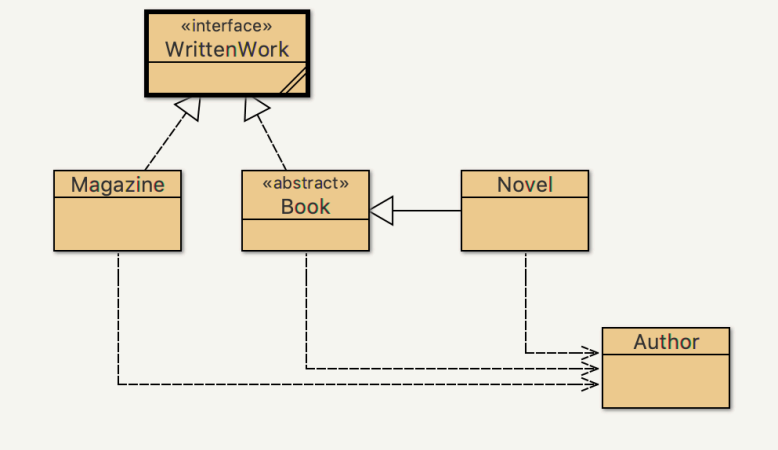
The University of the West Indies, St. Augustine
COMP 2603 Object Oriented Programming 1
2020/2021 Semester 2
Lab 6

In this lab, we will continue to explore the polymorphic behaviour of subclass and superclass instances. This lab builds on the concepts of abstract classes and interfaces. ArrayLists are introduced in this lab.

Part 1: Abstract Classes, Interfaces, ArrayLists



1. Create a new project in BlueJ called Lab 6.
2. Create an interface called **WrittenWork** with the following methods:
   - `public int getNumPages();`
   - `public String getAuthorName();`
   - `public String getTitle();`
   - `public double getPrice();`
   - `public void addAuthor(Author author);`

   Replace the semi-colon symbol from the **getPrice(**) method with curly brackets as for a normal method { }. What do you notice? Why does this happen? Restore the semi-colon when you are finished answering the questions.

   We get a compilation error. This is because interfaces cannot have methods with any functionality (body). All methods are abstract by default and thus must not contain a body.

3. Create a runner class called **BookStore**. Try to create a direct instance of type **WrittenWork**. What do you notice? Why does this happen?

We get a compilation error. This is because we cannot possibly create an instance of an interface. Interfaces have no constructors and are inherently abstract.

4. Create an **abstract** class called **Book**. Try to create a direct instance of type **Book** in the **BookStore** class. What do you notice? Why does this happen?

We get a compilation error. This is because we cannot possibly create an instance of an abstract class.

5. Modify the class signature of **Book** so that it implements the **WrittenWork** interface. Why is the **Book** class allowed to skip the implementation of the **WrittenWork** interface methods?

It is allowed to skip the implementation of WrittenWork because it is abstract. However, any concrete subtypes of Book must implement the methods.

6. Remove the **abstract** keyword from the **Book** class signature. Does the class compile? Explain what happened using the terms abstract class, concrete class and interface.

We get a compilation error. This is because the Book class will now be a concrete class, therefore it must provide implementation for the WrittenWork interface methods.

7. Import the **Author** class, from the lab source folder on myElearning, into your project.

8. Fill in the following details of the **Book** class:

Attribute	Type	Purpose
author	Author	The author of the Book
title	String	The title of the Book
numPages	int	The total number of pages in the Book
price	double	The price of the Book

Method	Return Type	Purpose
Book (String title, int numPages)		Book class constructor
addAuthor(Author a)	void	Sets the author of the Book
getAuthorName( )	String	Returns the Book author's name
getTitle( )	String	Accessor for the title attribute
getNumPages( )	int	Accessor for the numPages attribute

9. Create a new class called **Novel** and make it a subclass of **Book**.
    a. What is the first error flagged by the compiler? Why does this happen?

The first error is that the Novel class is not abstract and must implement methods from the WrittenWork interface. This is because all concrete methods that implement an interface (even indirectly) must provide implementations for its interface's methods.

    b. Fix the error by inserting a getPrice( ) method in the **Novel** class that calculates and returns the price of the Novel where each page costs 0.75 cents.

    c. What is the second error flagged by the compiler now? Why does this happen?

The second error is that the Novel class does not invoke its parent's constructor. We need to invoke this from within a Novel class constructor.

    d. Fix the error by inserting a constructor in the **Novel** class that accepts three parameters (author, title, numPages) and invokes the parent constructor using super(..)

    e. Add a t**oString( )** method that returns "NOVEL: " appended with the title, author name, price and number of pages on separate lines.

