|  |  |
| --- | --- |
|  | **ETCS@EBD** |
|  | TMS Simulator  Entwicklerdokumentation |
|  |  |
|  |  |
|  | Dokumentation, v0.2  Status: Draft, Stand: 15.01.2020 16:15  Autor: Werner Iberl |
|  |  |



|  |  |  |  |
| --- | --- | --- | --- |
|  | **Bearbeitung Editorial** | **Qualitätsprüfung Reviewer** | **Freigabe Approval** |
| **Name** | Werner Iberl | Frederik Düpmeier | Frederik Düpmeier |
| **Position/Rolle** | Softwareentwickler | Projektleiter | Projektleiter |
| **Datum** | 15.01.20 |  |  |
| **Unterschrift** | gez. Iberl | gez. Düpmeier | gez. Düpmeier |

Versionshistorie

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vers.** | **Datum** | **Kap.** | **Beschreibung** | **Autor** |
| 0.2 | 15/01/20 |  |  | WI |
| 0.1 | 09/01/20 |  | Erster Entwurf | WI |

1 Beschreibung des TMS Simulators 6

2 Struktur und Funktionalität 7

2.1 Allgemeiner Programmstart 7

2.1.1 Main-Entry-Class PerverEm 7

2.2 Model-Klassen 8

2.2.1 Betriebsstelle Wrapper 8

2.2.2 MitarbeiterComparator 9

2.2.3 MitarbeiterWrapper 9

2.2.4 Model 9

2.2.5 PerverTableModel 10

2.2.6 RunTimeData 10

2.2.7 TableEntry 11

2.2.8 TrainTurnaroundModel 12

2.3 Repository-Klassen 12

2.3.1 DefaultRepo 12

2.3.2 TrainTurnaroundRepo 13

2.4 Datenbankenklassen Entitäten 13

2.5 Viewklassen 13

2.5.1 AutoCompletion 13

2.5.2 ComboBoxCellEditor 14

2.6 Viewklassen im Main-Package 14

2.6.1 MainFrame 14

2.7 Controllerklassen 15

2.7.1 PerverDynController 15

2.7.2 PerverEmController 15

2.8 Em-Klassen 16

2.8.1 HibernateUtil 16

2.8.2 PerverDYN\_EM 16

2.8.3 PerverTaetigkeitenEM 16

3 Nutzung des TMS Simulators 18

# Beschreibung des TMS Simulators

Der TMS Simulator hat eine Oberfläche die ab Januar auf das PlanPro-Format fußt. Dieses Format liefert den Gleisplan. Es können Stellwerkszustände des Gleisplanes geändert werden. Es können Fahrzeuge angelegt werden. Diese erhalten ein Laufweg mit Farbe und Eigenschaft wie Fahrzeugtyp über eine Oberfläche. Es soll außerdem ein Geschwindigkeitsprofil für die Strecke erstellt werden können. Die SmartLogic überprüft, ob die Fahrerlaubnis zulässig ist.

# Struktur und Funktionalität

Es werden die Schnittstellen des User Interfaces und zur Datenbank explizit beschrieben.

## Programmstart im Allgemeinen Modus und auf Szenario Basis

Das Programm hat einen normalen Modus, in der der Nutzer das System verwendet. Des Weiteren wird ein Batchmodus unterstützt. Dieser ermöglicht, dass das RBC Szenarios erhält und demnach eine Teststrecke für den Zug ansteuert.

### Main-Entry-Class MainTmsSim

Die Klasse MainTmsSim ist der Eintrittspunkt zum TMS-MA-Generator.

|  |  |
| --- | --- |
| public enumTmsMode | Normal Mode ist der Modus zum Definieren der Fahrstraßen multipler Züge.  EBD Mode gibt Szenariobasiert informationen an das RBC |
| public staticvoid main(String[] args) | Startet das Programm mit der Oberfläche des Main Frames MainFrame(s.u.). |
| public staticJFrame MainFrame | Dieses Fenster ist das Hauptanwendungsfenster und wird genau einmal gehalten. |
| public staticJFrame createTmsFrame(TmsMode Mode) | Erzeugt den MainFrame in Abhängigkeit zum Modus, der angegeben wurde. |
| public staticJButton genCloseButton(JWindow CurrentWindow, String sButtonName) | Erzeugt einen Button, der das angegebene Fenster „CurrentWindow“ schließt. |
| public staticFlow.Subscription MaSubscription | Es gibt diese Subscription um über Plan-Data also graphische Elemente in der Topologie, Änderungen mitzuerhalten. |
| public staticMainGraphicPanel TrackPanel | Die Oberfläche auf der Track gezeichnet wird. |
| public staticvoid updateSubViews() | Zeichnet das Panel neu. |

## Model-Klassen

Model-Klassen werden als Teil der Fachkonzeptschicht angesehen.

