COMP3111: Software Engineering

Java Programming – a Warm-up exercise

Learning Outcomes:

- Be able to write a Java program and compile it with Gradle
- Be able to describe the concept interface and inheritance in Java

Supervised Lab Exercises

Environment: Eclipse (Version: Photon RC3 (4.8.0RC3)) with Java Development Kit (JDK 8 64-bits) installed on a Windows 10. The steps may be slightly difference if you are using other versions of Eclipse or Mac

Pre-requisite Reading:

We assume you have C++ experience (from COMP2011/2012 or 2012H) while you might not have any Java experience. Before coming to the lab, please read the given materials.

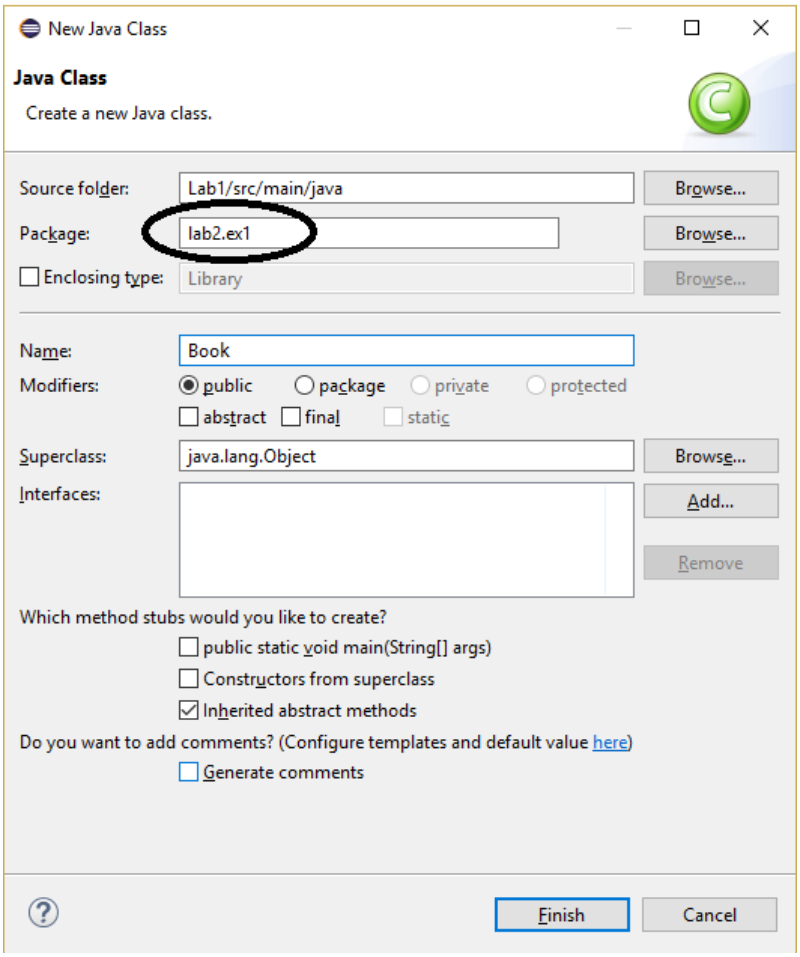
1. Lecture slide: COMP3021 Topic 1: Introduction to Java Programming.

Exercise 1: Programming the basics

Step 1.1. Repeat Lab 1 Step 1.1 to Step 1.8 to create a Gradle project and try to compile it. This time name your project Lab2.

Note: Alternatively you can reuse the project in Lab1.

Step 1.2. Add a new class Book under the package lab2.ex1 as follows:



Step 1.3 Open Library.java and edit it as follows. Please do not forget line 1. That lines says we will are going to use include the class Book from the package lab2.ex1.

Book.java | Library.java

```
1  import lab2.ex1.Book;
2  /*
3   * This Java source file was generated by the Gradle 'init' task.
4   */
5  public class Library {
6      /* Add this function */
7      public static void main(String arg[]) {
8          final String array[] = {"Basic Java", "Advance Java", "Forget about Java"};
9          Book b = new Book(array);
10         System.out.println("The title of Chapter 1 is " + b.getChapter(1));
11         String anotherArray[] = b.getChapters();
12
13         System.out.println("There are " + anotherArray.length + " chapters.");
14         System.out.println(anotherArray);
15
16     }
17
18     public boolean someLibraryMethod() {
19         return true;
20     }
21 }
22
```

Step 1.4 Create Book.java from the package lab2.ex1. You should start your Book.java with the following code and complete the program.

```java
package lab2.ex1;

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] = "Chapter " + i;
    }
    public Book(String argument[]) {

      /* construct the object with an array of chapter names */

    }
    public String getChapter(int i) {

      /* return the chapter by the given index. if the index is < 0 or
      >= to the array size, return "invalid chapter" */
    }
    public String[] getChapters() {
        return chapters;
    }
}
```

Your running result should be

```
:compileJava
:processResources NO-SOURCE
:classes
:run
The title of Chapter 1 is Advance Java
There are 3 chapters.
[Ljava.lang.String;@15db9742

BUILD SUCCESSFUL in 1m 10s
2 actionable tasks: 2 executed
```

