



Prof. Dr.-Ing. habil. Martin Wollschlaeger

# AUFGABENSTELLUNG - BELEG

## „SOFTWARE- UND PROGRAMMIERTECHNIK IM MASCHINENWESEN“

Sommersemester 2020

Hinweis: Lesen Sie sich zuerst alle Aufgaben durch und fangen Sie erst dann mit der Bearbeitung der Aufgaben an. Bitte wenden Sie sich bei Fragen an das Forum <https://bildungsportal.sachsen.de/opal/auth/RepositoryEntry/6481149968/CourseNode/91764267052038> oder in den Chat <https://matrix.tu-dresden.de/#/room/#mwinfo2:tu-dresden.de>.

## 1 MOTIVATION

Die meisten modernen Maschinen und Anlagen haben eine Komplexität erreicht, bei der es nur noch schwer möglich ist, sie von Hand, mechanisch oder analog elektronisch zu steuern. Selbst einfachste digitale Steuerungen sind heute mit programmierbaren Mikrokontrollern flexibler, schneller und billiger zu realisieren als mit diskreten kombinatorischen Schaltungen. Diese Tendenz verstärkt sich mit der steigenden Komplexität der Steuer- und Regelungsaufgaben sowie der wachsenden Verfügbarkeit von Sensoren und Aktuatoren. Dabei erfordert die Entwicklung von entsprechender Gerätesoftware Kenntnisse über die funktionalen Randbedingungen für den Betrieb von Maschinen und Anlagen. Selbst wenn Sie die benötigte Software nicht selbst erstellen, müssen Sie Aufgaben und Verhalten sowie Funktionen und Schnittstellen der beteiligten Objekte spezifizieren können. Unvollständige, fehlerhafte oder falsch interpretierte Spezifikationen können nicht nur für erhebliche Verzögerungen in der Entwicklung, sondern auch für hohe Folgekosten in der Fertigung sorgen. Deswegen ist es unerlässlich, Konzepte und Darstellungsmittel der Softwareentwicklung zu verstehen und deren Bedeutung für Programmierer und Softwaretechnologen zu kennen. Das ist nur möglich, wenn Sie alle Phasen in der Softwareentwicklung bearbeiten, da oft erst bei der Implementierung Stärken und Schwächen der Spezifikation erkannt werden.

Durch die Bearbeitung der Belegaufgabe sollen entsprechende Fähigkeiten entwickelt und geübt werden. Der Umgang mit UML-Diagrammen soll gefestigt und deren Überführung in Quelltext durchgeführt werden. Darauf aufbauend sollen durch die Implementierung die erworbenen Kenntnisse zur objektorientierten Programmierung verinnerlicht werden.

## 2 ABLAUF

Bitte beachten Sie die Formalien für Anmeldung und Bearbeitungszeitraum, die für Ihren Studiengang gültig sind. Diese wurden in der Vorlesung bekanntgegeben und sind auf den Seiten des Kurses „Informatik II“ im OPAL nachzulesen (<https://bildungsportal.sachsen.de/opal/auth/RepositoryEntry/6481149968/CourseNode/89259572902560>).

### SELBSTSTÄNDIGKEITSERKLÄRUNG

Der Vordruck für die Selbstständigkeitserklärung ist unter den verlinkten Dokumenten im OPAL zu finden. Dieser ist unbedingt zu verwenden. Die Selbstständigkeitserklärung muss korrekt und vollständig ausgefüllt und eigenhändig unterschrieben sein. Bitte beachten Sie dabei, dass alle benutzten Quellen korrekt aufzuführen sind (Bücher, Internetquellen, ...). Die Quellen sind dabei hinreichend zu konkretisieren, „stackoverflow“ oder „Nachhilfekurs“ sind unzureichend, während „<https://stackoverflow.com/questions/16251817/if-else-statements-in-c-sharp>“ legitim ist. Die Selbstständigkeitserklärung ist in Papierform im Original, entweder zu den Vorlesungs-, Übungs- und Praktikumsterminen oder spätestens zum Zeitpunkt der regulären Klausur im Sommersemester abzugeben.

### ABGABEN

Ihre Aufgabe ist in vier Teile untergliedert. Da die Teilaufgaben dabei aufeinander aufbauen, wird empfohlen, die Aufgaben in der angegebenen Reihenfolge zu bearbeiten. Die Bewertung der einzureichenden Teilaufgaben wird wie folgt gewichtet:

- 20% für Teilaufgabe 1 – Analyse
- 30% für Teilaufgabe 2 – Entwurf
- 50% für Teilaufgabe 3 und 4 – Implementierung

### 3 DISKURSBEREICH „SMARTE FABRIK“

In der industriellen Fertigung nimmt sowohl die Stückzahl- und Variantenflexibilität als auch die Vielfalt der Materialien und Prozesstechnologien ständig zu. Ein Ziel der aktuellen Industrie 4.0 Initiative ist es, diese Herausforderungen durch eine schnellere, flexiblere und effizientere Produktion zu lösen. Insbesondere in dem Ansatz der smarten Fabriken soll die Verantwortung der Produktion an dezentrale, sich selbst organisierende, selbstoptimierende und miteinander kommunizierende Produktionseinrichtungen delegiert werden. So bieten beispielsweise stationäre Fertigungsinseln unterschiedliche Verarbeitungsschritte an und können durch autonome mobile Transportroboter mit Teilen versorgt werden. Entsprechend eines produktspezifischen Rezeptes und der Auslastung der verschiedenen Fertigungsinseln suchen sich diese **Transportroboter dabei selbstständig ihren Weg**, um die Verarbeitungsschritte zur Fertigung eines Produktes abarbeiten zu lassen. Es entstehen somit lose gekoppelte virtuelle Fertigungsketten. Jederzeit lassen sich Produktionseinrichtungen dem Produktionsverbund hinzufügen oder entfernen.

