|  |  |
| --- | --- |
|  | **ETCS@EBD** |
|  | TMS Simulator und SL  Entwicklerdokumentation |
|  |  |
|  |  |
|  | Dokumentation, v0.3  Status: Draft, Stand: 12.08.2020 18:02  Autor: Werner Iberl |
|  |  |



|  |  |  |  |
| --- | --- | --- | --- |
|  | **Bearbeitung Editorial** | **Qualitätsprüfung Reviewer** | **Freigabe Approval** |
| **Name** | Werner Iberl | Georg Bolz | Frederik Düpmeier |
| **Position/Rolle** | Softwareentwickler | Teamleiter | Projektleiter |
| **Datum** | 12.08.20 |  |  |
| **Unterschrift** | gez. Iberl | gez. Bolz | gez. Düpmeier |

Versionshistorie

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vers.** | **Datum** | **Kap.** | **Beschreibung** | **Autor** |
| 0.3 | 12/08/20 |  | Javadoc eingepflegt | WI |
| 0.2 | 15/01/20 |  | Strukturierung des Programms | WI |
| 0.1 | 09/01/20 |  | Erster Entwurf | WI |

1 Beschreibung des TMS Simulators 7

2 Beschreibung der SmartLogic 8

3 Struktur und Funktionalität 9

3.1 Die SmartLogic 10

3.1.1 Class SmartLogic 10

3.1.2 Class EventBusManager 13

3.1.3 Enum BlockedArea.BLOCK\_Q\_SCALE 15

3.1.4 Main-Entry-Class MainTmsSim 16

3.2 Model-Klassen 17

3.2.1 ETCS\_DISTANCE 17

3.2.2 ETCS\_GRADIENT 17

3.2.3 ETCS\_SPEED 17

3.2.4 ETCS\_TIMER 17

3.2.5 NC\_CDDIFF 17

3.2.6 NC\_DIFF 17

3.2.7 Q\_SCALE 17

3.3 Ma-Klassen 18

3.3.1 Chainage Klasse 18

3.3.2 DangerPoint Klasse 18

3.3.3 EoA Klasse 18

3.3.4 GeoCoordinaten Klasse 19

3.3.5 GradientProfile Klasse 19

3.3.6 GradientSegment Klasse 19

3.3.7 LinearLocation Klasse 19

3.3.8 MARequest Klasse 20

3.3.9 MaRequestWrapper Klasse 20

3.3.10 MovementAuthority Klasse 20

3.3.11 Occupation 21

3.3.12 Overlap 21

3.3.13 RbcMA Klasse 21

3.3.14 Route Klasse 21

3.3.15 SectionOfLine Klasse 22

3.3.16 SpeedChange Klasse 22

3.3.17 SpeedSegment Klasse 22

3.3.18 SpotLocation Klasse 22

3.3.19 SSP Klasse 23

3.3.20 SvL Klasse 23

3.3.21 TrainMovement 23

3.3.22 Waypoint 23

3.4 Physische MA-Klassen 24

3.4.1 BranchingElement Klasse 24

3.4.2 ControlledTrackElement Klasse 24

3.4.3 DoubleSlip Klasse 24

3.4.4 EdgeOfMap Klasse 24

3.4.5 FlankAreaElement Klasse 24

3.4.6 LocatedNetEntity Klasse 25

3.4.7 Point Klasse 25

3.4.8 Point\_RemoteOperated Klasse 25

3.5 Topologiegenerierung 26

3.5.1 26

4 Nutzung des Tms Simulator 27

# 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.

# Beschreibung der SmartLogic

Die SmartLogic überprüft, ob an sie gestellte Movement Authorities gültig sind, bevor die Fahrstraße gestellt wird und das RBC eine Fahrerlaubnis für einen Zug erhält.

# Struktur und Funktionalität

Das Packet von TMS und SmartLogic greifen auf die gleichenjKlassen zurück.

Es ändert sich nur der Main-Entry-Point des Moduls, das dann zweimal gestartet werden kann.

Der Main-Entry-Point des TMS lautet: de.ibw.tms.MainTmsSim

Der Main-Entry-Point der SmartLogic lautet: de.ibw.smart.logic.intf.SmartLogic

Zuerst werden die Klassen, die von SmartLogic benutzt werden beschrieben.

