Software Development (CS2514) Assignment 2

Interfaces and Enumerated Types (Version 1)

(Specifications subject to change; Due: April 25; Marks: 50)

General Comments

Please, carefully read the submission guidelines before you submit the assignment.

This is an exercise about implementing *maintainable* classes and interfaces. You should always assume the specifications may change (slightly) and you should make sure your implementation can implement these changes with the minimum amount of effort.

Before you start implementing your classes, please make sure you understand the API. If you do, this'll make your life much much easier.

Each class should have a class `JavaDoc`. Besides that, there is *no need to put in `JavaDoc` comments.*

Learning Objectives

For this assignment you will learn about interfaces, enumerated types, and delegation in `Java`. You will learn:

- How to design class (and interface) hierarchies using interfaces;
- You will learn how to apply the *"Encapsulate What Varies"* design principle, which we studied in the lectures:
  - Define an interface for each class of APIs;
  - For each API and for each difference specific behaviour which arises in the API, define a concrete class which (1) implements the API and which (2) provides the required behaviour;
  - Use delegation to re-use the implementation of the concrete classes.
- How to use enumerated classes;
- Much, much more.

Please note that you are **not** supposed to use **inheritance**, so you may **not** extend classes. Of course, you may extend interfaces. You may use pre-defined/existing generic classes but you are not supposed to implement your own generic classes.

Your aim should be to maximise code re-use of shared behaviour in the hierarchy, so *don't use copy-and-paste* to "implement" shared behaviour.

Additional Implementation Details

Please remember that your classes should respect encapsulation, so all class and instance attributes should be `private`. Using `private` attributes is part of the challenge. If you use attributes that are not `private` you can lose up to 50% of the marks.

Each class in your implementation should be in its own `.java` source file. Multiple classes per source file are not allowed. You may lose up to 50% of the marks if you submit files consisting of more than one class.

There is no need to provide `JavaDoc` comments for public instance methods.

Main Task

Implement a class extension-free collection of classes for books, authors, and readers. Your classes should use interfaces, enumerated classes, and concrete final classes. You may use existing interfaces and classes (including generic and enumerated classes) but you may not extend classes and implement your own generic classes.

- Each book has a title, which may have a sub-title, an author, and a price.
- Books can be printed books (hardbacks, paperbacks, …) or audio books.
- A hardback costs 12 Euro, a paperback costs 10 Euro, and an audio book costs 15 Euro.
- Printed books have a page count (number of pages), e.g. 110.
- Audio books have a duration in minutes, e.g. 120.75.
- Both authors and readers are persons. The only difference is that authors can write books.
- Each person (author/reader) can buy books.
- Each person has a collection of books which they own and which they can print.
- Each author has a collection of published books, which they can print.
- Each person has earnings, which is a monetary value (initially zero).
  - ⋆ The instance method `receive( x )` increases the earnings by `x`.
  - ⋆ The instance method `charge( x )` decreases the earnings by `x`.
- The instance method `buy( b )` is a person's attempt to purchase the book `b`. The purchase should only succeed if the person's earnings are greater than or equal to the price of `b`.
- When a person successfully purchases a book, they should be charged for the book's price, the book should be added to their owned books, and the book's author should earn `0.10` times the book's price.
- You must implement authors using an enumerated `Author` class.
- Please include a `Main` class with a `main` method which demonstrates your constructor and method calls.
- **Careful**: The implementation of the printing of lists should be shared.

The instances of some of your concrete (delegate) classes will be used as delegates in other classes. The names of these concrete (delegate) classes should start with the word `Concrete`.

The following shows a possible scenario of an author/buyer simulation. Each book constructor automatically publishes the book for its author, i.e. it adds the book to the author's list of published books.

```
final Title t1 = new Title( "The Hobbit", "An Unexpected Journey" ); // create title #1: main title and subtitle
final Title t2 = new Title( "Catcher in the Rye" );                  // create title #2: main title only
final Author a1 = Author.JJR_Tolkien;                                // ‘‘create’’ author #1
final Author a2 = Author.JD_Sallinger;                               // ‘‘create’’ author #2
final Book b1 = new AudioBook( a1, t1, 1800.5 );                     // create and publish book #1
final Book b2 = new Paperback( a2, t2, 100 );                        // create and publish book #2
final Reader r1 = new Reader( "Joe", "Soap" );                       // create reader #1

a1.receive( 100 ); // author #1 receives some money to spend
a1.buy( b2 );      // author #1 buys book worth 10 Euro
r1.buy( b1 );      // reader #1 tries to buy book worth 15 Euro: fails
r1.receive( 100 ); // reader #1 receives some money to spend
r1.buy( b1 );      // reader #1 buys book worth 15 Euro: now works
r1.buy( b2 );      // reader #1 buys book worth 10 Euro
```

```
a1.printBooksOwned( );
a1.printBooksPublished( );
r1.printBooksOwned( );
// a1 owns 100 - 10 + 0.10 * 15 = 91.5 Euro
System.out.println( a1 + " owns " + a1.getEarnings( ) + " Euro" );
```

The following is the output.

```
You don't have enough money to buy
   AudioBook[ author = JJR Tolkien, title = The Hobbit / An Unexpected Journey, price = 15.0, duration = 1800.5 ]
JJR Tolkien owns:
   Catcher in the Rye
JJR Tolkien published:
   The Hobbit / An Unexpected Journey
Joe Soap owns:
   The Hobbit / An Unexpected Journey
   Catcher in the Rye
JJR Tolkien owns 91.5 Euro
```

Submission Details

- Remember that each class should be in its own file. You will lose marks if you violate this rule.
- Please provide your name and student ID as part of your class `JavaDoc`. You can use the `@author` tag for this:

```Java
@author Java Joe (ID 12345678)
```

- Use the CS2514 `Canvas` section to upload your classes as a single *.tgz* archive called *Lab-2.tgz* before 23.55pm, April 25, 2020. To create the *.tgz* archive, do the following:
  - Create a directory `Lab-2` in your working directory.
  - Copy `Main.java` and your other user-defined Java files into the directory. Do not copy any other files into the directory.
  - Run the command '`tar cvfz Lab-2.tgz Lab-2`' from your working directory. The option 'v' makes `tar` very chatty: it should tell you exactly what is going into the `.tgz` archive. Make sure you check the `tar` output before submitting your archive.
  - Note that file names in `Unix` are case sensitive and should not contain spaces.
- Note that the format of your submission should be `.tgz`: do *not* submit `zip` files, do *not* submit `tar` files, do *not* submit `bzip` files, and do *not* submit `rar` files. If you do, it may not be possible to unzip your assignment.