### ETCS\_DISTANCE

Diese Klasse ETCS\_DISTANCE gibt eine ETCS-Nachricht vor.

|  |  |
| --- | --- |
| publicshort sDistance | 10cm, 1m, 10m orientiert sicha an Q\_Scale mit 15 bits. Dadurch können Wert von 0 cm bis zu 327670 km gefasst werden. |

### ETCS\_GRADIENT

Diese Klasse stellt einen Höhengradient dar.

|  |  |
| --- | --- |
| publicbyte bGradient | Kann Werte von 0 bis 255% in 1% Schritten annehmen. |

### ETCS\_SPEED

Diese Klasse stellt eine Geschwindigkeitsgröße dar.

|  |  |
| --- | --- |
| publicbyte bSpeed | 7 bits geben in 5 km/h Schritten einen Wertebereich vor. Dieser geht von 0 bis 600km/h. |

### ETCS\_TIMER

Die Klasse stellt Zeitzusammenhänge dar.

|  |  |
| --- | --- |
| publicshort sTimer | 0 bis 1022 Sekunden werden in 1 Sekundenintervalle darstellbar gemacht. Der Wert 1023 steht für Unendlich. |

### NC\_CDDIFF

Diese Klasse stellt in Zusammenhang der Cant Deficiency erlaubten Geschwindigkeitsprofile dar.

|  |  |
| --- | --- |
| publicbyte bDiff | 0-15 Kategorien |

### NC\_DIFF

Diese Klasse stellt den Zusammenhang der Fracht zum erlaubten Geschwindigkeitsprofil dar.

|  |  |
| --- | --- |
| publicbyte bDiff | 0-15 Kategorien |

### Q\_SCALE

|  |  |
| --- | --- |
| publicenum Q\_SCALE | Enumeration der Kategorien:  SCALE\_10\_CM, SCALE\_1\_M, SCALE\_10\_M, SPARE |

## Ma-Klassen

Ma-Klassen dienen dazu Ma-Request erstellen zu können.

### Chainage Klasse

Chainage beschreibt die Streckenkilometrierung.

|  |  |
| --- | --- |
| publicChainage(int iMeters) | Gibt die Meter dieser Container-Klasse vor. |
| publicint getiMeters() | Holt den Wert aus dem Container. |
| publicString toString() | Stellt die Textausgabe mit Maßeinheit dar. |

### DangerPoint Klasse

Es gibt verschieden Gefahrpunkte auf einer Strecke. Diese Klasse DangerPoint bildet Gefahrpunkte ab.

|  |  |
| --- | --- |
| ETCS\_DISTANCE d\_OL | Entfernung von End of Authoritiy zum Danger Point. |
| ETCS\_SPEED v\_RELEASEDP; | Zulässige Geschwindigkeit hinsichtlich der Gefahr. |
| publicDangerPoint(ETCS\_DISTANCE d\_OL, ETCS\_SPEED v\_RELEASEDP) | Konstruktur mit sämtlichen Informationen. |

### EoA Klasse

Das Ende der gültigen Fahrstraße wird über die End-Of-Authority definiert. Das übernimmt die EoA Klasse.

