Lab 2: Java Programming Basics

Learning Outcomes

1) Be able to write a Java program and compile it with Gradle

2) Be able to describe the concept interface and inheritance in Java

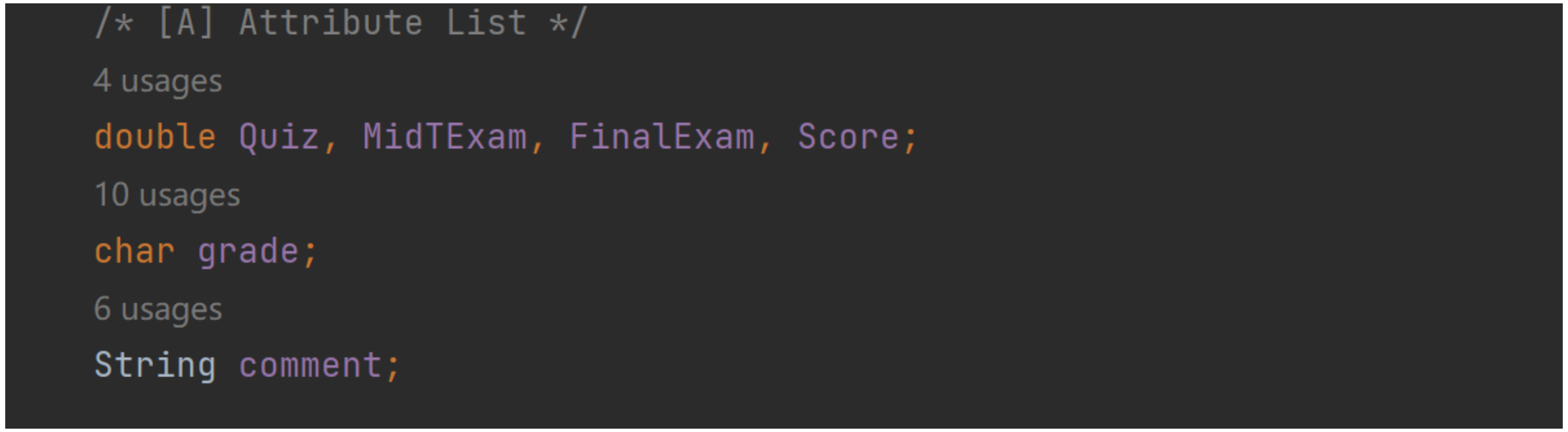
Java Basics

Java Classes/Objects

Java is an object-oriented programming language.

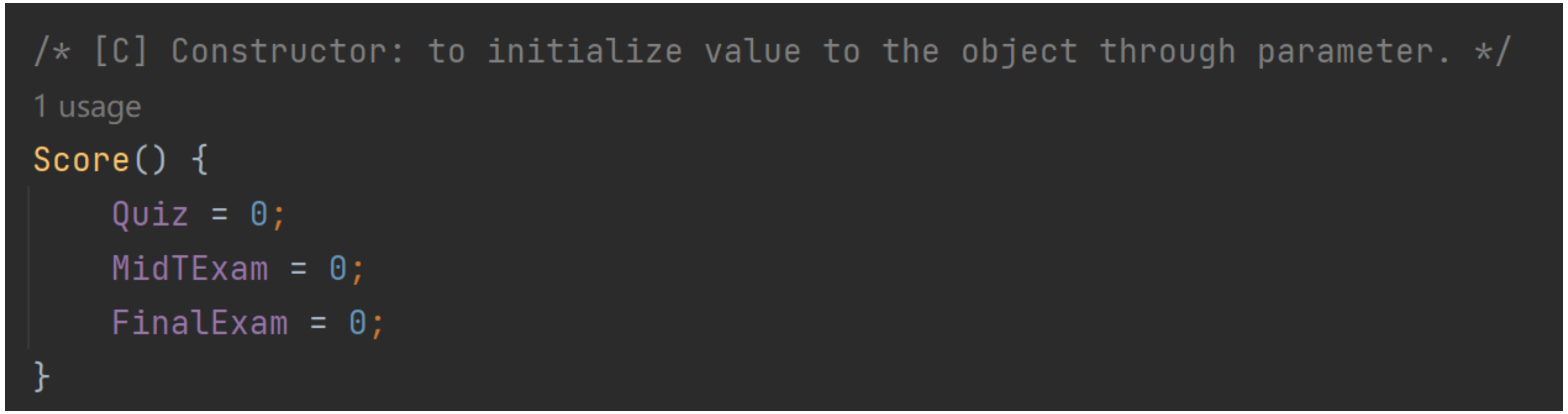
Everything in Java is associated with classes and objects, along with its attributes and methods.

Attribute: An attribute is a variable within a class that describes the characteristics or state of an object. Attributes store data related to an object and can be accessed and modified using accessor (getter) and mutator (setter) methods.



