Lab 16: Input, Output and Exceptions

Objective

In this lab, we will take advantage of the **java.io** package in order to add more flexibility to our **Inventory** system class.

Overview

In this lab you will:

- Populate a file with information about books.
- Read from that file and use the information to create Book objects.
- Exercise your code to ensure it is working correctly.

Step by Step Instructions

Exercise 1: Reading Information From a File

For our Inventory system, we'd like to load our **Inventory** from a text file. Doing so for all of the different inventory types would be a lengthy exercise and we will leave that open as a challenge to the student with time on his/her hands. Instead we will simply load some **Book** objects.

1. Create a new class named **BookExerciser** in project Store and package com.javaoo.store.drivers. Ensure that this new class has a `main()` method. Create a variable in the `main()` method of type **List<Book>** named `books`.

2. Add a method named `readBooksFromFile()` in the **BookExerciser** class. This method will be used to populate our **List<Book>** with book information that is read from a file. It should have the following signature:

   ```
   public static List<Book> readBooksFromFile(String name)
   ```

3. We are going to store information about our book inventory in a text file. We will pass the name of the text file to the `readBooksFromFile()` method. The first thing we need to do is open the file and ensure that we can read from it. We would like to read from the file line by line so we will do that with a **LineNumberReader** object. Do the following:

   a. If not using an IDE, import `java.io.*;`
   b. Create a **FileInputStream** from the file name passed into `readBooksFromFile()`
   c. Create an **InputStreamReader** from the **FileInputStream**
   d. Create a **LineNumberReader** from the **InputStreamReader**

e. Ensure that all of this is done inside of a `try-with-resources` block
f. This portion of the code should look as follows:

```
try (
        FileInputStream inFile = new FileInputStream(name);
        InputStreamReader inReader = new InputStreamReader(inFile);
        LineNumberReader lineReader = new LineNumberReader(inReader)
    ) {
```

4. Add a simple while loop to go through the file line by line and output the information to the console. Add a `catch` block to handle any exceptions of type **IOException** that might be generated by the code that is opening and reading the file. The entire method at this point should look as follows:

```
public static List<Book> readBooksFromFile(String name) {

    List<Book> books = null;

    try (
            FileInputStream inFile = new FileInputStream(name);
            InputStreamReader inReader = new InputStreamReader(inFile);
            LineNumberReader lineReader = new LineNumberReader(inReader)
        ) {
        String line;
        while ((line = lineReader.readLine()) != null) {
            System.out.printf("%d: %s%n", lineReader.getLineNumber(), line);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }

    return books;
}
```

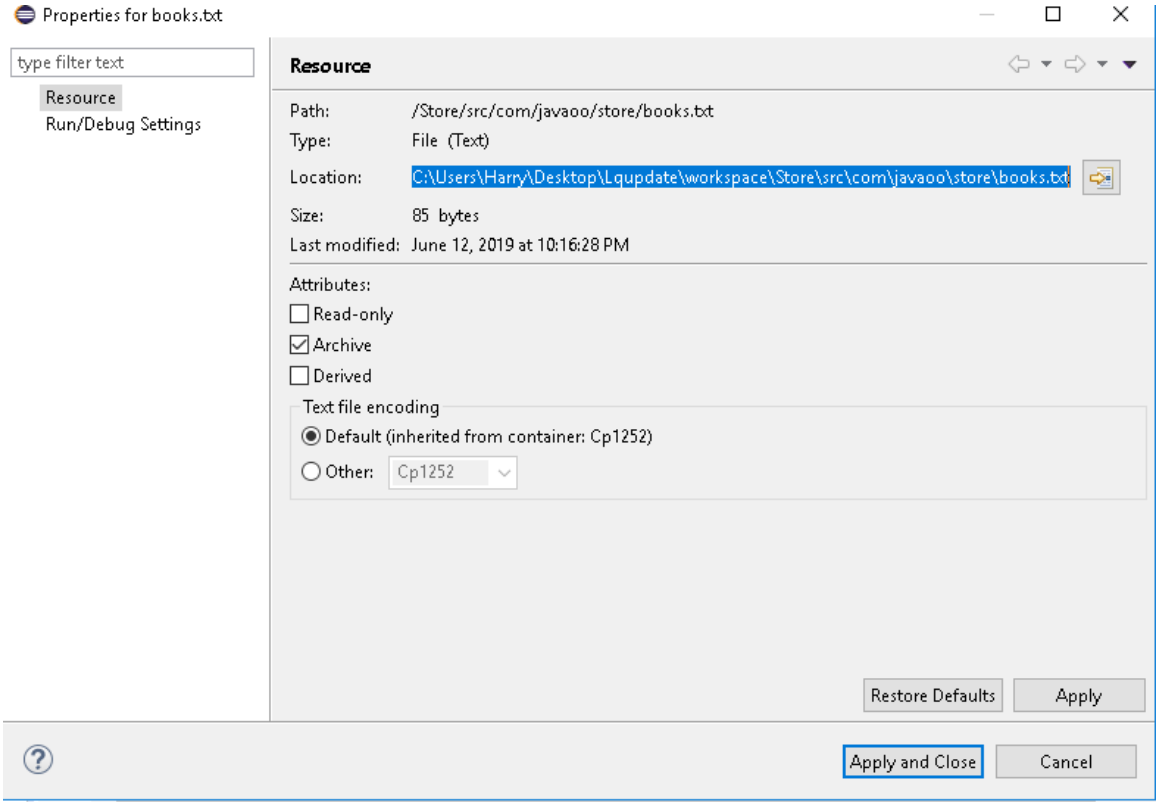
5. Now we'll need to set up the text file to hold **Book** information. Using the New File wizard, create a text file named `books.txt`. Add some books to the text file. To keep it simple we'll store title, author and price. Put one piece of data per line and add two or three books.