|  |  |
| --- | --- |
| int v\_EMA | Erlaubte Geschwindigkeit in dieser Authority. |
| boolean q\_ENDTIMER; | Gibt es einen Timer bis zum Ende der Sektion. |
| Q\_SCALE q\_scale; | Maß ob 10cm 1m 10m oder spare für das Feld d\_ENDTIMERSTARTLOC angesetzt wird. |
| ETCS\_DISTANCE d\_ENDTIMERSTARTLOC; | Entfernung von Timer-Start-Ort hin zur End of Authority. |
| ETCS\_TIMER t\_ENDTIMER; | Valide Zeit bis zum Erreichen des Endes der Fahrstraße. |
| boolean q\_DANGERPOINT; | Gibt es eine DangerPoint zu beachten true/false. |
| DangerPoint dangerPoint | Referenz auf eventuellen DangerPoint. |
| boolean q\_OVERLAP | True: erkennt eine Überlappung; false es gibt eine Überlappung. |
| Overlap overlap | Verweis auf Überlappung. |
| boolean isShunting | Gibt es den Shunting Mode.  Indiziert für das EBD, dass das Fahrzeug die letzte EoA vom TMS erhalten hat. |
| publicEoA(Chainage chainage, TrackElement trackElement, SectionOfLine lineSection) | Konstruktor mit notwendigen Informationen.  Shuning wird in dem Konstruktor deaktiviert (false). |

### GeoCoordinaten Klasse

Diese Klasse gibt Informationen über den Standpunkt x,y-Koordinate und Höhe

|  |  |
| --- | --- |
| double x | x-Koordinate |
| double Y | y-Koordinate |
| double height | Höhenangabe |

### GradientProfile Klasse

Gibt Angaben über das Neigungsprofil der Strecken.

|  |  |
| --- | --- |
| privateMovementAuthority movementAuthority; | Referenz zur MA, zu der das Profil gehört. |
| privateList<GradientSegment> segmentList | Die GradientenSegmente die das Profil ausmachen. |
| publicGradientProfile(MovementAuthority movementAuthority) | Konstruktor mit MA, und leerer Segment Liste s.o.. |
| publicvoid addSegment(GradientSegment GS) | Fügt ein Gradienten Segment der Liste hinzu, falls dieses Segment noch nicht darin sei. |

### GradientSegment Klasse

Ein Segment eines Gradienten-Neigungsprofils.

|  |  |
| --- | --- |
| privateETCS\_GRADIENT g\_A; | Neigungsgrad 0 – 254 % |
| privateboolean q\_GDIR; | Erkennt eine Steigung bei true, | ein Gefälle bei false |
| publicGradientSegment(SpotLocation begin, SpotLocation end, ApplicationDirection direction) | Konstruktor mit Beginn und Ende als Punkt und ob dr Verlauf in Fahrtrichtung verläuft. |
| publicvoidsetGradient(ETCS\_GRADIENT G, boolean isUphill) | Speichert die Daten des Gradienten. |

### LinearLocation Klasse

Eine Linear Location stellt einen Verlauf dar zum Beispiel einen Teil einer Fahrstraße.

|  |  |
| --- | --- |
| privateSpotLocation begin; | Beginn des Verlaufs als Spot. |
| privateSpotLocation end; | Ende des Verlaufs als Spot. |
| privateApplicationDirection | Ist der Verlauf in der gleichen Laufrichtung als der Streckenverlauf |
| publicLinearLocation(SpotLocation begin, SpotLocation end, ApplicationDirection direction) | Konstruktor mit allen Daten. |

### MARequest Klasse

Stellt einen Request dar. Es wird durch diese Klasse die Fahrstraße an das RBC gereicht.

|  |  |
| --- | --- |
| privateMovementAuthority ma; | Die Eigentliche Fahrstraße. |
| privateTrainMovement train; | Zugdaten. |
| privateTmsController tms, | Verweis auf die Logik des TMS. |
| privateRoute route; | Routeninformationen. |

### MitarbeiterComparator

Diese Klasse macht Mitarbeiter vergleichbar (zum Beispiel hinsichtlich Sortierung in der Tabelle).

|  |  |
| --- | --- |
| @Override publicint compare(MitarbeiterDO o1, MitarbeiterDO o2) | Diese Methode legt den Vergleich unter Mitarbeitern fest. Es wurde in der Definition der Nachnahme gewählt. Genauer die compareTo-Methode des Nachname-String-Objektes. |

### MitarbeiterWrapper

Diese Klasse macht Mitarbeiten zum Beispiel in ComboBoxen anzeigbar.

