PRACTICUM REPORT MODULE 4

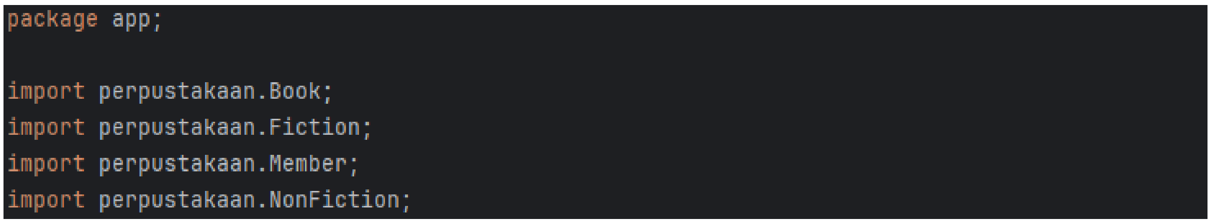
“Codelab”

Name: Ovan Keyva Kusumadewa

NIM: 202410370110506

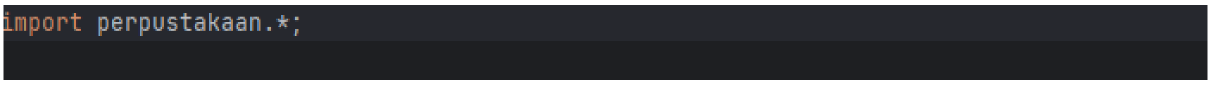
Class: 2A/Informatics

Importing packages



```java
package app;

import perpustakaan.Book;
import perpustakaan.Fiction;
import perpustakaan.Member;
import perpustakaan.NonFiction;
```

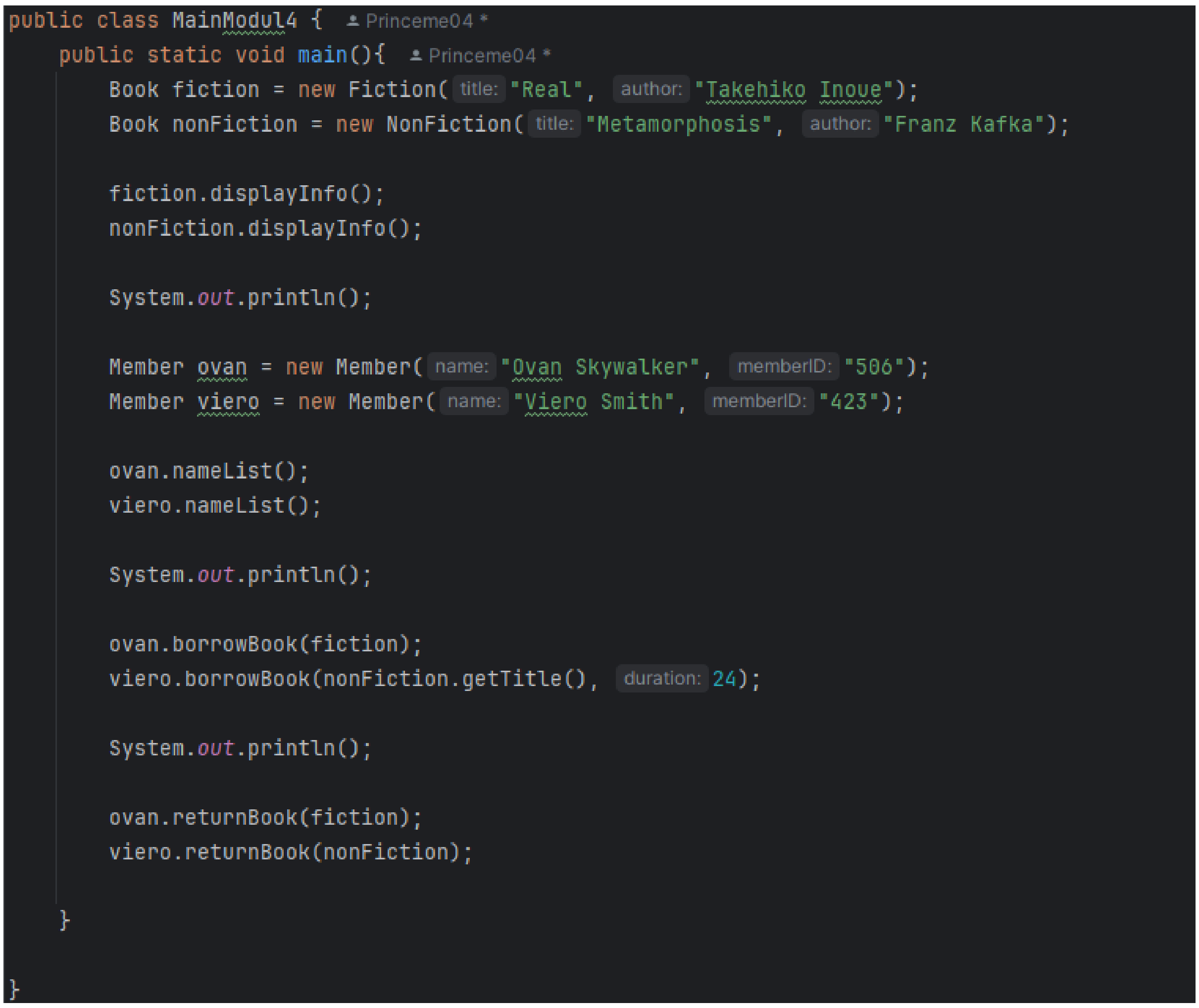
First step you need to import packages, why?, to gain access to each files needed. For example: perpustakaan package contain Book, Fiction, Member, NonFiction(there is one more files, BookLoan, but it doesn't need to be imported since it already implemented by Member) or you can just write



```java
import perpustakaan.*;
```