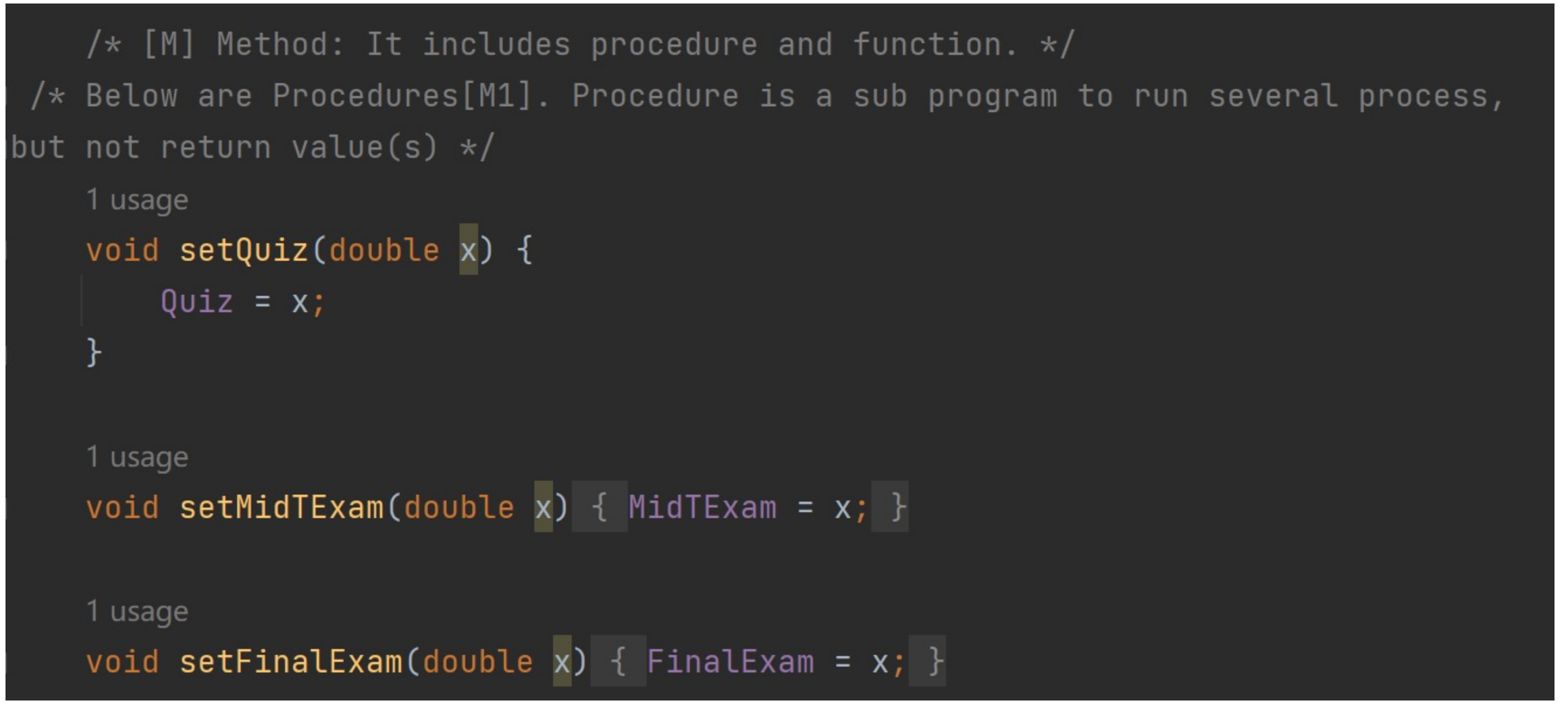
```
/* [A] Attribute List */
4 usages
double Quiz, MidTExam, FinalExam, Score;
10 usages
char grade;
6 usages
String comment;
```

Constructor: A constructor is a special type of method used to create and initialize objects of a class. When a new instance of a class is created, the constructor is automatically called. Its main purpose is to allocate memory for the object and set initial values.

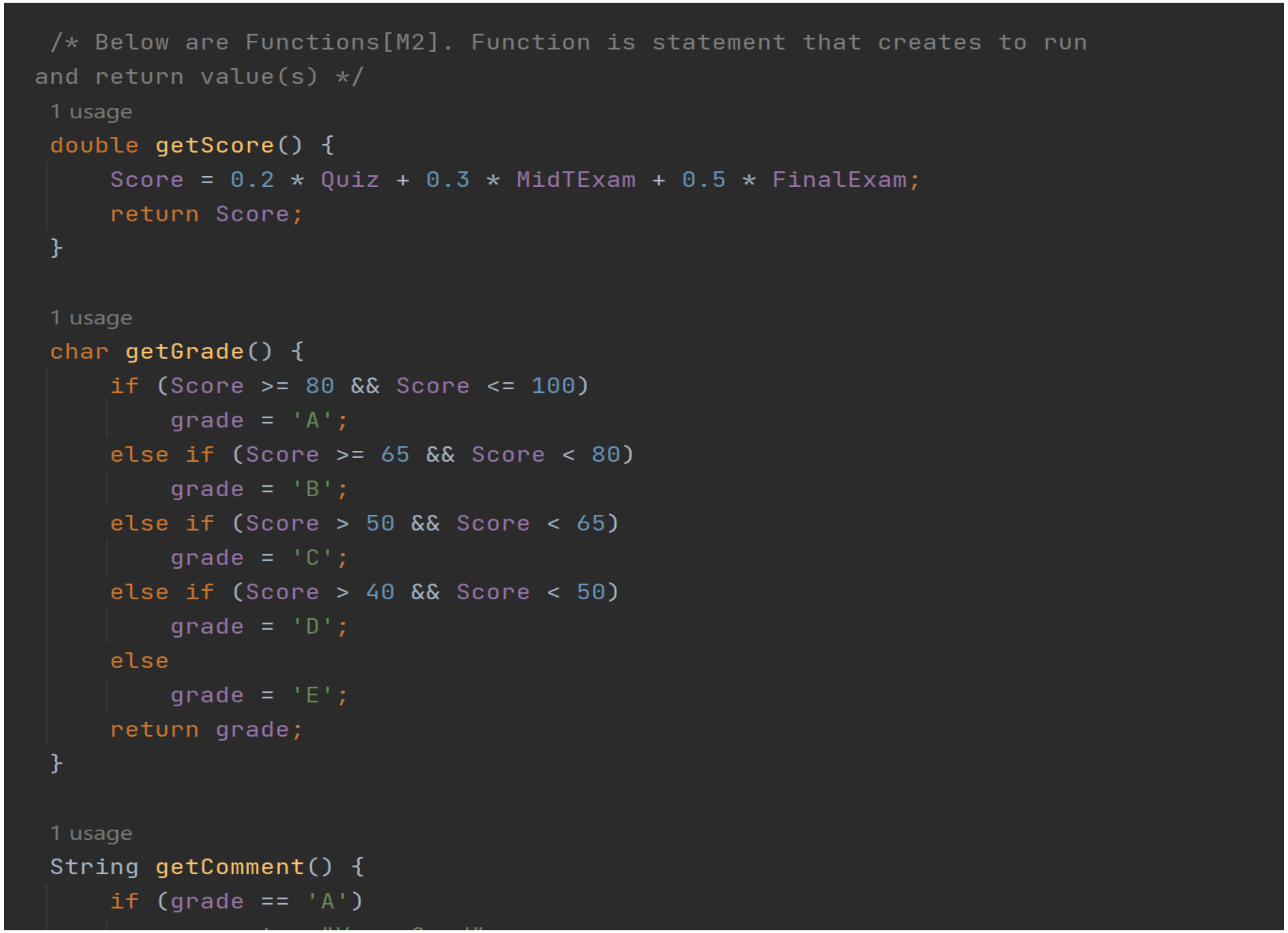


```
/* [C] Constructor: to initialize value to the object through parameter. */
1 usage
Score() {
    Quiz = 0;
    MidTExam = 0;
    FinalExam = 0;
}
```

Method: A method is a function within a class that defines the behavior or functionality of the class. Methods encapsulate a series of operations and can accept parameters and return values. They define the actions that a class can perform



