In, Out, and InOut Parameters in Class Diagrams

In **UML class diagrams**, **method parameters** can have **directionality**:

1. **In (Input Parameter)** → The method **receives** a value but **does not modify it**.
2. **Out (Output Parameter)** → The method **modifies and returns** the value to the caller.
3. **InOut (Input and Output Parameter)** → The method **both receives and modifies** the value.

These are **important when modeling method behavior** because they indicate how data flows between objects.

1. In Parameter (Input)

- The value is **passed to the method** but **not modified** inside the method.
- Default behavior in Java (since Java uses **pass-by-value** for primitive types and **pass-by-reference** for objects).

2.Out Parameter (Output) – Returns a New Value

- The **method does not modify** the passed argument directly.
- Instead, it **returns a new value**, which the caller must **store explicitly**.

3.InOut Parameter – Modifies an Existing Object

- The method **modifies the object that was passed**.
- In Java, this works **only with objects** because objects are passed **by reference**.

1️⃣One-to-One Association

```
class Passport {
    String passportNumber;
    Passport(String passportNumber) {
        this.passportNumber = passportNumber;
    }
}
class Person {
    String name;
    Passport passport;  // Association (Person has a Passport)
    Person(String name, Passport passport) {
        this.name = name;
        this.passport = passport;
    }
    void showDetails() {
        System.out.println(name + " has passport number: " + passport.passportNumber);
    }
}
public class Main {
    public static void main(String[] args) {
        Passport p1 = new Passport("A1234567");
        Person person1 = new Person("John", p1);
        person1.showDetails();  // Output: John has passport number: A1234567
    }   }
```

2️⃣ One-to-Many Association

```
import java.util.List;
import java.util.ArrayList;

class Book {
    String title;
    Book(String title) {
        this.title = title;
    }
}

class Library {
    String name;
    List<Book> books = new ArrayList<>(); // One Library has multiple Books

    Library(String name) {
        this.name = name;
    }

    void addBook(Book book) {
        books.add(book);
    }
    void showBooks() {
        System.out.println("Books in " + name + ":");
        for (Book b : books) {
            System.out.println("- " + b.title);
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Library library = new Library("City Library");
        Book book1 = new Book("Java Programming");
        Book book2 = new Book("Data Structures");
        library.addBook(book1);
        library.addBook(book2);
        library.showBooks();
    }
}
```