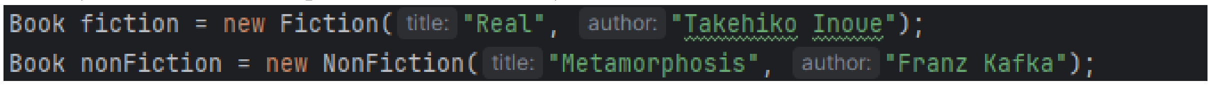
So You don't need to import files one by one.

1.1 MainModul4.java



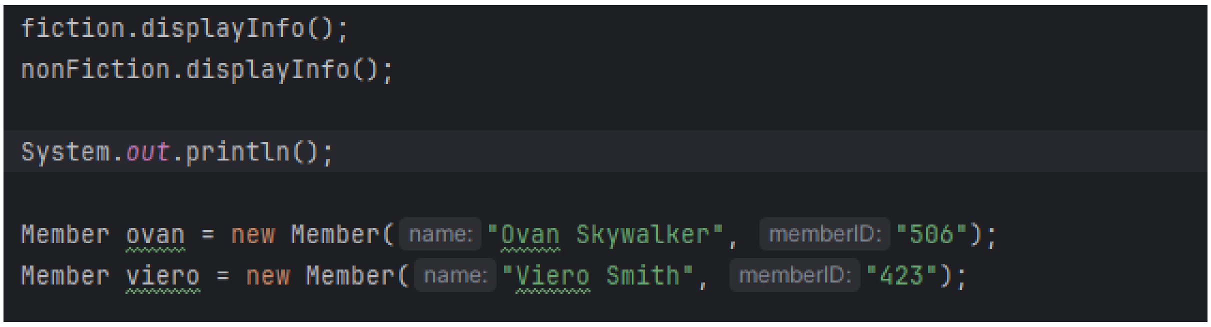
```java
public class MainModul4 {  Princeme04 *
    public static void main(){  Princeme04 *
        Book fiction = new Fiction( title: "Real",  author: "Takehiko Inoue");
        Book nonFiction = new NonFiction( title: "Metamorphosis",  author: "Franz Kafka");

        fiction.displayInfo();
        nonFiction.displayInfo();

        System.out.println();

        Member ovan = new Member( name: "Ovan Skywalker",  memberID: "506");
        Member viero = new Member( name: "Viero Smith",  memberID: "423");

        ovan.nameList();
        viero.nameList();

        System.out.println();

        ovan.borrowBook(fiction);
        viero.borrowBook(nonFiction.getTitle(),  duration: 24);

        System.out.println();

        ovan.returnBook(fiction);
        viero.returnBook(nonFiction);

    }

}
```

A class named MainModule4 as a main file start with create new object from Book.java which I'll explain later, new object are created with,