## Die SmartLogic

Die Klasse mit den Main-Entry Point heißt SmartLogic

Package [de.ibw.smart.logic.intf](file:///C:\\Workspace\\Javadoc\\SL\\de\\ibw\\smart\\logic\\intf\\package-summary.html)

### Class SmartLogic

Die SmartLogic-Klasse hat eine Ausgabe-Warteschlange

#### Felder der Smart-Logic-Klasse

|  |
| --- |
| static java.util.concurrent.SynchronousQueue<[SmartServerMessage](file:///C:\\Workspace\\Javadoc\\SL\\de\\ibw\\smart\\logic\\intf\\messages\\SmartServerMessage.html" \o "class in de.ibw.smart.logic.intf.messages)> outputQueue |

Das ist die Ausgangswarteschlange der SL an das TMS. Sie versendet SmartServerMessages.

Weiteres statische Felder dieser SmartLogic-Klasse:

|  |  |  |
| --- | --- | --- |
| Datentyp | Feld | Erklärung |
| static java.lang.String | [**SMART\_LOGIC**](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\SmartLogic.html#SMART_LOGIC) | Name der Smart-Logic Komponente für Log-files |

Daneben gibt es ein Timeout-Feld in diese Klasse

|  |
| --- |
| static int [TIMEOUT\_SETTING\_WAITING\_MA\_ACK\_IN\_SECONDS](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\SmartLogic.html#TIMEOUT_SETTING_WAITING_MA_ACK_IN_SECONDS) |

Er gibt an wie lange auf ein Acknowledge vom RBC gewartet wird.

#### Methoden der Smart-Logic-Klasse

|  |  |  |
| --- | --- | --- |
| Datentyp | Method | Description |
| static [RbcModul](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\RbcModul.html) | [getRbcClient](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\SmartLogic.html#getRbcClient())() | RBC Modul als Client an den Kommunikationsserver im RBC |
| static [SmartServer](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\SmartServer.html) | [getSmartServForTms](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\SmartLogic.html#getSmartServForTms())() | Server für Anfragen des TMS in der SmartLogic |
| static void | [main](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\SmartLogic.html#main(java.lang.String%5B%5D))​(java.lang.String[] args) | Main-Entry-Point der SmartLogic |
| static [Balise](file:///C:\Workspace\Javadoc\SL\de\ibw\feed\Balise.html) | [pickRandomBalise](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\SmartLogic.html#pickRandomBalise(int))​(int index) | Eine Funktion zum Testen der Position-Reports.  Der index entspricht der Zug-ID, wird aber nicht unbedingt benötigt. Es wird eine Konsolenausgabe auf der Standardkonsole unter „index“ ausgegeben. |
| static void | [sendDummyPositionReport](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\SmartLogic.html#sendDummyPositionReport())() | Test Utility Sendet als RBC einen PositionReport |
| static void | [startSmartLogic](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\SmartLogic.html#startSmartLogic())() | Starten der SL |

#### generateRandomDummyPositionReport

public static void generateRandomDummyPositionReport​(int iCount)

Generiert und schickt „iCount“ viele Dummy-Position-Reports an die SmartLogic

Parameters:

iCount - Anzahl der Position Reports

#### createSmartLogic

public static void createSmartLogic​(boolean bWithUi,

boolean bFakeRbcReceiver,

boolean bSendRbcRequest,

boolean bStartTms)

Intialisiert die Komponenten der SmartLogic

Parameters:

bWithUi - - boolean gibt an ob das TMS mit Oberfäche gestartet wird

bFakeRbcReceiver - - boolean gibt an ob auf der RBC Seite ein Empfänger simuliert werden soll

bSendRbcRequest - - boolean gibt an ob RBC eine Dummy Anfrage an das TMS stellen soll

bStartTms - - boolean gibt an ob das TMS mitgestartet werden soll

#### generatePositionInfo

public static ebd.rbc\_tms.util.PositionInfo generatePositionInfo

Test Utility generiert einen Position-Report

#### startTmsSide

public static void startTmsSide​(boolean bWithUi,

boolean bFakeRbcReceiver,

[RbcReceiver](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\RbcReceiver.html) remoteRbcReceiver)

Startet das TMS

Parameters:

bWithUi - - boolean ob die Oberfläche gestartet werden soll

bFakeRbcReceiver - - boolean ob die RBC Seite als Empfänger simuliert werden soll

remoteRbcReceiver - - Der RBC Receiver der als Simulation für echte Empfänger dient