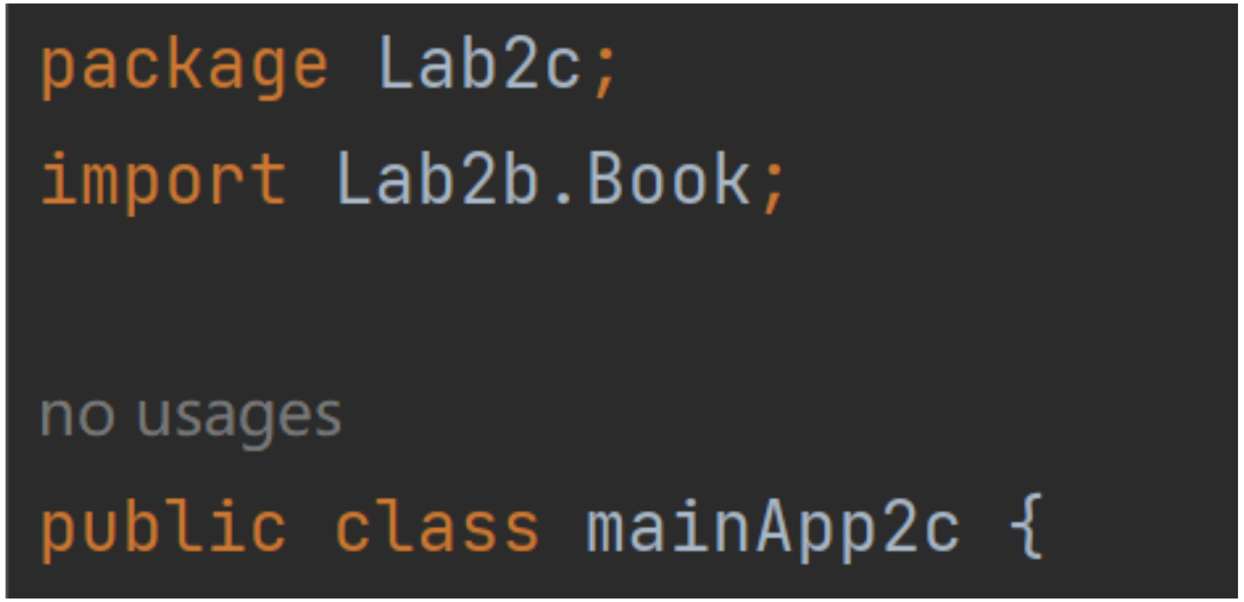
```
    /* [M] Method: It includes procedure and function. */
  /* Below are Procedures[M1]. Procedure is a sub program to run several process,
but not return value(s) */
    1 usage
    void setQuiz(double x) {
        Quiz = x;
    }

    1 usage
    void setMidTExam(double x) { MidTExam = x; }

    1 usage
    void setFinalExam(double x) { FinalExam = x; }
```



```
 /* Below are Functions[M2]. Function is statement that creates to run
and return value(s) */
 1 usage
 double getScore() {
     Score = 0.2 * Quiz + 0.3 * MidTExam + 0.5 * FinalExam;
     return Score;
 }

 1 usage
 char getGrade() {
     if (Score >= 80 && Score <= 100)
         grade = 'A';
     else if (Score >= 65 && Score < 80)
         grade = 'B';
     else if (Score > 50 && Score < 65)
         grade = 'C';
     else if (Score > 40 && Score < 50)
         grade = 'D';
     else
         grade = 'E';
     return grade;
 }

 1 usage
 String getComment() {
     if (grade == 'A')
```

Now you need to think about if we create another package like [Lab2b] under [src.main.java], can we have another [mainApp2b] that can call the [Score] class under [Lab2a]?
Yes, for sure we can work with cross package classes within a project.

But, how?

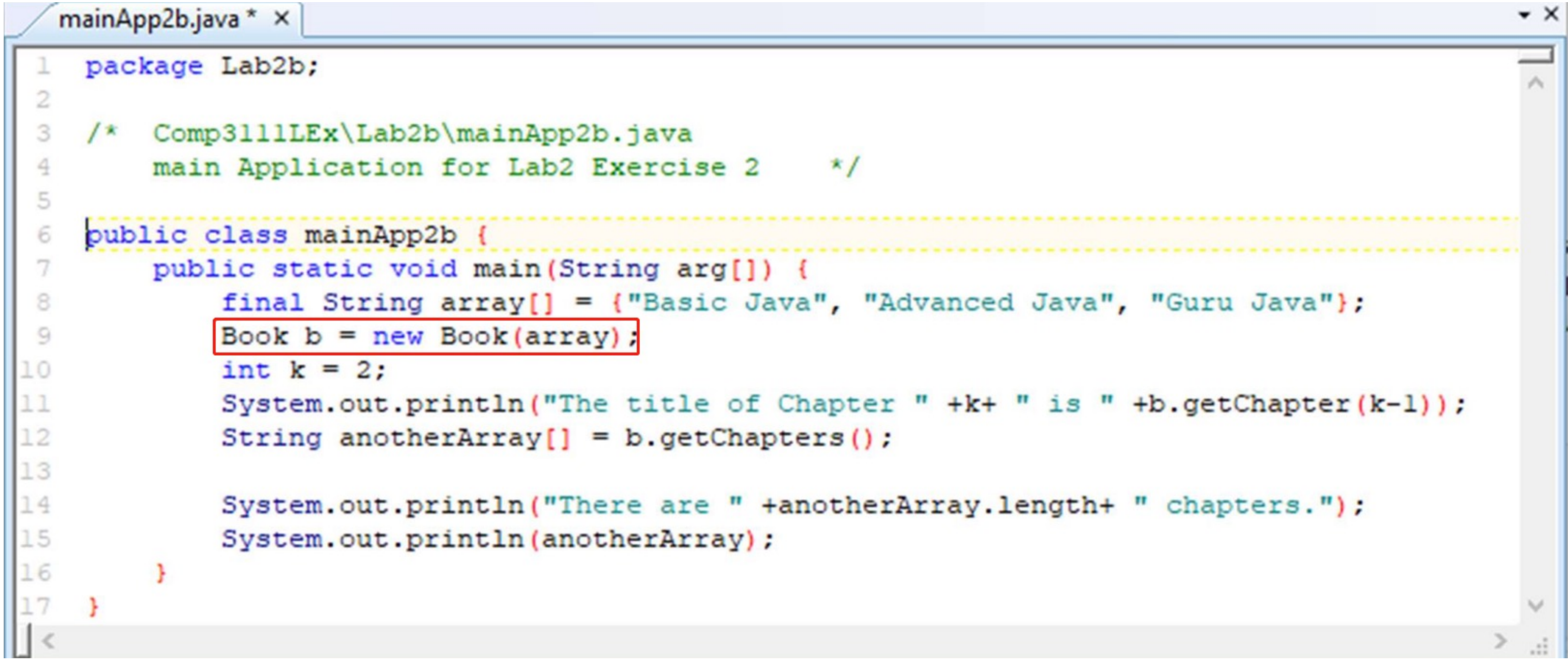


```
package Lab2c;
import Lab2b.Book;


no usages
public class mainApp2c {
```