|  |  |
| --- | --- |
| @Getter @Setter privateMitarbeiterDO worker; | Der gekapselte Mitarbeiter aus der Datenbank. |
| publicMitarbeiterWrapper(MitarbeiterDO worker); | Beim Konstruktor Aufruf wird der zu kapselnde Mitarbeiter aus der Datenbank bestimmt. |
| @Override publicString toString(); | Die Methode gibt den Mitarbeiternachnamen Kommagetrennt vom Vorname zurück.  Die Methode gibt den Anzeigenamen zum Beispiel für Combo-Boxen vor. |

### Model

Diese Klasse bildet das Fachkonzept für die Tabelle aus der Datenbank ab.

|  |  |
| --- | --- |
| @Getter @Setter privateMitarbeiterWrapper[] kiN; | Eine Liste der Kundeninformationsbelegschaft. |
| @Getter @Setter privateMitarbeiterWrapper[] tf; | Eine Liste der Triebfahrzeugführer. |
| @Getter @Setter privateString[] fz; | Eine Liste aller Fahrzeugnamen. |
| @Getter @Setter privateArrayList<TableEntry> entries; | Eine Liste von Tabelleneinträgen. Jede Listeneintrag entspricht einer Tabellenzeile. |
| @Getter (lazy = true) private static finalModel instance; | Eine Singleton-Struktur für diese Modellklasse. |

### PerverTableModel

Das Model der Tabelle die verwendet wurde.

|  |  |
| --- | --- |
| publicint getRowCount() | Die Zeilenanzahl aus der Modelklasse aus 2.2.4 |
| publicArrayList<TfWorkerActualDYN\_DO> tfWorkerDynList | Alle bereits selektierten Triebfahrzeugführer. Diese treten in der Ist-Spalte der Triebfahrzeugführer auf. |
| publicArrayList<KinWorkerActualDYN\_DO> kinWorkerDynList | Alle bereits selektierten Kundeninformations-Personen. Diese treten ebenfalls in der Ist-Spalte auf. |
| @Override public voidsetValueAt(Object aValue, int rowIndex, int columnIndex) | Setzt einen Wert in der Tabelle nämlich Objekt aValue |
| public intgetClumnCount() | Definiert die Anzahl der Spalten (8) |
| publicObject getValueAt(int rowIndex, int columnIndex) | Definiert den Tabelleneintrag. Es wird das zugrunde liegende Model aus 2.2.4 verwendet. |
| @Override public String getColumnName(int column) | Gibt die Spaltennamen vor. Diese Einträge sind statisch. |
| @Override publicboolean isCellEditable(int rowIndex, int columnIndex) | Gibt die Spalten an die der Nutzer verändern können soll. Dies sind die drei Ist-Spalten. |
| @Override publicClass getColumnClass(int c) | Definiert den Datentyp einer Spalte. E.g. Integer, MitarbeiterWrapper und String. |

### RunTimeData

Wenn ein Nutzer zur Laufzeit Einträge macht werden diese in eine weitere Datenbank geschrieben. Die Klasse RunTimeData macht diese Daten übertragbar.

|  |  |
| --- | --- |
| publicArrayList turnList; | Eine Liste von Einträgen über abweichende Wenden. |
| publicArrayList tfList; | Eine Liste von „Ist-Triebfahrzeugführer“ in Zusammenhang mit der Leistung. |
| publicArrayList kinList; | Eine Liste von „Ist-Kin-Mitarbeiter“ in Zusammenhang mit der Leistung. |
| publicArrayList trainList; | Eine Liste von „Ist-Zügen“ in Zusammenhang mit der Leistung. |

### TableEntry

Eine Tabellenzeile wird als TableEntry dargestellt.