Package [de.ibw.smart.logic](file:///C:\\Workspace\\Javadoc\\SL\\de\\ibw\\smart\\logic\\package-summary.html)

### Class EventBusManager

Diese Komponente verwaltet Logging GUI Server und das Logging der SL und des TMS

|  |  |  |
| --- | --- | --- |
| Datentyp | Method | Description |
| void | [log](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\EventBusManager.html#log(java.lang.String,java.lang.String))​(java.lang.String sMsg, java.lang.String sModuleId) | Loggt eine Nachricht untern Nennung des angegebenen Untermoduls  sMsg - String - Nachricht, die geloggt werden soll.  sModuleId - - String - Name des Untermoduls, bzw. Komponente. |
| static void | [main](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\EventBusManager.html#main(java.lang.String%5B%5D))​(java.lang.String[] args) | Test Main Methode  args – String[] - unbenutzt |
| static [EventBusManager](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\EventBusManager.html) | [registerOrGetBus](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\EventBusManager.html#registerOrGetBus(int,boolean))​(int id, boolean isTMS) | Factory Methode, die einen EventBusManager generiert, falls das Modul noch keinen Manager besitzt.  Parameters:  id - int - ModulId  isTMS - boolean - ist das Modul ein TMS  Returns:  EventBusManager - gibt den Manager für das Modul der id wider  Throws:  java.io.IOException - - falls es Probleme zum Logger gibt, kann eine Exception erfolgen. |

|  |  |  |
| --- | --- | --- |
| Datentyp | Method | Description |
| static void | [startLogGuiServer](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\EventBusManager.html#startLogGuiServer(boolean))​(boolean bIsTMS) | Startet den GUI-Server unter dem der GUI-Client die Nachrichten abrufen kann.  Parameters:  bIsTMS - boolean - ist der Server innerhalb des TMS  Throws:  java.lang.Exception - - wirft eine Exception wenn der GUIServer nicht generiert werden kann |

Package [de.ibw.smart.logic.datatypes](file:///C:\\Workspace\\Javadoc\\SL\\de\\ibw\\smart\\logic\\datatypes\\package-summary.html)

### Enum BlockedArea.BLOCK\_Q\_SCALE

* All Implemented Interfaces:

java.io.Serializable, java.lang.Comparable<[BlockedArea.BLOCK\_Q\_SCALE](file:///C:\\Workspace\\Javadoc\\SL\\de\\ibw\\smart\\logic\\datatypes\\BlockedArea.BLOCK_Q_SCALE.html" \o "enum in de.ibw.smart.logic.datatypes)>, java.lang.constant.Constable

Enclosing class:

[BlockedArea](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\datatypes\BlockedArea.html)

public static enum BlockedArea.BLOCK\_Q\_SCALE

extends java.lang.Enum<[BlockedArea.BLOCK\_Q\_SCALE](file:///C:\\Workspace\\Javadoc\\SL\\de\\ibw\\smart\\logic\\datatypes\\BlockedArea.BLOCK_Q_SCALE.html" \o "enum in de.ibw.smart.logic.datatypes)>

Dieser Datentyp ist die Einheit der zugeordneten Distanzen. 10CM, 1M und 10M kann verwendet werden.

Package [de.ibw.smart.logic.datatypes](file:///C:\\Workspace\\Javadoc\\SL\\de\\ibw\\smart\\logic\\datatypes\\package-summary.html)

### Class BlockedArea

public class BlockedArea

extends java.lang.Object

Diese Klasse stellt blockierte Gleisabschnitte dar, aber auch blockierte Knoten, wie zum Beispiel Weichen.