(This is an example in ex3)

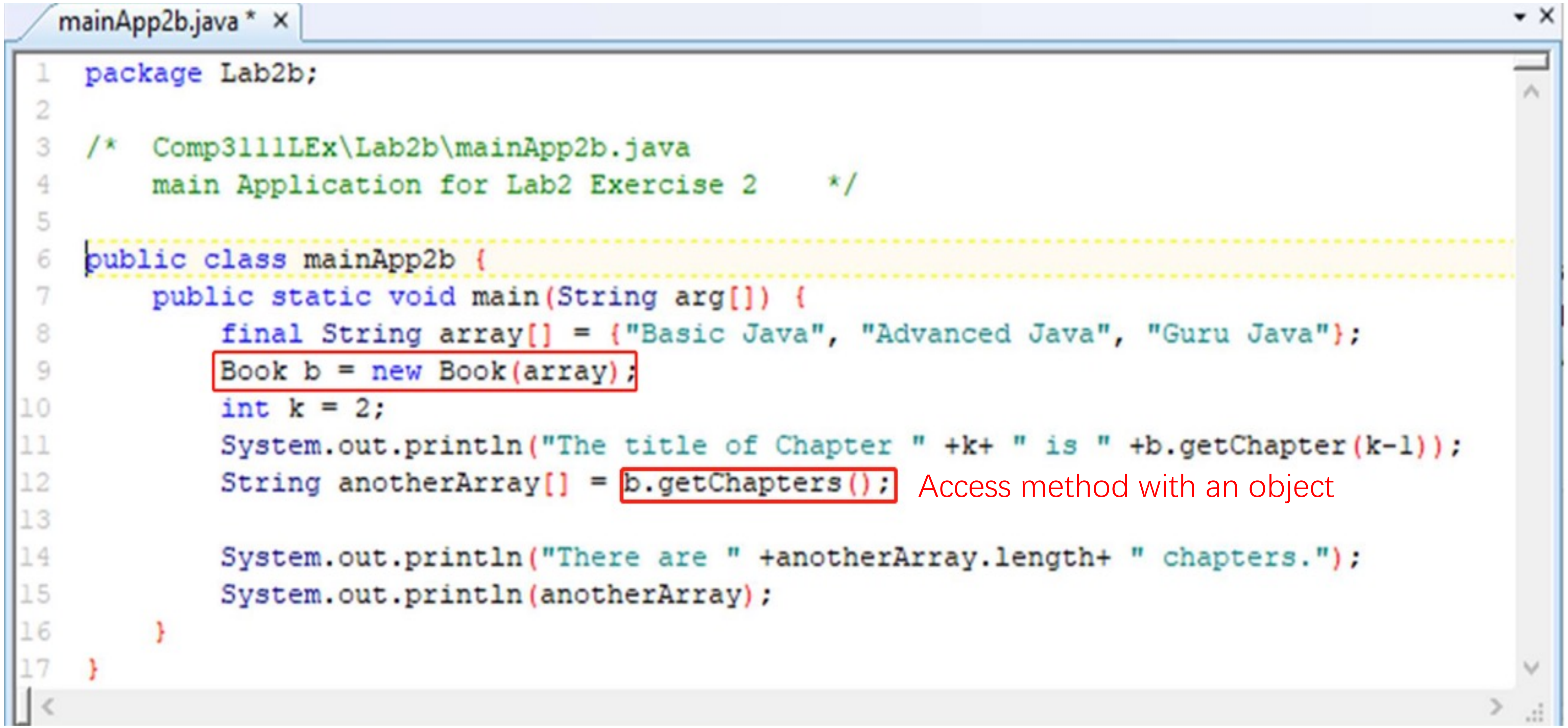
Exercise 2: Fill in the blank to complete an OOP Java application

Create an Object



```java
package Lab2b;

/*  Comp3111LEx\Lab2b\mainApp2b.java
    main Application for Lab2 Exercise 2    */

public class mainApp2b {
    public static void main(String arg[]) {
        final String array[] = {"Basic Java", "Advanced Java", "Guru Java"};
        Book b = new Book(array);
        int k = 2;
        System.out.println("The title of Chapter " +k+ " is " +b.getChapter(k-1));
        String anotherArray[] = b.getChapters();

        System.out.println("There are " +anotherArray.length+ " chapters.");
        System.out.println(anotherArray);
    }
}
```

Access Attributes/Methods With an Object

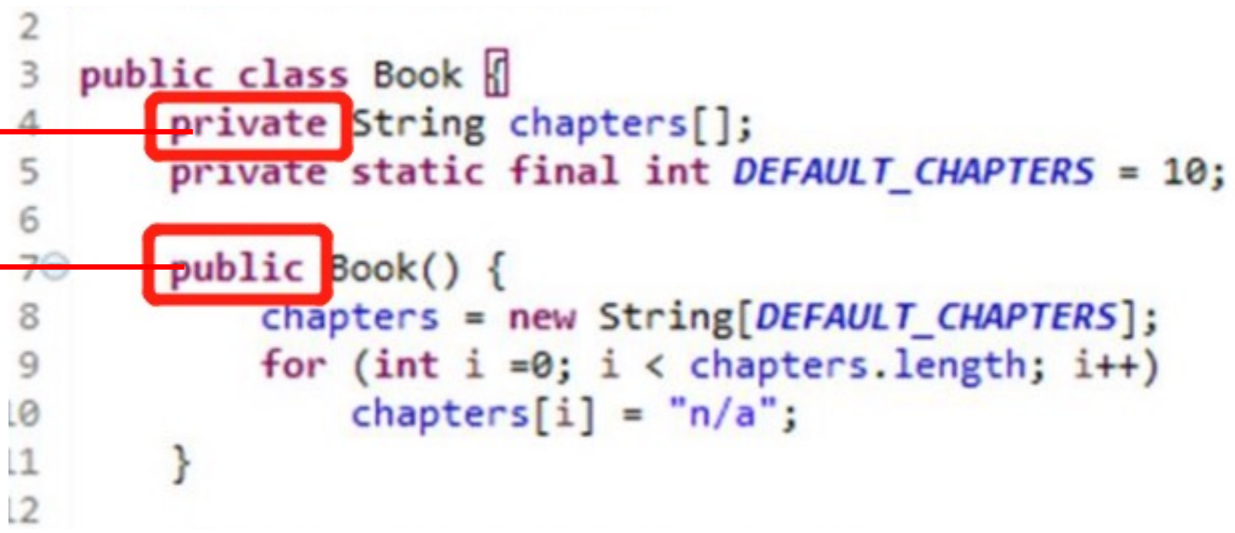


```
package Lab2b;

/*  Comp3111LEx\Lab2b\mainApp2b.java
    main Application for Lab2 Exercise 2    */

public class mainApp2b {
    public static void main(String arg[]) {
        final String array[] = {"Basic Java", "Advanced Java", "Guru Java"};
        Book b = new Book(array);
        int k = 2;
        System.out.println("The title of Chapter " +k+ " is " +b.getChapter(k-1));
        String anotherArray[] = b.getChapters();

        System.out.println("There are " +anotherArray.length+ " chapters.");
        System.out.println(anotherArray);
    }
}
```

