Die Hausaufgaben müssen von jedem Studierenden einzeln bearbeitet und abgegeben werden. Für die Hausaufgabe sind die aktuellen Informationen vom Blog https://seblog.cs.uni-kassel.de/ws2324/programming-and-modelling/ zu berücksichtigen.

Abgabefrist ist der 18.01.2024 - 23:59 Uhr

Abgabe

Wir benutzen für die Abgabe der Hausaufgaben Git. Jedes Repository ist nur für den Studierenden selbst sowie für die Betreuer und Korrektoren sichtbar.

Für die Hausaufgabe benötigst du **kein neues** Repository. Es wird das gleiche Repository benutzt, das bereits in Hausaufgabe 6 angelegt wurde. Dieses kann über folgenden Link erstellt werden, falls nicht bereits geschehen:

https://classroom.github.com/a/kl_i4rdv

Nicht oder zu spät gepushte (Teil-)Abgaben werden mit 0 Punkten bewertet!

Abgaben, die nicht lauffähig sind, werden mit 0 Punkten bewertet!

Projekte, deren GUI nicht mit FXML-Dateien umgesetzt sind, werden mit 0 Punkten bewertet!

Aufgabe 1 - Speichern und Laden (12P)

In TinyTransport soll es ermöglicht werden, den aktuellen Spielstand abzuspeichern und beim nächsten Öffnen des Spiels fortzusetzen. Das Speichern des Spiels soll automatisch passieren, wenn der Nutzer das Spiel über den Close-Button der Anwendung schließt. Des Weiteren soll die Spiel-Oberfläche erweitert werden. Füge einen Button zum Starten eines neuen Spiels hinzu (der aktuelle Spielstand wird damit gelöscht).

Ergänzungen in build.gradle

Füge die folgenden Dependencies deiner build.gradle-Datei innerhalb des `dependencies`-Blocks hinzu:

```
// https://mvnrepository.com/artifact/com.squareup.retrofit2/converter-jackson
implementation group: 'com.squareup.retrofit2', name: 'converter-jackson', version: '2.9.0'

// https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core
implementation group: 'com.fasterxml.jackson.core', name: 'jackson-core', version: '2.13.3'
```

Speichern und Laden

Implementiere das Speichern und Laden analog zur Vorlesung/Übung, unter Beachtung der oben beschriebenen Nutzerinteraktionen. Die Daten sollen im Ordner `data` in der Datei `gameData.json` gespeichert werden. Es reicht aus, die Daten des HQs und seine Autos zu speichern.

Aufgabe 2 - Spiel vervollständigen (8P)

In dieser Aufgabe musst du die letzten Funktionalitäten für das Spiel implementieren.

Ablaufen von Orders

Die verbleibende Zeit von angenommenen Aufträgen soll in geeigneter Form dargestellt werden. Beispielsweise können die verbleibenden Sekunden oder ein schrumpfender Balken angezeigt werden. Zusätzlich muss auch in der Logik zur Annahme/Abarbeitung von Orders beachtet werden, ob der Auftrag noch gültig oder schon abgelaufen ist.

Speedlimit beachten

In der Spiellogik bei der Abarbeitung von Orders soll nun auch das `speedLimit` der Straßen beachtet werden. Zusätzlich soll auf der Karte bei jeder Straße das zugehörige Speedlimit angezeigt werden.

Animation der Autos

Die Autos sollen außerdem nicht mehr von einer Location zur anderen springen, sondern sich durch eine Animation von A nach B bewegen, entsprechend des Speedlimits.

Korrektes Abmelden der PCLs

In der Übung wurde das Vorgehen zum Abmelden von PropertyChangeListenern korrigiert. Passe deinen Code entsprechend an, falls noch nicht geschehen.

Aufgabe 3 - Final Testination (6P)

Zu guter Letzt muss der kritische Pfad der Anwendung getestet werden. Wie in der Übung gezeigt soll ein **Seed** genutzt werden, um das Testen zu erleichtern und kontrollierte Zufallswerte zu erhalten.

Erweitere dafür den AppTest mit folgenden Aktionen:

- Eine Stadt anklicken, in der eine Order vorhanden ist
- Den Auftrag über den entsprechenden Button annehmen
- Prüfen, ob die Endlocation beim Auto angezeigt wird
- Prüfen, ob nach Abschluss des Auftrags der erhöhte Geldbetrag des HQs angezeigt wird

Alle Tests (alt/neu) müssen nach wie vor funktionieren, sollte dies nicht der Fall sein, wird diese Aufgabe mit 0 Punkten bewertet!

Abschluss

Nach Implementierung der Funktionalität dieser Hausaufgabe ist die Anwendung TinyTransport für dieses Semester **abgeschlossen**. Weitere Features dürfen bei gegebener intrinsischer Motivation umgesetzt werden, sind jedoch nicht Pflicht. In diesem Fall sollte jedoch der Commit, welcher die Aufgabenstellung abschließt, deutlich markiert werden, damit eventuelle Erweiterungen nicht zu Abzügen führen.

In der kommenden Hausaufgabe für Studiengänge mit 6CP wird eine neue Anwendung begonnen.

Für die Studierenden der Mechatronik (4CP) ist diese (Hausaufgabe 9) die letzte reguläre Hausaufgabe. In der kommenden Woche wird das Aufgabenblatt zum Projekt für Studierende der Mechatronik veröffentlicht.