|  |  |
| --- | --- |
| Constructor | Description |
| [BlockedArea](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\datatypes\BlockedArea.html#%3Cinit%3E(de.ibw.tms.ma.physical.TrackElement))​([TrackElement](file:///C:\\Workspace\\Javadoc\\SL\\de\\ibw\\tms\\ma\\physical\\TrackElement.html" \o "class in de.ibw.tms.ma.physical) Element) | Dieser Konstruktor instanziiert eine Sperrzone auf ein einzelnes Element wie eine Weiche |
| [BlockedArea](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\datatypes\BlockedArea.html#%3Cinit%3E(de.ibw.tms.plan_pro.adapter.topology.TopologyGraph.Edge,de.ibw.smart.logic.datatypes.BlockedArea.BLOCK_Q_SCALE,int,de.ibw.smart.logic.datatypes.BlockedArea.BLOCK_Q_SCALE,int))​([TopologyGraph.Edge](file:///C:\\Workspace\\Javadoc\\SL\\de\\ibw\\tms\\plan_pro\\adapter\\topology\\TopologyGraph.Edge.html" \o "class in de.ibw.tms.plan_pro.adapter.topology) E, [BlockedArea.BLOCK\_Q\_SCALE](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\datatypes\BlockedArea.BLOCK_Q_SCALE.html) Start\_Ma, int d\_A\_to\_Block\_Start, [BlockedArea.BLOCK\_Q\_SCALE](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\datatypes\BlockedArea.BLOCK_Q_SCALE.html) End\_Ma, int d\_A\_to\_Block\_End) | Dieser Konstruktor instanziiert eine Sperrzone auf einer Kante E von einem Startpunkt bis zu einem Endpunkt.  Parameters:  E – Kante die Abschnittsweise blockiert wird.  Start\_Ma - - Q\_Scale Einheit der Distanz von A bis zum Beginn der Sperrzone  d\_A\_to\_Block\_Start - - Entfernung bis zum Beginn der Sperrzone von Knoten A  End\_Ma - - Q\_Scale End Einheit der Distanz von A bis zum Ende der Sperzone  d\_A\_to\_Block\_End - - Distanz von A bis zum Ende der Sperrzone |

|  |  |  |
| --- | --- | --- |
| Datentyp | Methode | Description |
| boolean | compareIfIntersection(BlockedArea OtherArea | Vergleicht diese Sperrzone mit einer anderen Sperrzone ob es eine Überschneidung gibt.  OtherArea – BlockedArea ist die Sperrzone mit der verglichen wird  Returns true, es gibt eine Überschneidung  False,  es gibt keine Überschneidung |

Package [de.ibw.smart.logic.datatypes](file:///C:\\Workspace\\Javadoc\\SL\\de\\ibw\\smart\\logic\\datatypes\\package-summary.html)

### Class QueueUuidMapper

public class QueueUuidMapper

extends java.lang.Object

Diese Klasse verwaltet Warteschlangen für eine Nachricht-UUID (Kommunikationsverlauf). Die UUID wird für die Kommunikation zum RBC verwendet. Man kann dadurch registrieren ob ein RBC schon eine Nachricht empfangen hat, weil dann ein Acknowledge vom RBC gesendet wird.

|  |  |  |
| --- | --- | --- |
| Datentyp | Methode | Description |
| void | [createQueue](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\datatypes\QueueUuidMapper.html#createQueue(java.util.UUID))​(java.util.UUID uuid) | Diese Methode fügt eine neue Warteschlange für eine UUID zur Verwaltung ein. |
| void | [offer](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\datatypes\QueueUuidMapper.html#offer(java.util.UUID,java.lang.Boolean))​(java.util.UUID uuid, java.lang.Boolean b) | Diese Methode schreibt den Eingang eines Acknowledge in die Warteschlange der gegebenen UUID als Boolean.  Parameters:  uuid - - Der Kommunikationsverlauf  b - - Der Erfolg eines Acknowledge  Throws:  java.lang.InterruptedException - - wird geworfen wenn beim offer ein Interrupt erfolgt |

|  |  |  |
| --- | --- | --- |
| Datentyp | Methode | Beschreibung |
| java.lang.Boolean | [poll](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\datatypes\QueueUuidMapper.html#poll(java.util.UUID))​(java.util.UUID uuid) | Diese Methode wartet bis zu einem Timeout auf ein Acknowledge und gibt Erfolg oder ein Fehlen einer Antwort als Boolean zurück.  Parameters:  uuid - - Id der verfolgten Kommunikation  Returns:  Boolean - ist Acknowledge eingegangen  Throws:  java.lang.InterruptedException - - wird geworfen, wenn beim poll der Warteschlange es Interrupt gab |

Package [de.ibw.smart.logic.intf](file:///C:\\Workspace\\Javadoc\\SL\\de\\ibw\\smart\\logic\\intf\\package-summary.html)