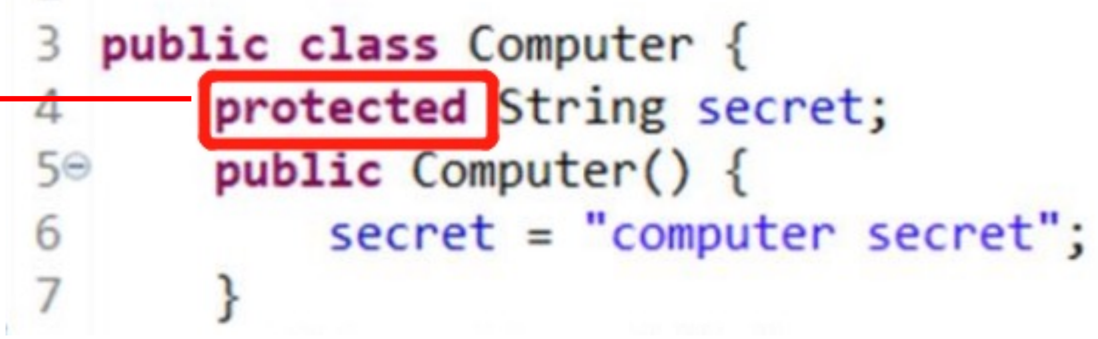
Access method with an object

Access Modifiers

For **classes**, you can use either `public` or *default*.
For **attributes, methods and constructors:**

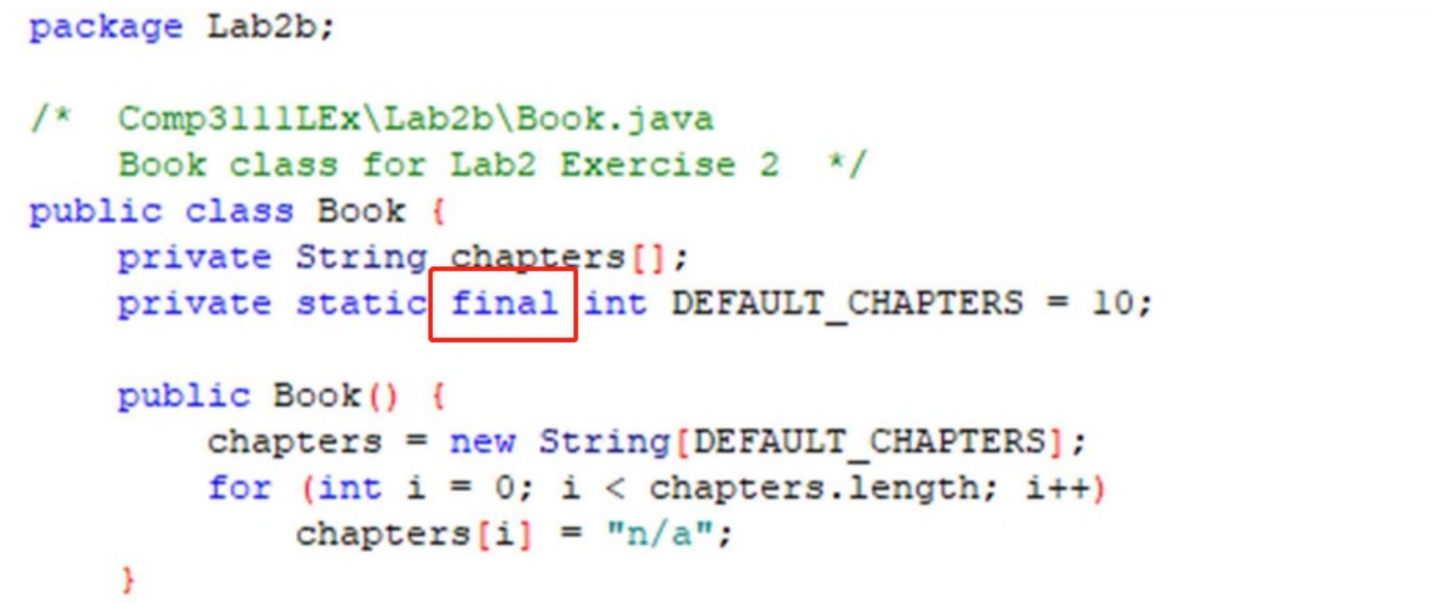


```java

public class Book {
    private String chapters[];
    private static final int DEFAULT_CHAPTERS = 10;

    public Book() {
        chapters = new String[DEFAULT_CHAPTERS];
        for (int i =0; i < chapters.length; i++)
            chapters[i] = "n/a";
    }

```



```java
public class Computer {
    protected String secret;
    public Computer() {
        secret = "computer secret";
    }
```

