SWEN20003
Object Oriented Software Development
Workshop 10 (Solutions)

Eleanor McMurtry

Semester 2, 2020

Workshop

Questions

1. Create a `CsvException` class to represent errors that may occur when handling comma-separated value (CSV) data. It should inherit from `Exception`, and should contain a descriptive message. For example, you may choose to include a file name.
2. Create the following exception classes inheriting from `CsvException`; they should have their own descriptive messages.
   (a) `TooManyEntriesException`: when a row has more entries than the header row
   (b) `NotEnoughEntriesException`: when a row has fewer entries than the header row
   (c) `UnmatchedQuoteException`: when a quotation mark `"` at the start of an entry is not matched by a quotation mark at the end of the entry

   Solution:

```
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CsvException extends Exception {
    public CsvException(String filename, String reason) {
        super("Error processing " + filename + (reason.length() > 0 ? ": " + reason : ""));
    }
}

public class TooManyEntriesException extends CsvException {
    public TooManyEntriesException(String filename, int row) {
        super(filename, "too many entries in row " + row);
    }
}

public class NotEnoughEntriesException extends CsvException {
    public NotEnoughEntriesException(String filename, int row) {
        super(filename, "not enough entries in row " + row);
    }
}

public class UnmatchedQuoteException extends CsvException {
    public UnmatchedQuoteException(String filename, int row) {
```

```
        super(filename, "unmatched quote in row " + row);
    }
}
```

3. Using the above exception classes, write a method

   ```
   List<List<String>> readCsv(String filename) throws IOException, CsvException
   ```

   that reads a CSV file and returns its contents as a two-dimensional `ArrayList`. The outer dimension should be the rows of the file, and the inner dimension should be the entries within each row. If any of the above exceptional cases are encountered, you should throw the appropriate exception.

   Solution: note this handles quotation marks a little more carefully than was really required.

```
public static List<List<String>> readCsv(String filename) throws IOException, CsvException {
    List<List<String>> result = new ArrayList<>();
    String contents = Files.readString(Path.of(filename), StandardCharsets.US_ASCII);
    String[] allRows = contents.split("\n");
    String header = allRows[0];
    String[] rows = Arrays.copyOfRange(allRows, 1, allRows.length);

    // Not really completely correct but a decent approximation.
    int columnCount = header.split(",").length;

    int rowNum = 1;
    for (String row : rows) {
        boolean inQuote = false;
        List<String> parsedRow = new ArrayList<>();
        String soFar = "";

        for (int i = 0; i < row.length(); ++i) {
            char at = row.charAt(i);
            if (at == '"') {
                inQuote = !inQuote;
            }
            if (at == ',') {
                if (!inQuote) {
                    parsedRow.add(soFar);
                    soFar = "";
                } else {
                    soFar += ",";
                }
            } else {
                soFar += at;
            }
        }

        // Check for matched quotes
        if (inQuote) {
            throw new UnmatchedQuoteException(filename, rowNum);
        }

        // Check for entry count
        if (parsedRow.size() < columnCount) {
            throw new NotEnoughEntriesException(filename, rowNum);
        }
        if (parsedRow.size() > columnCount) {
            throw new TooManyEntriesException(filename, rowNum);
        }
        result.add(parsedRow);
    }
```

```
    return result;
}
```

4. Write JUnit test cases for the CSV reader. You should test the correct case with the below test case:

```
student,mark
Alice,90
Bob,65
Carol,72
```

Write one unit test for each of the possible exceptions. Hint: use this annotation when an exception is expected:

```
@Test(expected=NotEnoughEntriesException.class)
```

Solution:

```
import org.junit.Test;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import static junit.framework.TestCase.*;

public class CsvTest {
    @Test
    public void testValid() {
        List<List<String>> result = new ArrayList<>();
        List<String> row1 = new ArrayList<>();
        row1.add("Alice");
        row1.add("90");
        List<String> row2 = new ArrayList<>();
        row2.add("Bob");
        row2.add("65");
        List<String> row3 = new ArrayList<>();
        row3.add("Carol");
        row3.add("72");
        result.add(row1);
        result.add(row2);
        result.add(row3);

        try {
            assertEquals(result, CsvException.readCsv("valid.csv"));
        } catch (IOException | CsvException e) {
            e.printStackTrace();
            // fail on throw
            fail();
        }
    }

    @Test(expected=NotEnoughEntriesException.class)
    public void testInvalid() throws IOException, CsvException {
        CsvException.readCsv("invalid.csv");
    }

    // etc.
}
```

invalid.csv:

```
student,mark
Alice,90
Bob,65
Carol
```