### Class PriorityMessage

All Implemented Interfaces:

java.lang.Comparable<[PriorityMessage](file:///C:\\Workspace\\Javadoc\\SL\\de\\ibw\\smart\\logic\\intf\\PriorityMessage.html" \o "class in de.ibw.smart.logic.intf)>

public class PriorityMessage

extends java.lang.Object

implements java.lang.Comparable<[PriorityMessage](file:///C:\\Workspace\\Javadoc\\SL\\de\\ibw\\smart\\logic\\intf\\PriorityMessage.html" \o "class in de.ibw.smart.logic.intf)>

Eine Nachricht, die Priority einstellbarer Art hat.

|  |  |
| --- | --- |
| [PriorityMessage](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\PriorityMessage.html#%3Cinit%3E(ebd.rbc_tms.Message,java.lang.Long))​(ebd.rbc\_tms.Message M, java.lang.Long iPriority) | Konstruktor instanziiert Priority Nachricht mit Inhalt einer Nachricht an das oder aus dem RBC |

|  |  |  |
| --- | --- | --- |
| Datentyp | Methode | Beschreibung |
| int | [compareTo](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\PriorityMessage.html#compareTo(de.ibw.smart.logic.intf.PriorityMessage))​([PriorityMessage](file:///C:\\Workspace\\Javadoc\\SL\\de\\ibw\\smart\\logic\\intf\\PriorityMessage.html" \o "class in de.ibw.smart.logic.intf) priorityMessage) | Priority Vergleich |
| java.lang.Long | [getiPriority](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\PriorityMessage.html#getiPriority())() | Gibt die Wichtigkeit der Nachricht als Long zurück |
| ebd.rbc\_tms.Message | [getMsg](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\PriorityMessage.html#getMsg())() | Holt die RBC-TMS-Nachricht aus dieser Klasse |
| void | [setiPriority](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\PriorityMessage.html#setiPriority(java.lang.Long))​(java.lang.Long iPriority) | Setzt die Priority als Long |

Package [de.ibw.smart.logic.intf](file:///C:\Workspace\Javadoc\SL\de\ibw\smart\logic\intf\package-summary.html)

## Class RbcModul

All Implemented Interfaces:

java.lang.Runnable

Direct Known Subclasses:

SmartClient, SmartServer

public class RbcModul

extends java.lang.Thread

Ein RBC Modul sendet mithilfe des Netty-Frameworks an das RBC. Kann Nachrichten aus dem RBC erhalten. Wird deshalb zweimal instanziiert.

### 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. |

### MaRequestWrapper Klasse

Diese Klasse umhüllt den MARequest. Sie ist notwendig das Model auf Basis von Frederik Düpmeier von der oberflächlichen Implementierung zu entkoppeln.

|  |  |
| --- | --- |
| publicMARequest Request | Der Request der umhüllt wird. |
| privateTrainModel Tm | Das Model des Zuges der von dem Request betroffen ist. |
| publicMaRequestWrapper(MARequest MAR) | Konstruktor des Wrappers. Er füllt den Request mit Default-Werten |
| publicvoid save() | Speichert den Wrapper in ein Repository für MA-Objekte |
| publicvoid delete() | Entfernt Wrapper mit MA aus dem Repository. |
| publicRoute getRoute() | Holt die Route aus dem Request. |
| publicvoid setRoute(Route R) | Ersetzt die Route dieses Request. |

### MovementAuthority Klasse

Eine Auszeichnung, wie die Fahrstraße unter welchen Parameter erlaubt ist.

|  |  |
| --- | --- |
| privateEoA endOfAuthority | Es wird das Ende der Gültigkeit der Fahrstraße ausgewiesen. |
| privateSvL superviesedLocation | Verknüpft die Fahrstraße mit einer Höchstgeschwindigkeit. |
| privateSSP speedProfile; | Stellt das Geschwindigkeitsprofil unter Betrachten der Streckendaten vor. Welche Geschwindigkeit lässt die Strecke zu. |
| privateGradientProfile gradientProfile | Bringt das Höhenprofil der Strecke vor. |
| privateTrainMovement trainMovement; | Die Zugdaten wurden noch nicht implementiert. |

### Occupation

Wurde noch nicht implementiert.

### Overlap

Muss noch weiterentwickelt werden.

