CS49J
Fall Semester 2023
Project

Description

For project #1, you had an opportunity to work with the Junit Test Framework. Now that you have some familiarity with it, you will now get chance to apply what you know. For this project, project #2, not only will you use the JUnit Test Framework, but you will also develop source code to read rows and columns of data from a File. What kind of data file?

The file you will be referring to is a file called pokemon.csv.

- Study the contents of this file and become familiar with how the data is layed out.
- The first row is the column headers for each data in each row
- Each row (line) is a character

High Level Agenda

Step 1: Create the source code for reading a data file and create a list of data objects

Step 2: Create some methods to gather statistics of the data

Step 3: Create JUnit tests for the statistical methods

Details

Part 1: Source Code

A) Create a .csv file reader class called CsvReader
    a. Method: public boolean readFile(String fileName)
        i. Return value is true if file was read okay, false otherwise (if this returns false, then something is not right and so you have to fix it. Make this method bullet proof as best as you can).
        ii. This method should store Character objects in a Hashset member variable in this class
        iii. Ideally, you should create a new Character object (for each row data line) in this method and then for each line (String) read, pass the information to the Character class method (this should be the constructor of the Character class)
    b. Method: public Hashset<Character> getCharacterSet()

i. This method allows the retrieval of the entire HashSet of Character objects that were created
ii. returns a hash set of Character objects (these are based on the rows of data)

B) Create a character class called Character
    a. Constructor: public Character(String dataLine)
        i. When creating a new character, this constructor should take in the row dataLine and then it should perform the line parsing i.e. using the split(..) command to get the individual data pieces from the data line.
        ii. The method should initiate the setting of the column data (which should be the member variables in this class). Note: use the same name spelling as the name that are in the column headers of the I/O file (pokemon.csv)
            1. The column header names is the first row of information.
    b. You will add any other additional supporting methods to get this class to work right for you. So you will decide what is needed and what is not based on the needs for the project.
    c. You should have a handful of get methods to retrieve data.
    d. NOTE: you can look at the information in PART 2 and this will give you a sense of which variables to create in your Character class.

C) Create a runner class (with main) called Pokemon
    a. Method: create a main class
        i. Creates the CsvReader object and calls the CsvReader.readFile method
        ii. Assuming CsvReader.readFile() runs okay, then get the HashSet of Character objects from CsvReader object (use the getCharacterSet() method from the CsvReader class.
        iii. With the data set, you will have base set of information to start working with for Part 2
            1. NOTE: good idea to print out some properties of the object just to ensure you can loop through the set. This is debug output only.

Part 2: Data Gathering

For this part, you will create a few methods that give you information about the set. In the Pokemon class in Part 1, you will add the following methods and test them in the main method:

A) Method: public HashSet<Character> getHitPointList(int maxHP, HashSet<Character> baseSet)
    a. This method will return a list of Pokemon characters that have a range of HP (hit points) from 0 to maxHP). The HP is the “hp” column in the .csv file.
    b. The baseSet is the set you got from Part C.a.ii
B) Method: public int getIsLegendaryCount(HashSet<Character> baseSet)

a. This method returns the total count of characters that are marked as legendary (This is the “is_legendary” column in the .csv file)

C) Method: public TreeSet<Character> getCharacterByFirstLetter(char firstLetter, HashSet<Character> baseSet)
   a. As the name mentions, this method should return a TreeSet that orders the Character objects by their name.
   b. Example: If you entered the letter ‘C’ as the input parameter, this method should return all the characters with names that start with the letter ‘C’ or ‘c’
   c. The name column is called “name” in the .csv file.
D) NOTE: For this part, you should run a couple of test runs to look at the output of the return values from each of these methods, just to make your code is working.

Part 3: JUnit testing

For this part, you just need to test the three methods in part 2. This part will be up to you on how you want to test the return. A positive and negative test for each method is all you need. Don’t create more.

Deliverables

- Capture screen shots of your runs
  - One screen shout should be a list of the Character data set that comes from the CsvReader.getCharacterSet() method.
    - The entire list is not necessary
    - Just a screenshot showing the beginning of the set being printed out
      - You should print out the name, hit points, and the is_legendary information
  - One screen shot showing the results after each JUnit test.
    - A screen shot of the positive test and another screen shot of the negative test for each method