Autonomer LKW

Spezifikation

► Builder, Mediator, Observer, Command, State, Composite, CoR, Proxy, Visitor, EventBus, JUnit

Builder 01: Ein **autonomer LKW** hat als Basis ein **Chassis**. Auf dem Chassis ist eine **Kabine** aufgesetzt. Dem Chassis sind **Motor** und drei (eine vordere und zwei hintere) **Achse**n zugeordnet. Die vordere Achse ist eine spezielle bewegliche Achse als Lenkung. Je Achse sind an der linken und rechte Seite je ein **Rad** und eine Bremse montiert. Vorne und hinten auf der linken und rechten Seite des Chassis ist je ein **Blinker** montiert. Hinten auf der linken und rechten Seite des Chassis ist je ein Bremslicht montiert. Das Chassis verfügt hinten über eine **Kupplung** zwecks Verbindung mit dem Sattelanhänger. Auf der linken und rechten vorderen Seite der Kabine ist je ein **Scheinwerfer** und ein **Außenspiegel** angebracht. In jedem Außenspiegel ist eine **Kamera** und ein **LiDAR** installiert. **Builder** 02: Der **Sattelanhänger** hat als Basis ein Chassis. Auf der hinteren linken und rechten Seite des Chassis ist je ein Blinker und Bremslicht sowie zwei hintere Achsen zugeordnet. Das Chassis verfügt über einen **Zapfen** zwecks Verbindung mit dem LKW. Auf dem Chassis befindet sich der Ladebereich. Der **Ladebereich** verfügt auf der linken/rechten Seite über je 8 **Stellplätze** für **Palette**n. Chassis, Zapfen und Ladebereich bilden eine physische Inklusion. **Mediator**: Bremslicht, Blinker, Scheinwerfer, Kamera und LiDAR sind synchronisiert. **Observer**: Über einen **Sensor** wird die **Zentraleinheit** automatisch informiert, wenn der Sattelanhänger angekuppelt wurde. Jeder Stellplatz ist mit einem Sensor ausgestattet. Die Zentraleinheit lädt den **Ladeplan** im **JSON-Format** und überwacht kontinuierlich mit den Sensoren den Ladevorgang. **Command**: Die Zentraleinheit steuert die assoziierten Bauteile über die Kommandos **BlinkerOn**, **BlinkerOff, BrakeLightOn, BrakeLightOff CameraOn**, **CameraOff**, **LiDAROn**, **LiDAROff**, **Brake(percentage)**, **EngineStart**, **EngineShutdown, MoveStraight(percentage), TurnLeft (degree,percentage)**, **TurnRight (degree,percentage)**. Über einen elektronischen Schlüssel wird das Fahrzeug aktiviert oder deaktiviert. Auf dem elektronischen **Schlüssel** ist das mit SHA256 verschlüsselte Passwort Kodiak2024 gespeichert. Die Zentraleinheit ist mit einem Empfangsmodul für das Signal (**SendSignal(password)**) des Schlüssels verbunden. **State**: Bei dem Fahrzeug werden die Status Inactive und Active unterschieden. Bei Empfang des Signals mit korrektem Passwort im Status Inactive erfolgt Wechsel nach Active, bei Active nach Inactive. **Composite**: Eine **Batterie** besteht aus 500 Hauptzellen. Eine Hauptzelle besteht aus 100 Subzellen. Eine Subzelle hat 5 Zellen. Das Attribut energy einer Zelle kann die Werte 0 (entladen) oder 1 (geladen) annehmen. 1% Geschwindigkeit benötigt 2 Einheiten energy. **Visitor**: Für Motor, Kamera und LiDAR existieren **dedizierte Prüfverfahren**. Mit einer Wahrscheinlichkeit von 5% liegt ein Defekt vor. Bei Defekten werden die Kategorien E01, E02 und E03 (zufällig bestimmt) unterschieden. **CoR**: Im Servicecenter existieren zwei Teams [i] Motor und [ii] Sensorik. Ein Team ist hierarchisch mit [01] **Supervisor**, [02a] **OperationTeamManager**, [02b] **EmergencyTeamManager** aufgebaut. Jedem TeamManager sind drei **TechnicalEngineer** zugeordnet. Defekte der Kategorie E01 und E02 werden vom einem TechnicalEngineer im OperationTeam gelöst. Defekte der Kategorie E03 werden von einem TechnicalEngineer im EmergencyTeam gelöst. Für die Reparatur wird je Bauteil ein dediziertes Verfahren angewandt. **Proxy**: Zur Anwendung des Verfahrens meldet sich der verantwortliche TechnicalEngineer mit einem Einmalpasswort von dem Supervisor an einem Proxy an und startet den Roboter für die Reparatur. **EventBus**: Der LKW verfügt über ein Verbindungsstück mit drei Polen. Über den ersten Pol werden die Events für die Bremslichter kommuniziert. Über den zweiten Pol werden die Events für die Blinker kommuniziert. Über den dritten Pol werden die Events für die Bremsen kommuniziert.