### RbcMA Klasse

RbcMA ein Wrapper des MA für das RBC.

|  |  |
| --- | --- |
| publicRbcMA(String sTrainId) | Dieser Konstruktor enthält die ZugId die von der MA betroffen ist. |
| publicString sTrainId | Die TrainId als Zeichenkette. |
| publicString toJson() throws JsonProcessingException | Gibt die Komplette MA als JSON-Zeichnkette aus. Zum Beispiel für die Übertragung an das RBC. |

### Route Klasse

Eine Route durch das Schienenetz.

|  |  |
| --- | --- |
| privateLinearLocation location | Start und Endpunkt der Route. |
| privateList<Waypoint> waypointsList | Eine Liste von Waypoints die entlang der Route durchfahren werden. In dieser Liste stehen alle ControlledTrackElements |
| publicIWaypoint retrieveWaypointOnTrack(TrackElement Element) | Holt zu dem TrackElement den zugehörigen Waypoint der Route. |
| publicvoid setStartSpot(TrackElement Element, int iMeters) | Setzt den Startpunkt-Track-Element dieser Route. |
| publicvoid addWaypoint(ControlledTrackElement TE, TrackElementStatus Status) | Fügt ein Controlled-Track-Element mit Status als Waypoint zur Route hinzu. |
| publicvoid removeWaypoint(ControlledTrackElement TE) | Entfernt das Controlled-Track-Element als Waypoint von der Route.  Wird der Waypoint nicht gefunden wird eine Exception geworfen. |
| publicvoid addWaypoint(Waypoint W) | Fügt ein Waypoint zur Route hinzu.  Ist der Waypoint bereits in der Route, wird eine Exception geworfen. |
| publicvoid setEndSpot(TrackElement Element, int iMeters) | Setzt den Endpunkt-Track-Element dieser Route. |

### SectionOfLine Klasse

Diese Klasse stellt eine Liste von Streckenabschnitte dar. Diese machen eine Sektion aus.

|  |  |
| --- | --- |
| public List<SpotLocation> locations | Eine Liste von Streckenabschnitte |
| publicSectionOfLine() | Der Konstruktor gibt eine leere Liste von Streckenabschnitte vor. |

### SpeedChange Klasse

Zeigt eine Geschwindigkeitsänderung an.

|  |  |
| --- | --- |
| publicSpeedChange(Chainage chainage, TrackElement trackElement, SectionOfLine lineSection) | Chainage zeigt den Ort der Geschwindikeitsänderung im Streckenverlauf an.  TrackElement stellt das Element dar, in der die Änderung stattfindet. |

### SpeedSegment Klasse

Das Speed Segment beschreibt einen Abschnitt einer Strecke unter der eine Geschwindigkeit vorgeschrieben wird.

|  |  |
| --- | --- |
| privateSSP ssp; | Die Profilzugehörigkeit dieses Abschnittes. |
| privateSpeedChange speedChange; | Zu implementieren |
| privateETCS\_SPEED v\_STATIC; | Die erlaubte Geschwindigkeit in diesem Abschnitt. |
| privateNC\_CDDIFF nc\_CDDIFF; | Zu implementieren |
| privateNC\_DIFF nc\_DIFF; | Zu implementieren |

### SpotLocation Klasse

Eine einzelne Koordinate.

|  |  |
| --- | --- |
| privateChainage chainage; | Abstand in Streckenverlauf zur SpotLocation. |
| privateTrackElement trackElement; | Das TrackElement auf diesem Spot. |
| privateSectionOfLine lineSection; | Die Sektion die zu dem Streckenabschnitt gehört. |
| publicSpotLocation(Chainage chainage, TrackElement trackElement, SectionOfLine lineSection) | Der Konstruktor mit den Daten für die Spot Location. |

### SSP Klasse

Das Geschwindigkeitsprofil wird durch das SSP abgebildet.

|  |  |
| --- | --- |
| privateMovementAuthority movementAuthority | Die MA die dieses Streckenprofil angehört. |
| privateList<SpeedSegment> speedSegments; | Eine Liste von Geschwindigkeitsänderungen im Verlauf. |

### SvL Klasse

Die Supervised Location gibt die Maximale Geschwindigkeit für die MA an. Mir ist noch nicht klar warum eine Maximalgeschwindigkeit einer MA nur an einem Punkt benötigt wird.

|  |  |
| --- | --- |
| privateMovementAuthority movementAuthority; | Die Ma zur SvL |
| privateint vmax | Die Höchstgeschwindigkeit für die MA. |
| publicSvL(Chainage chainage, TrackElement trackElement, SectionOfLine lineSection) | Der Konstrukor |

### TrainMovement

Noch zu implementieren.