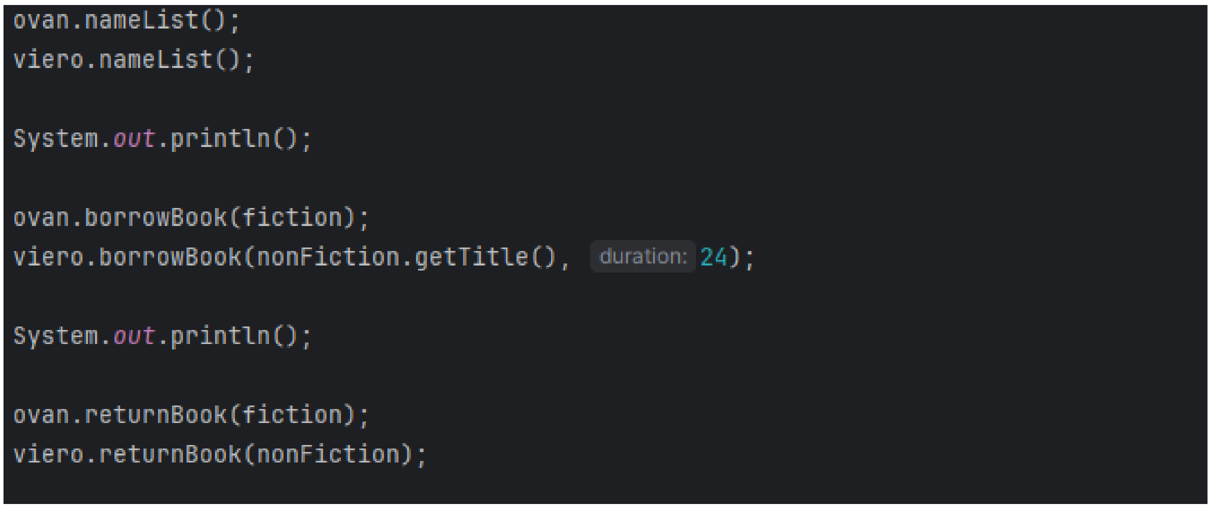
```java
Book fiction = new Fiction( title: "Real",  author: "Takehiko Inoue");
Book nonFiction = new NonFiction( title: "Metamorphosis",  author: "Franz Kafka");
```

also the parameter contain String for title, and author.



```
fiction.displayInfo();
nonFiction.displayInfo();

System.out.println();

Member ovan = new Member( name: "Ovan Skywalker",  memberID: "506");
Member viero = new Member( name: "Viero Smith",  memberID: "423");
```

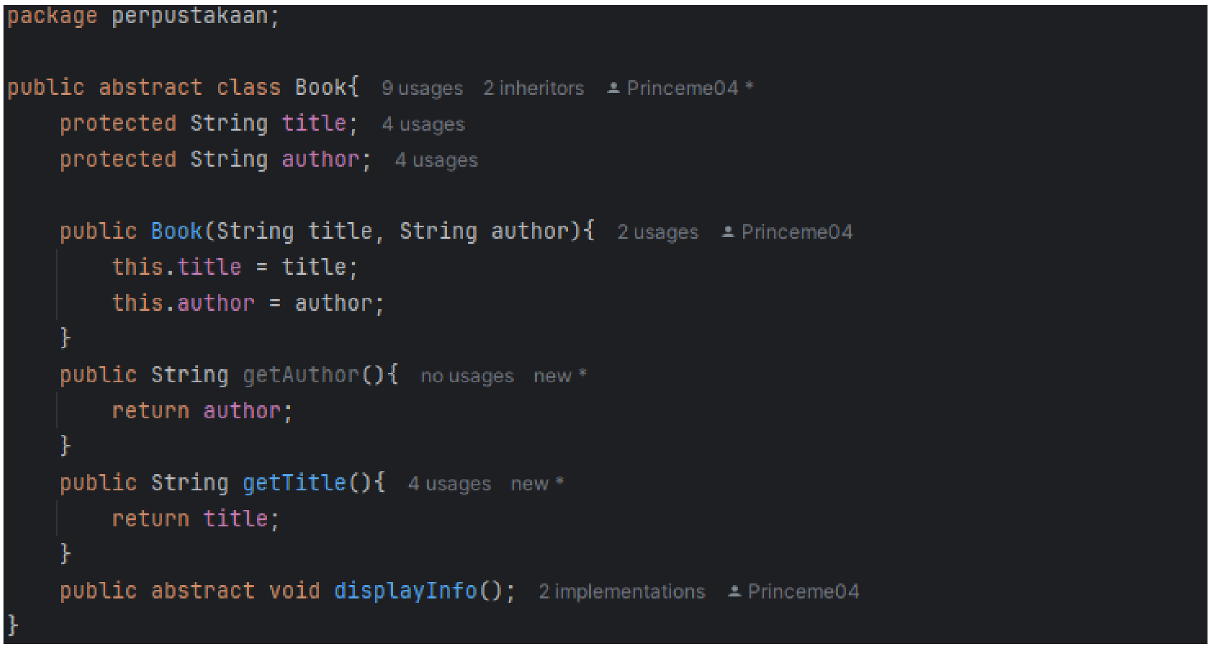
displayInfo(); is for print status of the book which the method written in Book.java, and the output depends on the object created fiction and nonfiction.
New object created for Member.java, which contain member name and member ID,
Member ovan = new Member(“Ovan Skywalker”, “506”);
Member viero = new Member(“Viero Smith”, “423”);
Both parameter contains member name and member ID.



```
ovan.nameList();
viero.nameList();

System.out.println();

ovan.borrowBook(fiction);
viero.borrowBook(nonFiction.getTitle(),  duration: 24);

System.out.println();

ovan.returnBook(fiction);
viero.returnBook(nonFiction);
```