10. Create and associate the following **Author** and **Novel** objects in the **BookStore** runner class:

Object	Declared Type	Features
a1	Author	Malcolm Gladwell
a2	Author	Steven Johnson
a3	Author	Mathias Johansson
a4	Author	Evan Ackerman
a5	Author	Erico Guizzo
a6	Author	Fan Shi
w1	WrittenWork	What the Dog Saw and other adventures, 503 pages, Author: Malcolm Gladwell
w2	WrittenWork	How We Got to Now: Six Innovations That Made the Modern World, 320 pages Author: Steven Johnson
w3	WrittenWork	Everything Bad is Good for You: How Today's Popular Culture is Actually Making Us Smarter, 254 pages, Author: Steven Johnson

Print the details of all **Novel** objects using the **toString( )** method.

11. In the **BookStore** runner class: Create an ArrayList of **WrittenWork** objects called **products**, and add the appropriate objects from Step 10. Example:

```
ArrayList<WrittenWork> products = new ArrayList<>( );
products.add(w1);
```

12. In the **BookStore** runner class: Print the details of the objects in the **products** ArrayList using a for loop:

```
for( WrittenWork w: products)
  System.out.println(w.toString( ) );
```

13. In the **BookStore** runner class: Create an **ArrayList** of **Author** objects called **authors**, and add the appropriate objects from Step 10.

14. In the **BookStore** runner class: Print the details of the objects in the **author** ArrayList using a for loop. What do you notice?

The details of the author objects were printed by invoking the Author class's toString method.

15. In the **BookStore** runner class: Write code that would go through the ArrayLists, check whether an author has written a book, and update the appropriate count in the author object. Repeat Step 14 to verify that your code works.

16. Create a new concrete class called **Magazine** and make it a subtype of **WrittenWork**. What do you notice?

We get a compilation error. This is because the Magazine class is concrete, therefore it must provide implementation for the WrittenWork interface methods.

a. Introduce one attribute in the **Magazine** class: a **title**

b. Introduce the following methods in the **Magazine** class:

Method	Return Type	Purpose
Magazine (String title)		Magazine class constructor (sets the title)
addAuthor (Author a)	void	Adds an author to the Magazine. **Leave empty for now**
getAuthorName( )	String	Returns a list of the names of all authors who contributed to the magazine. **Return null for now.**
getTitle( )	String	Accessor for the Magazine title attribute
getNumPages( )	int	Returns the total number of pages in the magazine. **Return 0 for now**
getPrice( )	double	Returns 50.00 (Same price for all Magazines).

17. In the **Magazine** class:

    a. Introduce an **ArrayList** called **authors** that holds **Author** objects. A Magazine can have multiple authors and we want to be able to reflect this.

    b. Modify the **addAuthor(..)** method so that a new author is added to the authors ArrayList when the method is invoked. (See ArrayList add method in the API).

    Test that your method works by creating a **Magazine** object in the **BookStore** class as follows:

    ```
    WrittenWork mag = new Magazine("IEEE Spectrum");

    products.add(mag);
    mag.addAuthor(a3);
    mag.addAuthor(a4);
    mag.addAuthor(a5);
    mag.addAuthor(a6);
    ```

    c. Modify the **getAuthorName( )** method so that a list of all of the authors of a Magazine is returned. Test that your method works by calling the appropriate method in the BookStore class.

    d. Modify the **getNumPages( )** method so that the total number of pages in the magazine is determined by the total number of authors where each author contributes 3 pages. (See ArrayList size method in the API). Test that your method works by calling the appropriate method in the BookStore class.

18. Modify the WrittenWork interface to include one more method:

    public boolean hasAuthor(Author a);

    a. What must be done in the classes that implement the interface?

    We get a compilation error. This is because the Magazine and Novel classes must provide implementation for the hasAuthor() method in the WrittenWork interface.

    b. Make changes to the **Novel** class as appropriate.

    c. Make changes to the **Magazine** class as appropriate. (See ArrayList contains method in the API).

19. Verify that your code results in the following for all WrittenWork objects:

    a. Malcolm Gladwell authored a Novel specifically.

    b. Steven Johnson wrote the largest number of Books in the book store.

    c. IEEE Spectrum has the largest number of Authors in the book store.

    d. No WrittenWork objects were authored by Spencer Johnson.