The final Keyword



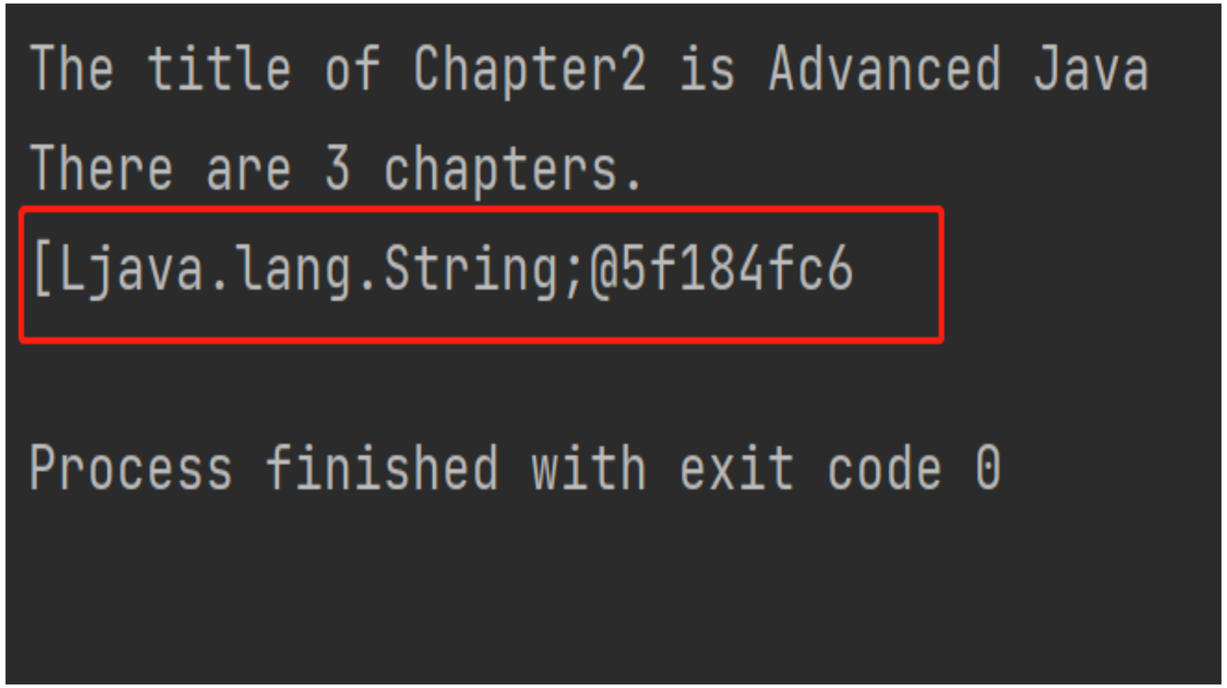
```
package Lab2b;

/*  Comp3111LEx\Lab2b\Book.java
    Book class for Lab2 Exercise 2   */
public class Book {
    private String chapters[];
    private static final int DEFAULT_CHAPTERS = 10;

    public Book() {
        chapters = new String[DEFAULT_CHAPTERS];
        for (int i = 0; i < chapters.length; i++)
            chapters[i] = "n/a";
    }
```

final variable can only be assigned once, after that they become read-only.

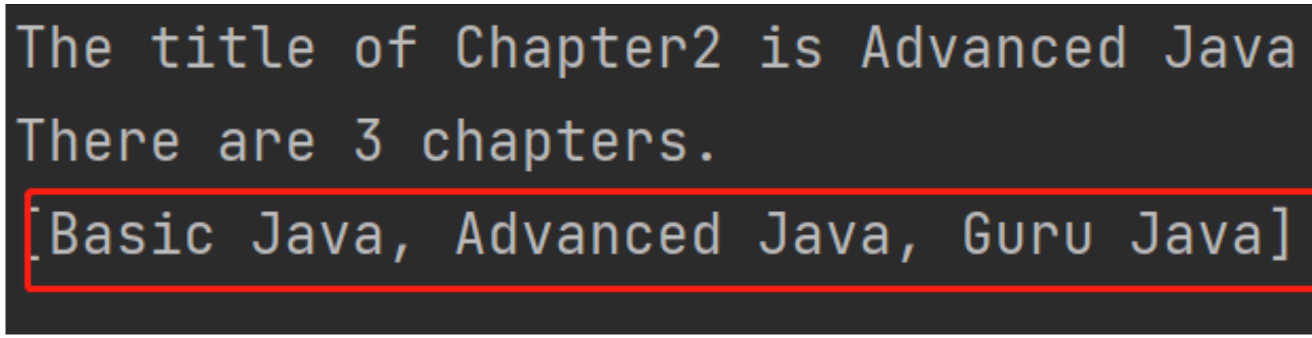
Using java.util.Arrays.toString

Without using java.util.Arrays.toString

Using java.util.Arrays.toString



```
The title of Chapter2 is Advanced Java
There are 3 chapters.
[Ljava.lang.String;@5f184fc6

Process finished with exit code 0
```



```
The title of Chapter2 is Advanced Java
There are 3 chapters.
[Basic Java, Advanced Java, Guru Java]
```

Exercise 3: Learning and practicing Interface and Inheritance in Java

Inheritance

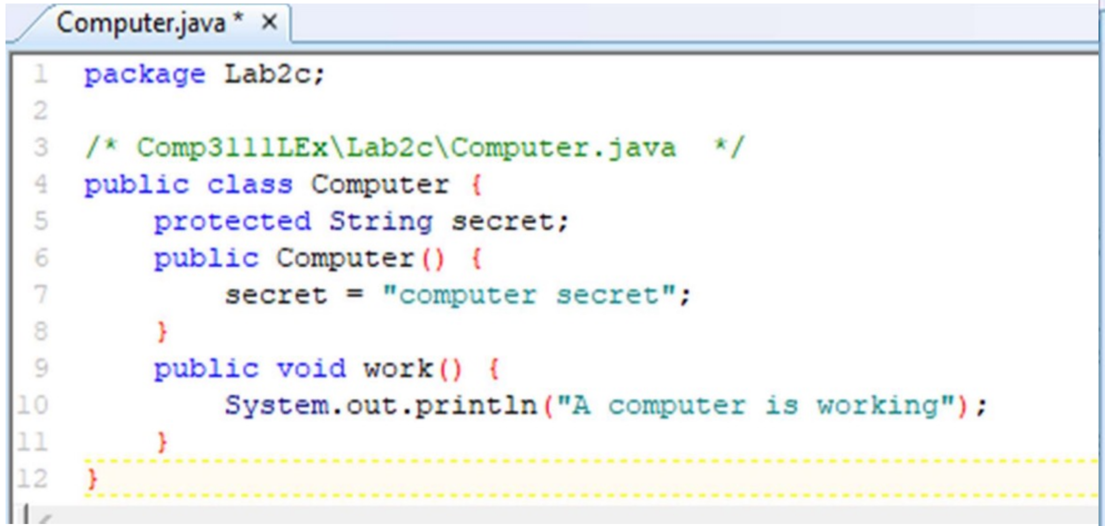
In Java, it is possible to inherit attributes and methods from one class to another.