Step 1.5: When we print an array variable directly, it will only print the type and its address. We can instead use some APIs like java.util.Arrays for help. Change line 14 of Library.java to

```
System.out.println(java.util.Arrays.toString(anotherArray));
```

Your new running result should be

```
:compileJava
:processResources NO-SOURCE
:classes
:run
The title of Chapter 1 is Advance Java
There are 3 chapters.
[Basic Java, Advance Java, Forget about Java]

BUILD SUCCESSFUL in 5s
2 actionable tasks: 2 executed
```

Note: the API java.util.Arrays is from another package. Therefore, we need to either be verbose to refer the class together with the package name (like what we did) or to import the package at the top of the program (adding import java.util.Arrays;)

Note2: to learn how other API works, please refer to the Java 8 API. E.g. *https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html*

Exercise 2: About inheritance and interface

Step 2.1. Create another package lab2.ex2
Step 2.2. Create two classes: Computer and MobileComputer as below, so that MobileComputer inherits from Computer.

Book.java | Library.java | Computer.java | MobileComputer.java

```java
package lab2.ex2;

public class Computer {
    protected String secret;
    public Computer() {
        secret = "computer secret";
    }
    public void work() {
        System.out.println("A computer is working");
    }
}

```

Book.java | Library.java | Computer.java | *MobileComputer.java

```java
package lab2.ex2;

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++;
    }
}
```

*Note: We use the keyword **extends** to inherit a base class.*

Note: @Override is an annotation. This annotation explicitly tell the compiler that we are overriding the parent's method (or member function in C++ terminology).

Step 2.3. Change the driver program Library.java as below. Run the program to verify your result.

```
import lab2.ex1.Book;
import lab2.ex2.*;
/*
 * This Java source file was generated by the Gradle 'init' task.
 */
public class Library {
    /* Add this function */
    public static void main(String arg[]) {
        final String array[] = {"Basic Java", "Advance Java", "Forget about Java"};
        Book b = new Book(array);
        System.out.println("The title of Chapter 1 is " + b.getChapter(1));
        String anotherArray[] = b.getChapters();

        System.out.println("There are " + anotherArray.length + " chapters.");
        System.out.println(java.util.Arrays.toString(anotherArray));

        /** Add the following**/
        Computer a = new MobileComputer();
        for (int i = 0; i < 10; i++)
            a.work();


    }
}
```

Your result should look like

```
:compileJava
:processResources NO-SOURCE
:classes
:run
The title of Chapter 1 is Advance Java
There are 3 chapters.
[Basic Java, Advance Java, Forget about Java]
It is working on my lap.
It is working on my lap.
It is working on my lap.
It is working on my lap.
It is working on my lap.
Running out of battery
Running out of battery
Running out of battery
Running out of battery
Running out of battery

BUILD SUCCESSFUL in 0s
2 actionable tasks: 2 executed
```

Step 2.4. Create a class called Charger that has one function. Also create the interface Chargeable inside this Charger.java

Book.java | Library.java | Computer.java | MobileComputer.java | Charger.java

```
1  package lab2.ex2;
2
3  interface Chargeable {
4      public void charge();
5  }
6
7  public class Charger {
8      public void charge(Chargeable c) { c.charge(); }
9  }
10
```

Step 2.5. Create another class Phone as below.

Book.java | Library.java | Computer.java | MobileComputer.java | Phone.java

```
1  package lab2.ex2;
2
3  public class Phone implements Chargeable {
4      @Override
5      public void charge() {
6          System.out.println("Charge this phone");
7      }
8  }
```

Note: Using C++ terminology, an interface in Java has only pure-virtual functions. Unlike an abstract class which is allowed to have non pure-virtual functions, an interface cannot contain any implementation.

Step 2.6. Then change the driver program Library.java as

Book.java | Library.java | Computer.java | MobileComputer.java | Phone.java

```
5   */
6  public class Library {
7      /* Add this function */
8      public static void main(String arg[]) {
9          final String array[] = {"Basic Java", "Advance Java", "Forget about Java"};
10         Book b = new Book(array);
11         System.out.println("The title of Chapter 1 is " + b.getChapter(1));
12         String anotherArray[] = b.getChapters();
13
14         System.out.println("There are " + anotherArray.length + " chapters.");
15         System.out.println(java.util.Arrays.toString(anotherArray));
16
17         /** Add the following**/
18         Computer a = new MobileComputer();
19         for (int i = 0; i < 10; i++)
20             a.work();
21
22         Charger c = new Charger();
23         Phone p = new Phone();
24         MobileComputer m = new MobileComputer();
25
26         c.charge(p);
27         c.charge(m); //this does not work without fixing MobileComputer
28
29     }
30 }
```

Note, this program would not compile on line 27

Lab Activity and Assessment

Lab Activity

1) Complete the missing code in lab2.ex1.Book.
2) Understand why Exercise 2 does not compile.
3) Fix lab2.ex2.MobileComputer so that it works

Assessment

1) Show your code to the TA
2) Explain why your fix in MobileComputer would work

Note: Unlike Lab 1, you cannot submit this online.

Reference

Tutorial on interface: https://docs.oracle.com/javase/tutorial/java/IandI/index.html