|  |  |
| --- | --- |
| @Getter @Setter privateInteger zugNr; | Eine Liste von Einträgen über abweichende Wenden. |
| @Getter @Setter privateString additionalIndex | Es kann sein, dass eine Leistung mehrere Zeilen in der Tabelle ausmacht, deswegen gibt es einen weiteren Index; additionalIndex. |
| publicint getIntegerAdditionalIndex() | Diese Methode konvertiert den String des additionalIndex in eine Integer-Zahl. Ist der additionalIndex eine leere Zeichenkette wird 0 zurückgegeben. |
| @Getter privateMitarbeiterWrapper WorkerKiIst | Ist Wert des Kundeninformationspersonals als Objekt. |
| @Getter privateString kiNSoll | Soll Wert eines Kundenbetreuenden aus der Stammdatenbank als String. |
| @Getter privateMitarbeiterWrapper WorkerTfIst | Ist Wert des Zugfahrers als Objekt |
| @Getter privateString tfSoll | Soll Wert eine Zugfahrenden aus der Stammdatenbank als String. |
| @Getter @Setter privateString fzIst | Fahrzeugnummer als Istwert für außerplanmäßige Fahrzeuge |
| @Getter privateString fzSoll | Fahrzeugnummer als Sollwert für geplantes Fahrzeug |
| @Getter @Setter privateSchichtDO schichtTf | Schicht für Schichtdialog des Fahrzeugführenden. |
| @Getter privateSchichtDO schichtKiN | Schicht für Schichtdialog der Kundeninformation. |
| public staticList<TableEntry> balanceTat(Integer zugNr, List<MitarbeiterDO> tfWorkers, List<TaetigkeitDO> tfTat,  List<MitarbeiterDO> kinWorkers,  List<TaetigkeitDO> kinTat, boolean tfIsDominant, ArrayList<FahrzeugDO> fahrzeuge) | Generiert für eine Zugnummer alle dazugehörigen Tabellenzeilen als TableEntry.  Diese Methode verarbeitet zum Teil eine Datenbankabfrage, die die gesamten Daten der Datenbank in nur einer Abfrage zieht. |
| publicTableEntry() | Default Konstruktor erlaubt Datenbank Entries zu generieren. |

|  |  |
| --- | --- |
| publicvoidsetTfIst(MitarbeiterWrapper WorkerTf, boolean aktualisiereNachfolger, String sollName, Integer zugNr, int additionalIndex, PerverTableModel perverTableModel) | Setzt einen Lokfahrenden und Nachfolgerzeilen der Tabelle. Es ist gewünscht, dass alle TF geändert werden, wenn in der Ist-Zeile eine TF geändert werden. Es werden Rekursiv Nachfolger aktualisiert. Dies wird sowohl von der Datenbank als auch durch den Benutzer angesteuert. |
| publicvoidsetKiNIst(MitarbeiterWrapper WorkerKiN, boolean aktualisiereNachfolger, String sollName, Integer zugNr, int additionalIndex, PerverTableModel perverTableModel) | Setzt die Kundenbetreuenden und die Nachfolgerzeilen der Tabelle. Dies wird sowohl von der Datenbank als auch durch den Benutzer angesteuert. |

### TrainTurnaroundModel

Diese Klasse beschreibt die abweichende Wende als Modelform.

|  |  |
| --- | --- |
| @Getter privateLeistungDO LeiOfTrain; | Die Leistung dieses Models. Da eine abweichende Wende Bezug zu einer Leistung hat. |
| @Getter privateList<BetriebsstelleWrapper> sortedListOfStation | Der Fahrplan des Zuges für diese Leistung. Die Betriebsstellen werden angeführt. |
| @Getter @Setter privateBetriebsstelleWrapper SelectedBetriebsstelle | Die ausgewählte abweichende Wende. Es wird Betriebsstelle der abweichende Wende gespeichert, die ausgewählt wurde. |
| publicString getTrainNr() | Gibt die Zugnummer als Zeichenkette zurück. |
| publicString getTurnTo() | Gibt die letzte Betriebsstelle im Plan zurück. Also die Soll-Wende. |
| publicTrainTurnaroundModel(LeistungDO TrainEffort) | Der Konstruktor zieht die Stationen aus der Datenhaltung eines Repositorium. |

## Repository-Klassen

Repositorien verwalten die Datenbankeinträge im Ram.

### DefaultRepo

Die Hauptklasse die ein Repository generisch aus einer HashMap erzeugt.

|  |  |
| --- | --- |
| publicvoid update(K key, V mapValue) | Setzt in der HashMap den key auf den mapValue |
| publicV getModel(K key) | Holt den Wert aus der HashMap. |
| publicvoid removeKey(K key) | Entfernt den Datensatz aus der HashMap über den Schlüssel. |
| publicList<V> getAll() | Holt alle Einträge aus dem Repository. |

### TrainTurnaroundRepo

Das Zug-Wende Repository für die Dynamischen Daten aus der zweiten Bewegungsdatenbank.