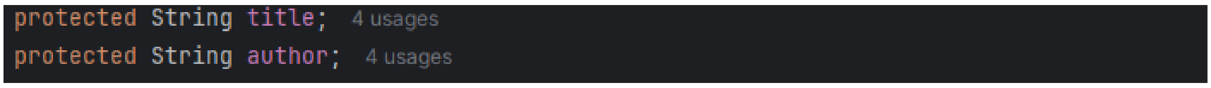
This part only contains object to call method from files mentioned, and “inject” parameter according to the object mentioned.

1.2 Book.java



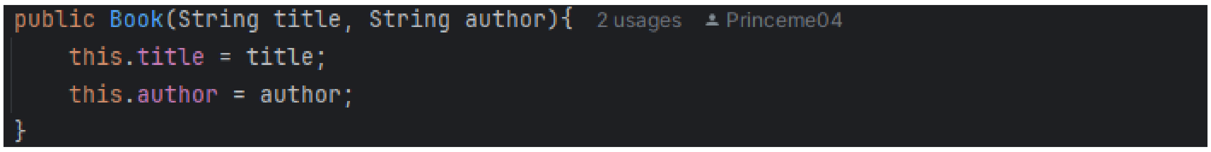
```
package perpustakaan;

public abstract class Book{  9 usages  2 inheritors  Princeme04 *
    protected String title;  4 usages
    protected String author;  4 usages

    public Book(String title, String author){  2 usages  Princeme04
        this.title = title;
        this.author = author;
    }
    public String getAuthor(){  no usages  new *
        return author;
    }
    public String getTitle(){  4 usages  new *
        return title;
    }
    public abstract void displayInfo();  2 implementations  Princeme04
}
```

For Book.java contains abstract class which means restricted class that cannot be used to create objects (to access it, it must be inherited from another class).



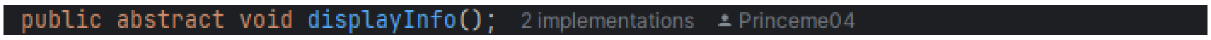
```
protected String title;  4 usages
protected String author;  4 usages
```

protected keyword is added to make sure subclass can accessed it, since abstract classes need subclasses to work. Declare data types String named title, and author. There is also constructor for it,



```
public Book(String title, String author){  2 usages  Princeme04
    this.title = title;
    this.author = author;
}
```

constructor is used to initialize object and called when object of a class created, for example in this code, constructer used in getTitle and getAuthor method(but in this case getAuthor is not used).