We group the "inheritance concept" into two categories:

- **subclass** (child) - the class that inherits from another class
- **superclass** (parent) - the class being inherited from

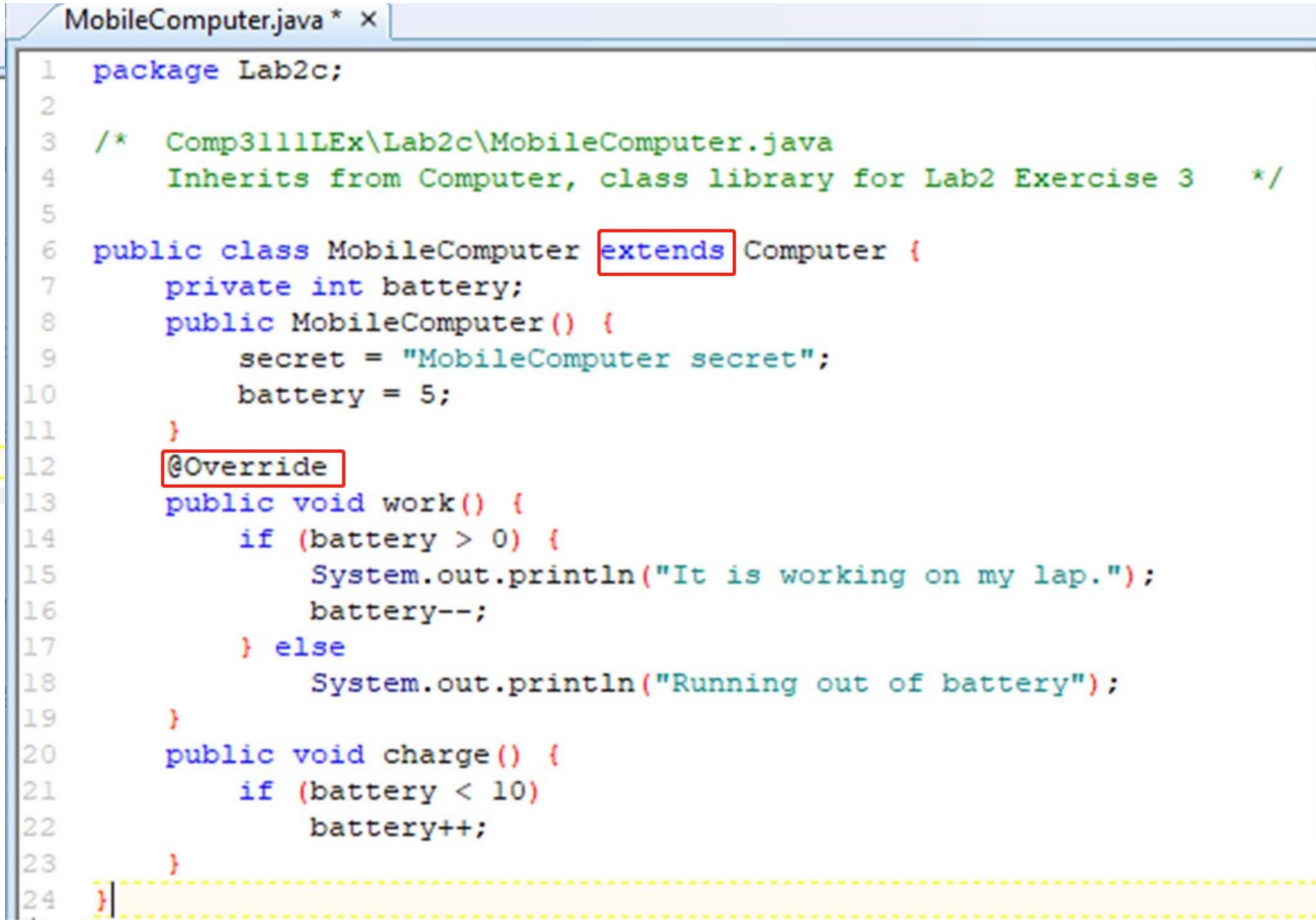
To inherit from a class, use the `extends` keyword.

An example of Inheritance



Computer.java * ×

```java
package Lab2c;

/* Comp3111LEx\Lab2c\Computer.java  */
public class Computer {
    protected String secret;
    public Computer() {
        secret = "computer secret";
    }
    public void work() {
        System.out.println("A computer is working");
    }
}
```



MobileComputer.java * ×

```java
package Lab2c;

/*  Comp3111LEx\Lab2c\MobileComputer.java
    Inherits from Computer, class library for Lab2 Exercise 3   */

public class MobileComputer extends Computer {
    private int battery;
    public MobileComputer() {
        secret = "MobileComputer secret";
        battery = 5;
    }
    @Override
    public void work() {
        if (battery > 0) {
            System.out.println("It is working on my lap.");
            battery--;
        } else
            System.out.println("Running out of battery");
    }
    public void charge() {
        if (battery < 10)
            battery++;
    }
}
```

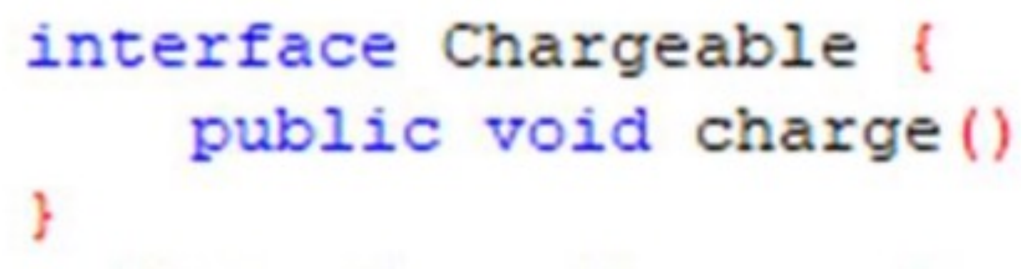
a) We use the keyword extends to inherit a base class.

b) @Override is an annotation.

