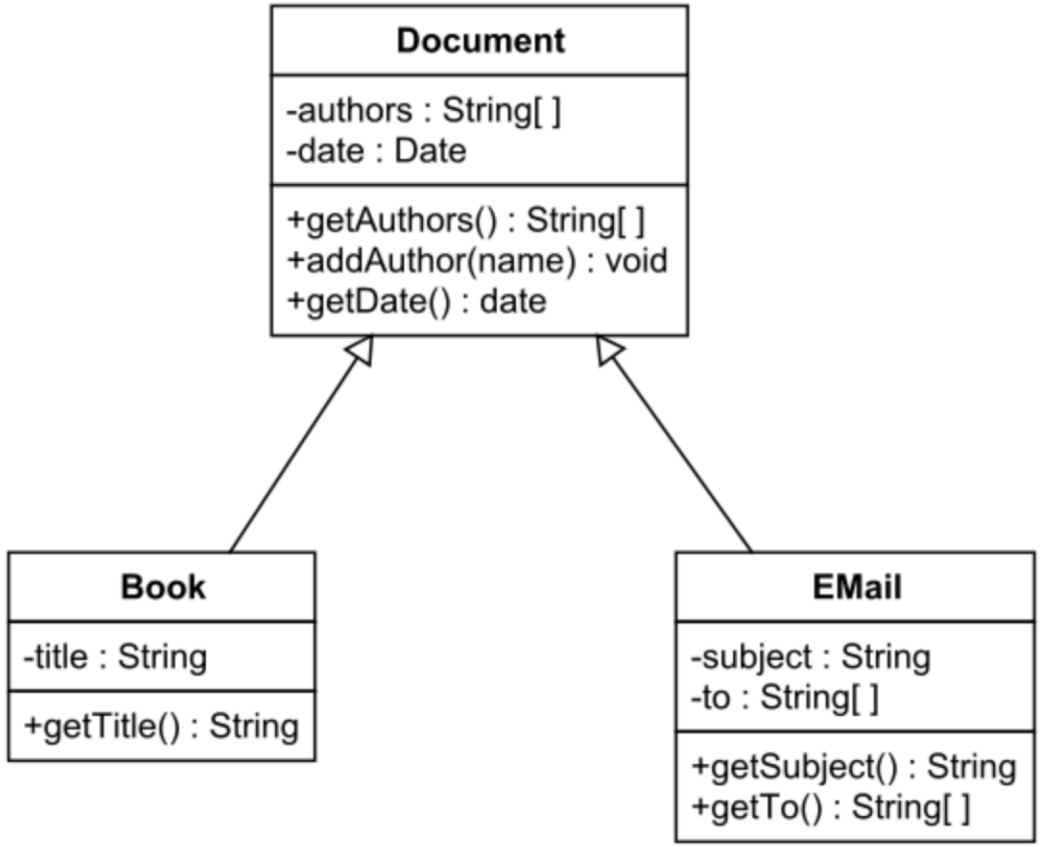
```java
public class Person {
        private String name;
        private int age;

        public Person(String initialName) {
                this.name = initialName;
                this.age = 0;
        }

        public void printPerson() {
                System.out.println(this.name + ", age " + this.age + " years");
        }
 }
```



EXAMPLE


Warehouse
-capacity: double
-balance: double
+Warehouse(capacity: double)
+getBalance( ): double
+getCapacity( ): double
+howMuchSpaceLeft( ): double
+addToWarehouse(amount: double): void
+takeFromWarehouse(amount: double): double
+toString( ): String
ProductWarehouse
-name: String
+ProductWarehouse(name: String, capacity: double)
+getName( ): String
+setName(name: String): void
+toString( ): String

EXAMPLE


Document
-authors : String[ ]
-date : Date
+getAuthors() : String[ ]
+addAuthor(name) : void
+getDate() : date
Book
-title : String
+getTitle() : String
EMail
-subject : String
-to : String[ ]
+getSubject() : String
+getTo() : String[ ]

THANK YOU!

Any questions?