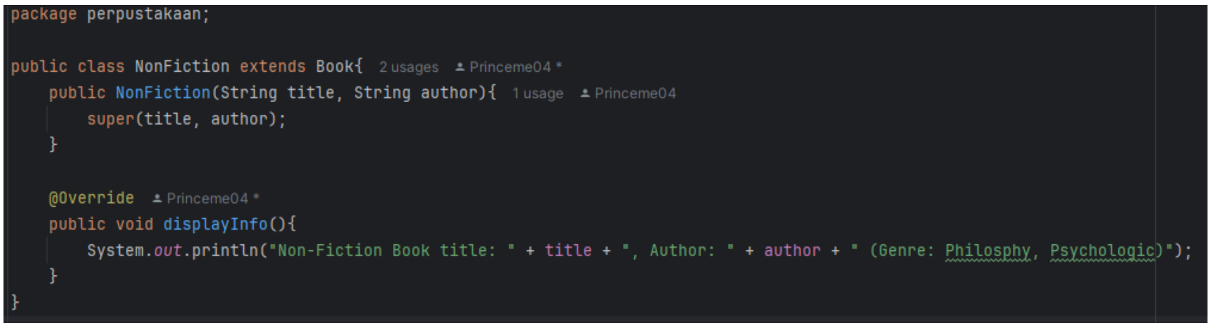
```
public abstract void displayInfo();  2 implementations  Princeme04
```

Abstract method is used since the method does not have body.

1.3 Fiction.java & NonFiction.java

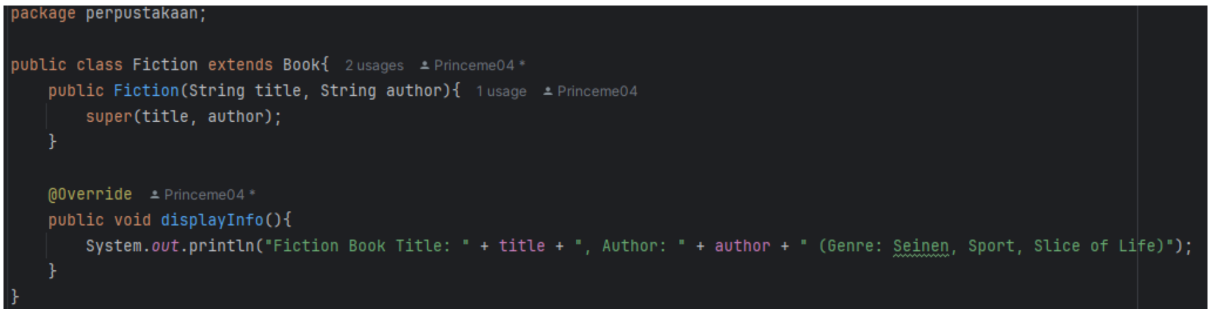
I put them in the part cuz they're pretty similar, the only difference is the title and author "injected" from the MainModul4.java files.

NonFiction.java



```
package perpustakaan;

public class NonFiction extends Book{  2 usages  Princeme04 *
    public NonFiction(String title, String author){  1 usage  Princeme04
        super(title, author);
    }

    @Override  Princeme04 *
    public void displayInfo(){
        System.out.println("Non-Fiction Book title: " + title + ", Author: " + author + " (Genre: Philosphy, Psychologic)");
    }
}
```

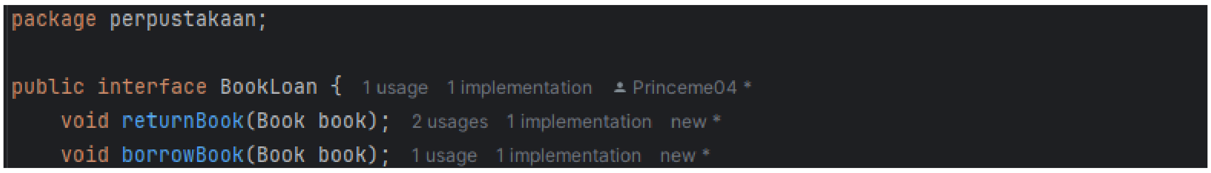
Fiction.java



```java
package perpustakaan;

public class Fiction extends Book{  2 usages  Princeme04 *
    public Fiction(String title, String author){  1 usage  Princeme04
        super(title, author);
    }

    @Override  Princeme04 *
    public void displayInfo(){
        System.out.println("Fiction Book Title: " + title + ", Author: " + author + " (Genre: Seinen, Sport, Slice of Life)");
    }
}
```

super keyword means to interact withh superclasses, since Fiction & NonFiction files are subclasses. And also @Override is added to overide displayInfo() methods to be specific which displayInfo() wanted to be ran.