   Here's an example of what the file should look like:

   COBOL for Java Programmers
   Isaac B. Mortimer
   45.95
   3D Game Programming for the IPod
   Hawley Smoot
   111.95

Save the file;

6. In the main() method of BookExerciser, create a String variable called location. Right click on books.txt in the Package Explorer, select Properties and copy the file location from the wizard.



Select cancel. Paste the clipboard as a value for the location variable;

7. In the `main()` method in **BookExerciser**, call the `readBooksFromFile()` method, passing it location as the file name. (Backslash means 'escape the next character' in Java so we need to escape the backslash.) Your main method should look as follows:

```
public static void main(String[] args) {
    String location = "E:\\eclipse-workspace\\Store\\src\\com\\javaoo\\store\\drivers\\books.txt";
    List<Book> books = readBooksFromFile(location);
}
```

Note: We are not populating the **List<Book>**, yet. We will do that next.

8. Execute the program. You should see the contents of your text file in the console.

9. For each book, we're going to read three lines of text and the third line will need to be converted to a `double`. Think carefully about the logic you'll need to make this work (assuming all the data in the file is correct). Look at the **Double** class in `java.lang` for guidance. Modify the `while` loop so that it reads the data three lines at a time storing each piece of information in a local

variable. Print out each of these local variables after the 3rd line is read to ensure you are parsing the data correctly.

10. Now we need to use that data and create a **Book** object. You can use the constructor that takes all 6 fields. Set the `quantity` to **5**, the `publisher` to **null** and the `category` to **NON-FICTION**. Create an **ArrayList** to to hold the books, and each time through the loop, add the book in the **List<Book>** `books`. Your code should look something like:

```
public static List<Book> readBooksFromFile(String name) {

    List<Book> books = null;

    try (
            FileInputStream inFile = new FileInputStream(name);
            InputStreamReader inReader = new InputStreamReader(inFile);
            LineNumberReader lineReader = new LineNumberReader(inReader)
        ) {

        books = new ArrayList<>();
        String line;
        while ((line = lineReader.readLine()) != null) {
            System.out.printf("%d: %s%n", lineReader.getLineNumber(), line);
            String title = line;
            String author = lineReader.readLine();
            double price = Double.parseDouble(lineReader.readLine());
            System.out.printf("Book: [Title: %s, Author: %s, Price: $%.2f%n", title, author, price);
            books.add(new Book(title, price, 5, author, null, "NON-FICTION"));
        }
    } catch (IOException e) {
        e.printStackTrace();
    }

    return books;
}
```

11. Add a loop to your `main()` method to print out the titles of all of the books in the `books` collection. Here is how your `main()` method should look.

```
public static void main(String[] args) {
    String location = "C:\\Users\\Harry\\Desktop\\Lqupdate\\workspace\\Store\\src\\com\\javaoo\\store\\books.txt";
    List<Book> books = readBooksFromFile(location);

    for (Book book : books) {
        out.println(book.getTitle());
    }
}
```

12. Execute the program and ensure that it prints out your titles.

Challenge Exercise

13. What happens if the file does not contain valid data? Experiment with different techniques for detecting bad data, and more importantly, recovering from it. This is a very open-ended problem and can be one of the most challenging aspects of writing good code.