|  |  |
| --- | --- |
| Flow.Subscription RepositorySubscription | Aus dem Flow-Pattern wird hier eine Subscription angewendet, die aus der Datenbank sich aktualisieren kann. |
| @Override  publicvoid onSubscribe(Flow.Subscription subscription) | Bietet Aufruf bereit, um Daten zu erhalten. |
| @Override  publicvoid onNext(RunTimeData item) | Diese Methode wird aufgerufen, wenn aus der Datenbank neue Bewegungsdaten anliegen. |
| @Override  publicvoid onError(Throwable throwable) | Behandelt Fehler. |
| @Override  Public onComplete() | Die Übertragung bleibt erhalten, darum bleibt der Abschluss leer. |

## Datenbankenklassen Entitäten

Entitäten stellen Objekt-Klassen der Datenbank dar.

Im Paket de.disposim.entities befindet sich **FredZuglaufDO.**

Im Paket de.disposim.entities.runtime die neuen Runtime-ObjektKlasse.

**RealTurnAroundStationDYN\_DO** speichert für eine Leistung und Index die abweichende Wende ab.

**TfWorkerActualDYN\_DO** speichert für eine Leistung und Index die Soll-Fahrzeugfahrenden.

**KinWorkerActualDYN\_DO** speichert für eine Leistung und Index die Soll-Kundeniformations-Belegschaft.

**RealTrainDYN\_DO** speichert für eine Leistung und Index geänderte Zugfahrzeuge.

## Viewklassen

View-Klassen werden hier als Komponenten der UI verstanden. Sie befinden sich im Package view.

### AutoCompletion

Ermöglicht die Vervollständigung der ComboBoxen.

|  |  |
| --- | --- |
| publicAutoCompletion(final JComboBox comboBox) | Die AutoCompletion verändert die übergebene Combobox zu einer Select-2 ähnlichen Komponente. |
| publicstaticvoid enable(JComboBox comboBox) | Diese Methode verändert die übergebene Combobox als Selekt-2 Komponente. |
| publicvoid remove(int offs, int len) throws BadLocationException | Ermöglicht entfernen von Text aus der Select-2-Componente. |
| publicvoid insertString(int offs, String str, AttributeSet a) throws BadLocationException | Ermöglicht Text hinzuzufügen um neue Vorschläge für die Auto-Completion zu generieren. |

### ComboBoxCellEditor

Der CellEditor ermöglicht, dass eine ComboBox in der Tabellenzeile als Auswahlfeld existiert.

|  |  |
| --- | --- |
| publicComboBoxCellEditor(JComboBox comboBox) | Verwaltet eine Combobox innerhalb eine Tabelle. |
| publicvoid actionPerformed(ActionEvent e) | Dieser Handler prüft ob die Combobox eine Auswahl erhalten hat und verwandelt die Zelle zurück. |
| publicObject getCellEditorValue() | Gibt ausgewählten Wert der ComboBox zurück. |
| publicboolean stopCellEditing() | Löst ein Event aus, dass die Zelle editiert wurde. |

## Viewklassen im Main-Package

Im Main Package befindet sich auch der Hauptframe.

### MainFrame

|  |  |
| --- | --- |
| @Getter(lazy = true) private staticfinalMainFrame instance | Singleton-Instanz der Klasse. |
| publicvoid onSubscribe(Flow.Subscription subscription) | Dieser Frame kann RunTimeData zugeschickt bekommen. Dafür bietet er eine Subscription an. |
| publicvoid onNext(RunTimeData item) | Diese Methode konsumiert die RunTimeData. |
| @Override public void onError(Throwable throwable) | Verarbeitet Übertragungsfehler |

## Controllerklassen

Controllerklassen verwalten Datenhaltung und Oberflächen.

