IOOP2 2025 Lab Sheet B
“Library of Legends” Intro

This Lab Sheet contains material based on Week 2 topics.

The deadline for Moodle submission of this lab exercise is 12:00 noon on Thursday 9 October.

Aims and objectives

- Writing properly formed Java classes including fields and methods
- Using provided code to test your own code
- Understanding and explaining how Java classes interact with each other

Starter code

Download the file **LabB.zip** from the Moodle page and unzip it somewhere in your own file space. Open the **LabB** folder in VSCode to access the starter code.

Tasks

This lab begins our development of “Library of Legends”, a semester-long programming project where we will slowly develop a system including magical books, heroic characters, evil villains, and the interactions between them. Each lab will involve tasks based on the weekly topics, building on the previous week’s work, and by the end of the course we will integrate those pieces into a shared codebase that supports a simple fantasy battling game.

In this first week, we will create Java classes corresponding to the two basic entities in the game world: the **Book**, which contains information about how to perform skills, and the **Hero**, who can read the books to learn new skills. We have also provided a simple game harness that you can use to test your code.

Task 1: Representing a Book

Create a new Java class **Book** to represent a magical book. You can create a new class through the “New Java File” menu, which you can access by right-clicking on the **src** directory in your **LabB** folder in VSCode.

The **Book** class should have fields representing the following information:

- The title (represented as a String)
- The author (represented as a String)
- The number of pages (represented as an int)
- The skill that the book can teach the user (represented as a String)

One example book might be:

- Title: “Sword Mastery”
- Author: “K. Steel”
- Number of pages: 250
- Skill: “Swordplay”

Your **Book** class should include a constructor that sets all the required fields, which should be specified in the order given above. Your class should also have **get** methods for all the fields; you should not implement any **set** methods. Check the code in **GameHarness.java** to make sure that your constructor arguments match what we are expecting.

You should also include a method with the following signature:

```
public void printDetails()
```

> This method should print all the details of the book to **System.out**, in a human-readable manner.

Task 2: Representing a Hero

Create another new Java class called **Hero** to represent the hero of the game world.

The **Hero** class should have the following fields:

- The name (represented as a String)
- The currently equipped skill (represented as a String)

The constructor for the **Hero** class should set the hero’s name to the given string and should initialise the skill to the string “None”. You should implement **get** and **set** methods for both fields.

You should also implement the following methods:

```
public void printDetails()
```

> This method should print all details of the hero to **System.out**, in a human-readable manner.

```
public void useSkill (String skillName)
```

> This method should check whether the user’s current skill is identical to **skillName**. If it matches, the method should print “[Hero name] uses [skillName]” (e.g., “Rowan uses Swordplay”). If it does not match, the method should print “[Hero name] does not know how to do that”.

```
public void readBook (Book book)
```

This method should do two things:

- Set the hero’s current skill to the skill provided by the book
- Print out “[Hero name] has read ‘[Book title]’ and now knows: [skill name]”. For example, “Rowan has read ‘Sword Mastery’ and now knows: Swordplay”

Testing your code

You can test your code using the **main** method of the provided **GameHarness.java** file. **You should not change this code in any way** – if the code is not working, this means that there is an error in your implementation. We will test your code using our version of **GameHarness.java**!

What to submit

Before the deadline, you should submit your final version of the two source files, **Book.java** and **Hero.java**. You should not submit **GameHarness.java** as you should not have made any changes to this file.

In addition to the source files, you should also submit a text document (MS Word or plain text) including the following information:

1. A trace of **GameHarness** running on your code, cut and pasted from the Terminal output window in VSCode. For the trace, you must create the Hero with **your own name**, and your trace should show the Hero **reading at least one book** and **attempting to use at least one skill**.
2. A short walkthrough in English of how your **Hero.readBook()** method works – explain how you access the skill provided by the book and how you update your Hero's skill.

Marking scheme

This lab is marked out of 6, as follows.

- 1 mark each for submitting a version of **Hero.java** and **Book.java**, whether working or not
- 1 mark each for fully correct versions of **Hero** and **Book**, fully meeting the specification given here and following good object-oriented principles – partial marks will be given if the code has issues
- 1 mark for the **GameHarness** trace, which must include all the items mentioned above
- 1 mark for a correct English walkthrough of the **Hero.readBook()** method functions

Optional tasks

If you have completed the lab tasks, here are some additional things you might want to try. Note that we will be extending these classes significantly over the next several weeks, but if you want to try something new in the meantime, here are some ideas.

1. Add a **level** field to **Hero** and increase it every time they read a book.
2. Add a **level** field to **Book** and only allow a hero to read a book if their level is high enough.
3. Allow the hero to "forget" a skill.

You should attempt these changes in a copy of the lab code, or after you have submitted your answer to Lab B, as some of these changes will modify the behaviour of the basic game engine in an incompatible way.