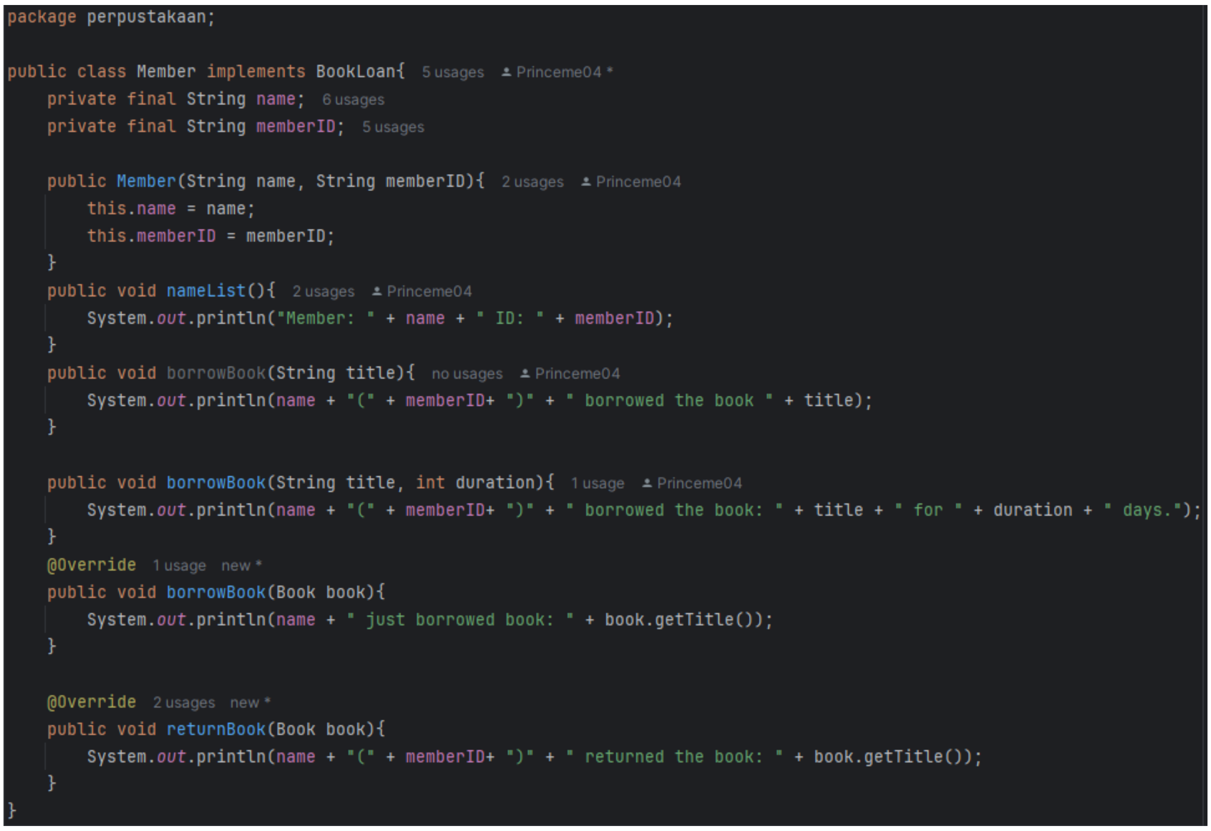
1.4 BookLoan.java



```java
package perpustakaan;

public interface BookLoan {  1 usage  1 implementation  Princeme04 *
    void returnBook(Book book);  2 usages  1 implementation  new *
    void borrowBook(Book book);  1 usage  1 implementation  new *
```

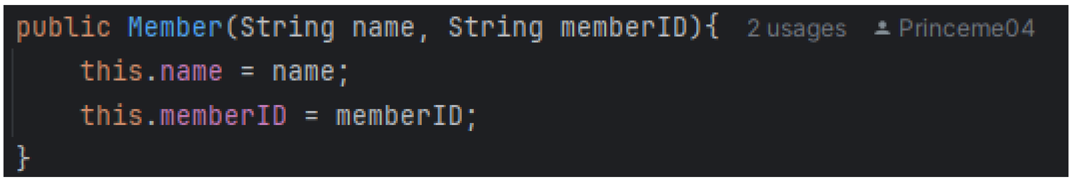
Interface classes contain method without body, and Interface classes needed to be “implemented” it’s kinda like inheritance but using keyword implements instead of extends. And the parameter contain (Book book) “book”means a variable representing in the object passed in from “Book” classes