### PerverDynController

Dieser Controller kümmert sich hauptsächlich um die Bewegungsdaten.

|  |  |
| --- | --- |
| publicstaticvoid updateTrainModel(TrainTurnaroundModel TrainModel) | Speichert eine Abweichende Wende in die Bewegungsdatenbank. |
| public staticvoid removeTrainModel(LeistungDO leiOfTrain) | Löscht eine Abweichende Wende aus der Bewegungsdatenbank. |
| public staticvoid deleteEntityList(ArrayList deletionList) | Löscht alle übergebenen Entitäten. |

### PerverEmController

Dieser Controller verwaltet die Verteilung der RunTimeData.

|  |  |
| --- | --- |
| TrainTurnaroundRepo SubscriberRepo | Dieses Repository kann RunTimeDaten erhalten. |
| public staticvoid removeTrainModel(LeistungDO leiOfTrain) | Löscht eine Abweichende Wende aus der Bewegungsdatenbank für die angegebene Leistung leiOfTrain. |
| public staticvoid deleteEntityList(ArrayList deletionList) | Löscht alle übergebenen Entitäten. |

## Em-Klassen

Die Em-Klassen verwalten den Zugriff auf die Datenbank. Sie befinden sich im Package „em“.

### HibernateUtil

|  |  |
| --- | --- |
| staticSessionFactory sf | Diese Instanz des Hibernate-Session-Generator ermöglich, das Erzeugen von Sessions. Sie wird wie ein Singleton benutzt. |
| public staticSession getHibernateSession() | Erzeugt eine neue Session. |
| public staticvoid closeSessionFactory() | Schließt die SessionFactory beim Beenden des Programmes. |

### PerverDYN\_EM

Die Klasse PerverDYN\_EM verwaltet die Bewegungsdaten.

|  |  |
| --- | --- |
| public staticRunTimeData fetchDynData() | Diese Methode holt sämtliche Bewegungsdaten und gibt diese als RunTimeData-Objekt zurück. |

### PerverTaetigkeitenEM

Diese Klasse verwaltet die statischen Daten der Datenbank.

|  |  |
| --- | --- |
| public staticList<FahrzeugDO> fetchTrainsFromDB() | Holt sämtlich Zugfahrzeugdaten aus der statischen Datenbank. |
| public staticList<BetriebsstelleDO> fetchStationsFromDB() | Holt sämtliche Betriebsstellen aus der statischen Datenbank. |
| public staticList<MitarbeiterDO> fetchWorkerFromDB() | Holt sämtliche Mitarbeiterdaten aus der statischen Datenbank. |
| public staticvoid fillSelectableTrains(Model resultModel) | Holt die Fahrzeugdaten die in der Combobox auswählbar sind und schreibt sie in das (Kapitel 2.2.4). |
| public staticvoid fillAllEntries(Model resultModel) | Schreibt sämtliche statische Daten innerhalb nur einer Datenbankabfrage in das Model. |
| public staticModel getModelFromDb(Model resultModel) | Hasht die Auswahlmitarbeiter und die Auswahlzüge für die ComboBoxen in das Model (Kapitel 2.2.4). Ebenfalls wird die gesamte Tabelle gehasht. |
| publicSchichtDO getSchichtOfWorker(MitarbeiterDO Worker) | Zieht die Schicht eines Mitarbeiters „Worker“ und gibt diese zurück. |
| publicLeistungDO getLeistung(int id) | Holt die Leistung mit dem Primary key „id“. |
| publicList<TaetigkeitDO> getZugbesetzungForLeistung(String leistungNameSearched) | Holt alle Fahrplantätigkeiten mithilfe des Leistungsnamens. Der Leistungsname ist die Id-Kennziffer in Form des String-Parameter. |
| publicList getListOfStationByLeistung() | Holt alle Betriebsstellen für gegliedert für alle Leistungen. Das ist ein Fahrplan für die jeweilige Leistung. Das heißt eine Zeile des Ergebnisses der Query enthält zwei Objekt. Object[0] ist die Betriebsstelle Objekt[1] ist die Id einer Leistung als Integer. |

# Nutzung des Tms Simulator

Die PerverEm startet über die GUI mit javaw.exe, wie üblich für alle Java Programme.

Leider ist im Normal-Start-Mechanismus, der Localhost der angenommene Datenbank-Server.

Deswegen sollte man einen Parameter übergeben. Weiterhin wurde PerverEm mit Java 11 geschrieben.

Deswegen wird eine dementsprechende Java-Version zum Start notwendig.

Ein gültiger Java-Aufruf für das EBD lautet: javaw.exe -jar perverem.jar <DB\_Host>

Es wird empfohlen eine Link über eine Windowsverknüpfung auf diesen Befehl zu erstellen, wodurch manuelle Eingabe umgangen werden kann.