Activity 5.2 - GradeBook

Overview

This activity demonstrates how to handle File I/O and Exception handling in Java by creating a simple GradeBook application. This application reads student records, including their grade, from a Comma Separated Value formatted text file (CSV). Each line in a csv file represents a single student record. The GradeBook application will read the student records from the provided CSV file, create a new Student object for each record using the provided Student.java class file, add the objects to an ArrayList, then finally print the entire ArrayList to the console.

Instructions

Getting Started

1. Create an "`Activity5.2`" project in VScode.
2. Import "Student.java", "GradeBook.java", and "gradebook.csv"

Part 1: Reading from a file using a Scanner

Work through `TODO` comments 1, 2 and 3 in `GradeBook.java`. This implements the code that opens the provided gradebook.csv file and steps through it line by line. To verify your code is working, try using a print statement to display each line of the file in the console. See the sample output below:

```
DEBUG: Processing student record -> Stinson,Barney,0001,63
DEBUG: Processing student record -> Scherbatsky,Robin,3002,103
DEBUG: Processing student record -> Mosby,Ted,6003,91
DEBUG: Processing student record -> Aldrin,Lily,1413,82
DEBUG: Processing student record -> Eriksen,Marshall,1423,87
DEBUG: Processing student record -> Blauman,Gary,7123,21
DEBUG: Processing student record -> Zinman,Stella,6326,76
```

Part 2: Accessing individual fields in CSV formatted data using a Scanner

Work through `TODO` comment 4 in `GradeBook.java`. This code will be executed for each student record in the CSV file. The data fields within each line of CSV data are formatted as follows:

```
Lastname,Firstname,StudentID,Grade
```

To access the individual fields, the first step is to **create a second Scanner** to process each line (String). In order to extract each data field, call the `useDelimiter()` method on the `Scanner` to change the delimiter from whitespace characters to a single comma. Now when we iterate with the scanner, each call to `next()` will return the fields (tokens) in the following order (lastname, firstname, id, grade). Use the `next()` method on the Scanner to retrieve each field as a string and store it to a local variable.

Part 3: Create Student objects and add to ArrayList

Work through `TODO` comment 5 in `GradeBook.java`. Using the local variables initialized from the student record field data, create a new Student object and use the `setGrade()` method to assign the grade data.

```
    Student student = new Student(firstName, lastName, id);
    student.setGrade(grade);
```

Finally, add the student to the `gradebook` `ArrayList`.

Part 4: Print the gradebook

Work through `TODO` comment 6 in `GradeBook.java`. Use a foreach loop and the `toString()` method from the Student object to print the gradebook to the console as shown below:

```
Barney Stinson has a D.
Robin Scherbatsky has an A.
Ted Mosby has an A.
Lily Aldrin has a B.
Marshall Eriksen has a B.
Gary Blauman has an F.
Stella Zinman has a C.
```

Terminology Identification

In your code add comments identifying examples of the following: parsing, token, delimiter, scanner, throwing (or catching) an exception. These should be identified with an @keyterm tag within the comment.

Code Review

When you are finished with this activity, pair up with a classmate and review each other's code to make sure it meets all the requirements.

Submission

After completing the assignment, use the assignment link in Canvas and follow the submission instructions there. You will upload your .java files and put your reflection in the "Comments" box.

Reflection Requirements

Write a one paragraph reflection describing your experience with this activity. The reflection should also include the name of your code review partner AND something interesting you found in their code. Please review the activity rubric for more details.