SWEN20003
Object Oriented Software Development
Workshop 10

Eleanor McMurtry

Semester 2, 2020

Workshop

This week, we are reviewing **exceptions** and **testing**.

1. Exceptions are the standard Java way to handle **runtime errors**. They may be either **unchecked** (such as null references, array overflows, division by zero) or **checked** (such as IO errors).
2. Checked exceptions **must** be either caught or declared in the method header with the `throws` keyword.
3. Exceptions fill a similar role to return codes in C (e.g. for `malloc`).
4. **Automated testing** makes it easier to detect **regressions** (when a new feature breaks an existing one) among other things.
5. A **unit test** tests a single **unit** (e.g. method or class).

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
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.
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)
   ```