1.5 Member.java



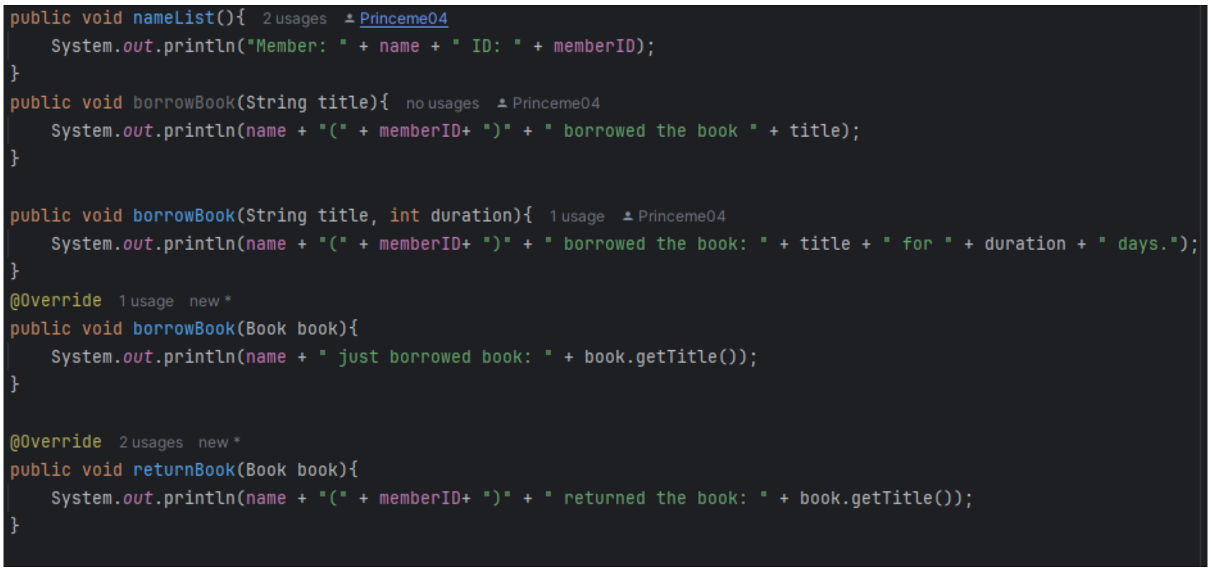
```java
package perpustakaan;

public class Member implements BookLoan{  5 usages  Princeme04 *
    private final String name;  6 usages
    private final String memberID;  5 usages

    public Member(String name, String memberID){  2 usages  Princeme04
        this.name = name;
        this.memberID = memberID;
    }
    public void nameList(){  2 usages  Princeme04
        System.out.println("Member: " + name + " ID: " + memberID);
    }
    public void borrowBook(String title){  no usages  Princeme04
        System.out.println(name + "(" + memberID+ ")" + " borrowed the book " + title);
    }

    public void borrowBook(String title, int duration){  1 usage  Princeme04
        System.out.println(name + "(" + memberID+ ")" + " borrowed the book: " + title + " for " + duration + " days.");
    }
    @Override  1 usage  new *
    public void borrowBook(Book book){
        System.out.println(name + " just borrowed book: " + book.getTitle());
    }

    @Override  2 usages  new *
    public void returnBook(Book book){
        System.out.println(name + "(" + memberID+ ")" + " returned the book: " + book.getTitle());
    }
}
```

There are some keyword before data types, private means the variable can only be accessed inside the same class, final mean the variable is defined, unchangeable, or only initialized once it could be from declaration or constructor.



```java
public Member(String name, String memberID){  2 usages  Princeme04
    this.name = name;
    this.memberID = memberID;
}
```

Constructor to name and member ID.



```java
public void nameList(){  2 usages  Princeme04
    System.out.println("Member: " + name + " ID: " + memberID);
}
public void borrowBook(String title){  no usages  Princeme04
    System.out.println(name + "(" + memberID+ ")" + " borrowed the book " + title);
}

public void borrowBook(String title, int duration){  1 usage  Princeme04
    System.out.println(name + "(" + memberID+ ")" + " borrowed the book: " + title + " for " + duration + " days.");
}
@Override  1 usage  new *
public void borrowBook(Book book){
    System.out.println(name + " just borrowed book: " + book.getTitle());
}

@Override  2 usages  new *
public void returnBook(Book book){
    System.out.println(name + "(" + memberID+ ")" + " returned the book: " + book.getTitle());
}
```

Contains many methods of borrow and returning activity, and also include overloading example on borrowBook(); method.