### Waypoint

Ein Wegpunkt einer Route.

|  |  |
| --- | --- |
| privateControlledTrackElement TrackElement; | Ein TrackElement, das in irgendeiner Form sich ändern kann. |
| privateTrackElementStatus ElementStatus; | Der Status des Track-Elements, das den Waypoint ausmacht. |
| publicWaypoint(ControlledTrackElement trackElement, TrackElementStatus elementStatus) | Ein Konstruktor mit den Daten: Track-Elemet und Element-Status. |

## Physische MA-Klassen

Die MA-Klassen enthalten auch Klassen, die physische Stellwerke beinhalten. Diese werden nachfolgend behandelt.

### BranchingElement Klasse

Ist im Wesentlichen eine Art von ControlledTrackElement und hat Verzweigungen. Es ist aber erst abgeleitet nicht abstrakt.

### ControlledTrackElement Klasse

Ein Veränderbares Trackelement ist hier erst abstrakt und durch Ableiten nutzbar.

|  |  |
| --- | --- |
| TrackElementStatus Status; | Status des Steuer-Track-Elements. |
| privateint operationTime = 2; | Die Schaltzeit die eine Statusänderung benötigt. |
| publicControlledTrackElement(TrackElementStatus status, int operationTime) | Der Konstruktor, der die obigen Daten übergeben bekommt. |

### DoubleSlip Klasse

Eine Eigenschaft eine Doppelweiche, diese hat zwei Stellschlüpfe.

|  |  |
| --- | --- |
| privateSingleSlip FirstSlipA; | Schlupf A |
| privateSingleSlip SecondSlipB; | Schlupf B |
| publicDoubleSlip(Chainage C) | Konstruktor mit Abstand in Streckenkilometrierung. |
| @Override publicvoid updatePositionedRelation(List<PositionedRelation> relationList) | Verknüpft die Gleise mit den Schlüpfen. |

### EdgeOfMap Klasse

In einem Gleisplan gibt es ein Ende der Gleisplankarte. Diese wird durch die EdgeOfMap dargestellt.

|  |  |
| --- | --- |
| String sViewName; | Eine Bezeichnung für das Kartenende, falls es irgendwo als Bezeichnung oder Ende eines Gleises dargestellt werden soll. |

### FlankAreaElement Klasse

Diese Klasse ist abstrakt und kann erst abgeleitet verwendet werden. Es stellt Elemente mit Flanken dar. Dazu gehören zum Beispiel Trails.

### LocatedNetEntity Klasse

Diese Klasse stellt Elemente dar die eine Positionierung haben können. Dazu gehören die Track-Elemente.

|  |  |
| --- | --- |
| privateChainage chainageBeginn; | Die Streckenkilometrierung nach der das Element beginnt. |
| privateChainage chainageEnd; | Die Streckenkilometrierung nach der das Element endet. |
| GeoCoordinates geoCoordinates; | Die Position des Elmentes als Geocoordinate. |

### Point Klasse

Diese Klasse stellt eine Abzweigung dar und ist ein Branching-Element.

|  |  |
| --- | --- |
| privateint turnaroundLength; | Mir ist noch nicht bewusst was das bedeutet. |
| privateint turnaroundGradient; | Mir ist noch nicht bewusst was das bedeutet. |
| publicPoint(PointType type, int turnaroundLength, int turnaroundGradient, PositionedRelation turnoutNeighbour) | Ein Konstruktor mit detailierten Übergabeparameter. |
| publicPoint(PositionedRelation TurnRelation) | Übergibt nur die Verknpüfung, es wird ein Remote-Operated-Point dann angenommen. |

### Point\_RemoteOperated Klasse

Diese Klasse ist ein Point, mit Remote Opration.

|  |  |
| --- | --- |
| privateint turnaroundLength; | Mir ist noch nicht bewusst was das bedeutet. |
| privateint turnaroundGradient; | Mir ist noch nicht bewusst was das bedeutet. |
| publicPoint(PointType type, int turnaroundLength, int turnaroundGradient, PositionedRelation turnoutNeighbour) | Ein Konstruktor mit detailierten Übergabeparameter. |

## Topologiegenerierung

Es wurde eine neue Klasse geschrieben, die das Generieren der Topologie übernimmt. Diese Topologie wird als Graph dargestellt. Er wird in der Klasse PlanData abgelegt und liegt dann Oberflächen zum Zeichnen vor.

### 

# 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.