Testfälle

1. **Builder 01**: Autonomer LKW besteht aus einem Chassis (einschl. Motor, Achsen, Blinker, Bremslichter und Kupplung), einer Kabine, zwei Außenspiegel, zwei Kamera und zwei LiDAR. Die Blinker, Bremslichter und Außenspiegel sind an der richtigen Seite platziert. In je einem Außenspiegel ist eine Kamera und ein LiDAR installiert.
2. **Builder 02**: Sattelanhänger besteht aus einem Chassis (Blinker, Bremslicht, Achsen, Zapfen) und Ladebereich (Stellplätze).
3. Bei **Aktivieren des Fahrzeugs** werden die beiden Kamera angeschaltet (CameraOn), die beiden LiDAR angeschaltet (LiDAROn), die Engine gestartet (EngineStart) und die Lenkung geradegestellt (MoveStraight(0)).
4. Bei **Deaktivieren des Fahrzeugs** werden die beiden Kamera ausgeschaltet (CameraOff), die beiden LiDAR ausgeschaltet (LiDAROff), die Engine gestoppt (EngineShutdown) und die Lenkung geradegestellt (MoveStraight(0)).
5. **Observer**: Die Zentraleinheit wird automatisch informiert, wenn [i] ein Sattelanhänger angekuppelt und [ii] der Ladevorgang gemäß Ladeplan abgeschlossen ist.
6. Die **Kommandos** BlinkerOn, BlinkerOff, BrakeLightOn, BrakeLightOff CameraOn, CameraOff, LiDAROn, LiDAROff, Brake(percentage), EngineStart, EngineShutdown, MoveStraight (percentage), TurnLeft(degree,percentage), TurnRight (degree,percentage). von der Zentraleinheit werden in korrektes Verhalten umgesetzt.
7. Das **Passwort** wird korrekt auf dem elektronischen **Schlüssel** gespeichert.

   Eine **Aktivierung** und **Deaktivierung** ist nur mit gültigem Passwort möglich.

   Bei Empfang des Signals vom elektronischen Schlüssel erfolgt korrekter Wechsel des Status.
8. Beim **Fahrt auf gerader Strecke** sind der vordere und hintere linke Blinker beim dem LKW sowie der hintere linke Blinker bei dem Trailer ausgeschaltet. Aus Sicherheitsaspekten wird mit 75% Geschwindigkeit (MoveStraight(75)) gefahren.
9. Beim **Linksabbiegen** wird der vordere und hintere linke Blinker bei dem LKW sowie der hintere linke Blinker bei dem Trailer angeschaltet. Des Weiteren wird die Geschwindigkeit auf 50% reduziert (Brake(25), TurnLeft(15,50)).
10. Beim **Rechtsabbiegen** wird der vordere und hintere rechte Blinker bei dem LKW sowie der hintere rechte Blinker bei dem Trailer angeschaltet. Des Weiteren wird die Geschwindigkeit auf 50% reduziert (Brake(25), TurnRight(15,50)).
11. **Composite**: Der Energieverbrauch ist in Abhängigkeit von der Geschwindigkeit korrekt.
12. **Visitor**: Motor, Kamera und LiDAR werden mit dedizierten Verfahren geprüft.
13. **CoR**: Defekte der Kategorie E01 und E02 am Motor werden von einem TechnicalEngineer im OperationTeam Motor gelöst. Defekte der Kategorie E01 und E02 an Kamera oder LiDAR werden von einem TechnicalEngineer im OperationTeam Sensorik gelöst. Defekte der Kategorie E03 am Motor werden von einem TechnicalEngineer im EmergencyTeam Motor gelöst. Defekte der Kategorie E03 an Kamera oder LiDAR werden von einem TechnicalEngineer im EmergencyTeam gelöst.
14. **EventBus**: Bei einem verbundenen Sattelanhänger erzeugen die kommunizierten Events bei den assoziierten Bauteilen das korrekte Verhalten.

Wichtige Hinweise für die Bearbeitung

- **Bearbeitung** erfolgt **individuell.**
- **Studium** der **Struktur** und **Funktionsweise** der beteiligten **Design Patterns**.
- **Programmiersprache** | **Java 21.0.2 (LTS)**
  https://www.oracle.com/de/java/technologies/downloads/#jdk21-windows
- **IntelliJ IDEA** Community **2023.3.3** oder höher
  https://www.jetbrains.com/idea/download
- Verwendung geeigneter **englische**r Begriffe für **Namen** und **Bezeichnungen**.
- Die im Dokument „style_guide.pdf“ genannten **Konventionen** sind **verbindlich**.
- **Modellierung** Klassendiagramm in **Visual Paradigm 17**.
  Bitte
  - achten Sie auf ein geordnetes Gesamtbild[1].
  - benennen Sie die **Datei** mit **self_driving_truck.vpp**.
  - benennen Sie das Klassendiagramm in Abhängigkeit von dem Entwurfsmuster.
  - löschen Sie die *.bak-Dateien von Visual Paradigm.
  - generieren Sie nicht das UML-Diagramm aus dem Quellcode!
- **Implementierung** einer technisch einwandfrei lauffähigen Applikation[2].
  Bitte
  - erstellen Sie ein **IntelliJ-Projekt**.
  - erstellen Sie vorzugsweise für jedes Entwurfsmuster ein dediziertes **Paket**.
  - nutzen Sie die **camelCase-Notation**, um die Lesbarkeit zu vereinfachen.
  - stellen Sie sicher, dass Modellierung und Implementierung harmonieren.
- **Clean-Up** und **Formatierung** des **Source Code**.
  (Code ► Reformat Code [Optimize imports, Rearrange entries, Cleanup code])
- Erstellung einer **7-Zip-Datei self_driving_truck_[matrikelnummer].7z**
- **Zeitansatz:** 20 Stunden
- **Bewertung:** **maximal 100 Punkte**[3]
- **Abgabetermin:** **Upload in Moodle bis** spätestens Sonntag, **03.03.2024**

1 Ungeordnetes Gesamtbild und nicht nachvollziehbare Modellierung resultieren in Punktabzügen!
2 Technische Leistungsfähigkeit muss durch aussagekräftige Junit-Tests nachgewiesen werden.
3 **Modellierung**: 20 Punkte, **Implementierung**: 50 Punkte, **Testmanagement**: 30 Punkte.