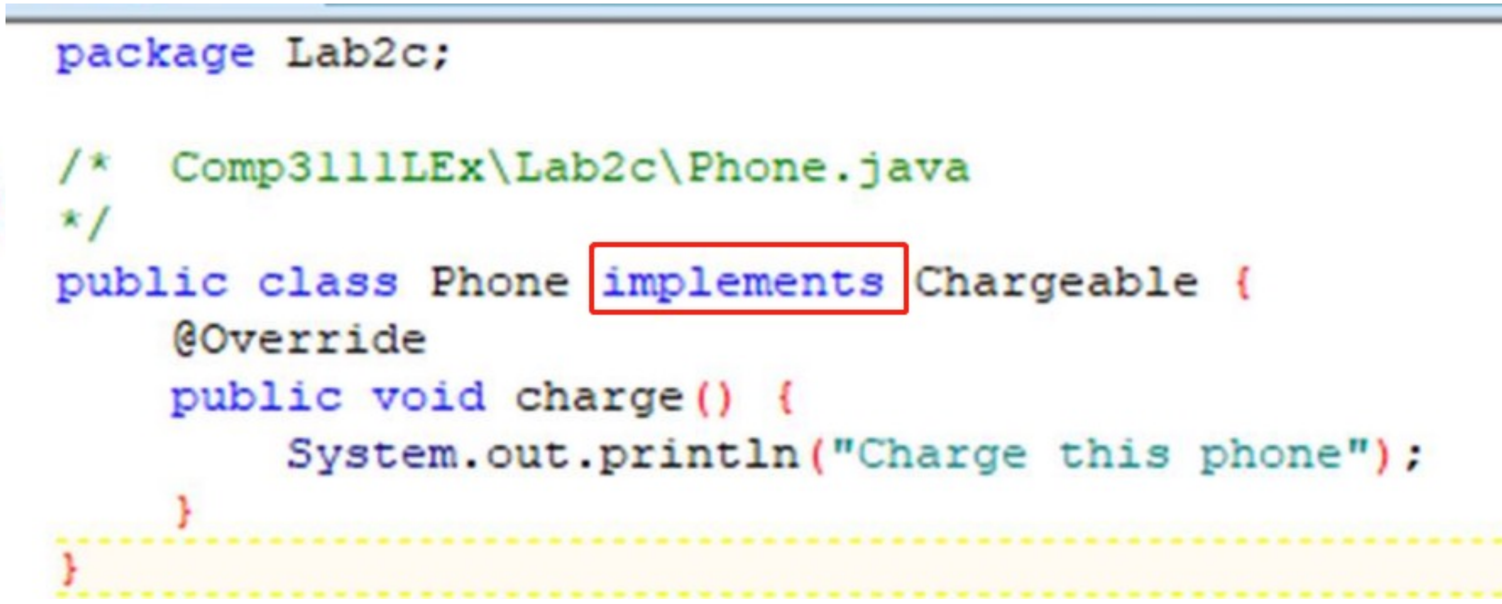
This annotation explicitly tells the compiler that we are overriding the parent’ s method (or member function in C++ terminology).

Interface

Why Use Interfaces?
1) To achieve security - hide certain details and only show the important details of an object (interface).
2) Java does not support "multiple inheritance‟. However, it can be achieved with interfaces, because the class can **implement** multiple interfaces.



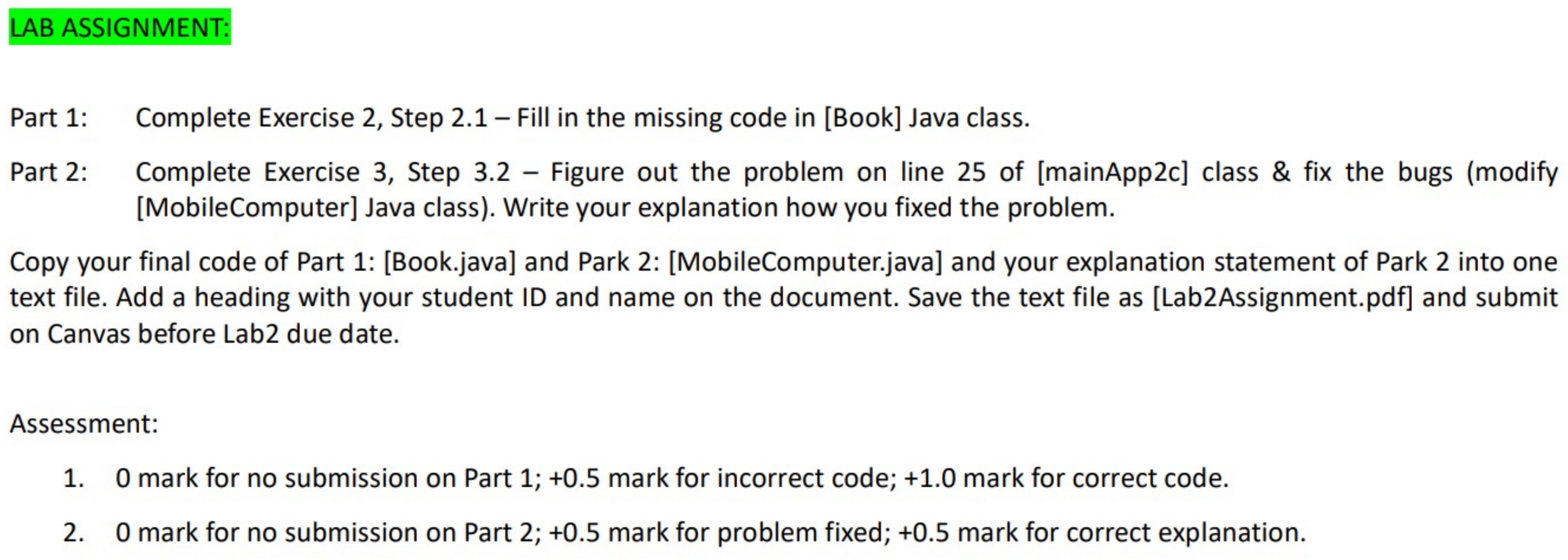
```
interface Chargeable {
    public void charge()
}
```



```
package Lab2c;

/*  Comp3111LEx\Lab2c\Phone.java
*/
public class Phone implements Chargeable {
    @Override
    public void charge() {
        System.out.println("Charge this phone");
    }
}
```

To access the interface methods,
the interface must be "implemented" by another class with the `implements` keyword.

Submission



LAB ASSIGNMENT:

Part 1: Complete Exercise 2, Step 2.1 – Fill in the missing code in [Book] Java class.

Part 2: Complete Exercise 3, Step 3.2 – Figure out the problem on line 25 of [mainApp2c] class & fix the bugs (modify [MobileComputer] Java class). Write your explanation how you fixed the problem.

Copy your final code of Part 1: [Book.java] and Park 2: [MobileComputer.java] and your explanation statement of Park 2 into one text file. Add a heading with your student ID and name on the document. Save the text file as [Lab2Assignment.pdf] and submit on Canvas before Lab2 due date.

Assessment:

1. 0 mark for no submission on Part 1; +0.5 mark for incorrect code; +1.0 mark for correct code.
2. 0 mark for no submission on Part 2; +0.5 mark for problem fixed; +0.5 mark for correct explanation.