Für die Entwickler

Klassen

App:	Start der Applikation.
LottoApplikation:	Verwaltet die Annahme des Userinputs und Kommunikation mit dem User über die Konsole (Abfrage, Fehlermeldungen, Presentation der Tipps etc.). Verfügt über ein Lottomodel.
Lottomodel:	Verwaltet die Unglückszahlen und Generierung der Tippreihen. Verfügt über GameLotto und GameEurojackpot für die Generierung der Tippreihen und CSVReader und CSVWriter für die Verwaltung der Unglückszahlen.
Game:	Oberklasse, welche die Logik für die Generierung der Tippreihen enthält. Generiert abhängig von den Parameter n Zahlen von 0 bis m. Die Unterklassen definieren die Parameter und die Anzahl der Generierungen.
GameLotto:	Unterklasse von Game. Enthält nötige Parameter, welche mithilfe der Oberklasse zur Generierung der Tipps für Lotto benutzt wird.
GameEurojackpot:	Unterklasse von Game. Enthält nötige Parameter, welche mithilfe der Oberklasse zur Generierung der Tipps für Eurojackpot benutzt wird.
CSVReader:	Liest die Unglückszahlen aus der CSV und sendet diese als List<Integer> zurück.
CSVWriter:	Erhält als Parameter die neuen Unglückszahlen, welche genutzt werden um die aktuelle Datei zu ersetzen.
LottoLogger:	Erstellt eine Textdatei unter demo/src/log/, welche im Laufe der Applikation erweitert wird. Alle Klassen haben Zugriff zu dem Logger, sodass jede Klasse ihre eigenen Nachrichten in die Logfiles schreiben können.

Mögliche Verbesserungen

Stellenweise fehlt eine geeignete Exceptionhandeling. Einige catchblöcke sind leer oder verfügen über einen Kommenter der auf schlechtes Exceptionhandeling hinweist (Es fehlt

mir an Wissen und Erfahrung, wie man richtig mit Exceptions umgeht).
Die Init-methoden in dem CSVReader und CSVWriter können in den Konstruktor geschoben werden, da sonst bei jedem Schreib- und Lesezugriff neue Klassen erstellt werden.
Die generierten Tippreihen werden als String und nicht als List<Integer> returnt, da Eurojackpot sonst zwei Listen returnen müsste. In dieser Applikation ist es kein Problem, jedoch könnte es mit zunehmender Anzahl von unterschiedlichen Spielen zu einem Problem werden, sodass es in so einem Fall Sinn machen würde, eine List<List<Integer>> zu returnen oder eine spezielle Klasse dafür zu entwickeln.

Tests

Verwendet eine leere csv (test.csv) für das Testen von Schreib- und Leseeingriffen und eine unluckyNumbersTest.csv mit Unglückszahlen um die Generierung zu testen.
Die Tests arbeiten ohne den Logger, da zu viele Logfiles erstellt werden würden. Die Test verfügen über mehrere Asserts, damit das Testen auf einer leeren Liste von Unglückszahlen gewährleistet werden kann oder das Gegenteil bewiesen wird.

Mögliche Verbesserungen

Es wurden im Rahmen der Bewerbungsaufgabe nicht alle Tests geschrieben die für ein echtes Projekt verwendet werden müssten, sodass das Testen von nicht getesteten Methoden (vorallem im LottoModel) notwendig wäre. Hinzufügend werden nicht alle Anforderungen getestet (z.B. die maximale Anzahl an Unglückszahlen).
Die Anzahl der Parameter im CsvSource kann erhöht werden oder sogar generiert werden. Um Code zu sparen und die Tests wartbarer zu machen, kann man auf den String Parameter, welche anschließend in eine List<Integer> umgewandelt wird, verzichten und stattdessen z.B. globale Listen mit zu testenden Zahlen verwenden.