Die Belegaufgabe betrachtet folgendes Szenario: Eine Fertigungsabteilung eines mittelständischen Unternehmens der Metallindustrie soll in einer Simulation nachgebildet werden. Die Fertigungsabteilung soll im Stande sein, verschiedene Teile herzustellen und besteht dazu aus unterschiedlichen Produktionseinrichtungen, welche jeweils spezifische Eigenschaften und Arbeitsabläufe besitzen. **Neben verschiedenen Fertigungsinseln** gibt es genau **ein Eingangs- und ein Ausgangslager** sowie **einen Transportroboter**. Die Anordnung der Produktionseinrichtungen ist in Abbildung 1 zu sehen. Der **Transportroboter** kommuniziert funkbasiert mit den verschiedenen Fertigungsinseln und den beiden Lagern und **organisiert den Ablauf** der Fertigung in **Kooperation mit den Produktionseinrichtungen**. Dabei wird die Reihenfolge der **Verarbeitungsschritte** durch ein **auftragsspezifisches (Produkt-)Rezept** vorgegeben. Der **Start eines Produktionszyklus** erfolgt durch die **Initialisierung eines Teils im Eingangslager**. Im Laufe der Fertigung kann ein Teil **verschiedene Fertigungsinseln mehrmals passieren**. Das **fertige Teil** wird **abschließend im Ausgangslager** eingelagert. In der hier zu entwickelnden Simulation soll der Transportroboter simuliert werden. Im Folgenden werden die beteiligten Bestandteile näher beschrieben.

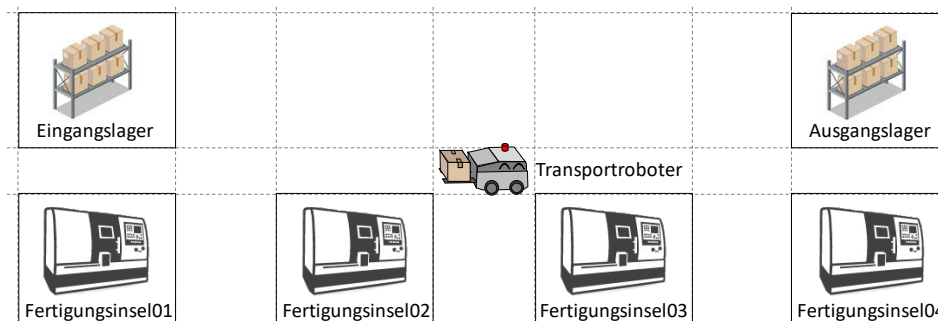


Abbildung 1 - Einrichtungen der smarten Fabrik

## Verarbeitungsschritt

Ein Verarbeitungsschritt beschreibt sowohl die auszuführenden Tätigkeiten, die entsprechend eines **Rezeptes an den Teilen** durchgeführt werden müssen, als auch die Verarbeitungsfähigkeiten, die die einzelnen **Produktionseinrichtungen** haben können (z. B. INITIALISIEREN, LOETEN, FRAESEN, TRENNEN, BESCHICHTEN und EINLAGERN).

## Fertigungsabteilung

Die **Konfiguration** der Produktionseinrichtungen der Fertigungsabteilung wird **als Textdatei vorgegeben**. Die **Fertigungsabteilung** ist eine Klasse, die die **Konfigurationsdatei einliest** und in **Abhängigkeit der Konfiguration** die vorgegebenen Produktionseinrichtungen **initialisiert**. Anschließend wird der **Transportroboter initialisiert** und die **Produktionseinrichtungen** bei dem Transportroboter **registriert**. Die **Fertigungsabteilung** besteht aus **Produktionseinrichtungen und Transportroboter**.

## Produktionseinrichtung

**Fertigungsinseln und Eingangslager sowie Ausgangslager sind spezielle Produktionseinrichtungen**. Eine Produktionseinrichtung stellt gemeinsame Funktionen und Eigenschaften bereit. Sie hat einen **Namen** und eine Liste von **Verarbeitungsfähigkeiten** zusammen mit den jeweiligen **Verarbeitungsdauern** auf dieser Produktionseinrichtung. Die **Verarbeitungsdauern** sind zur Vereinfachung ausschließlich als ganze positive Zahl (Sekunden) angegeben. **Zusätzlich muss eine Produktionseinrichtung ihren gegenwärtigen Status EMPFANGSBEREIT, ABHOLBEREIT oder BELEGT berechnen können**. Jede Produktionseinrichtung kann **Auskunft über ihre Attribute, insbesondere ihre Verarbeitungsfähigkeiten mit der zugehörigen Dauer** geben. Jede Produktionseinrichtung kann ihren Status berechnen. Die Ausprägung ist in den speziellen Produktionseinrichtungen allerdings unterschiedlich.

## Teil

Die allgemeinste Beschreibung für die Dinge, die in der Fertigungsabteilung kursieren, sind Teile, wobei jedes von ihnen eine **Seriennummer** hat. Ein Teil besitzt sowohl eine Liste mit noch **durchzuführenden Verarbeitungsschritten (Rezept)** als auch eine Liste mit allen an ihm **bereits durchgeführten Verarbeitungsschritten (Historie)**. Ein Teil kann über den **nächsten Verarbeitungsschritt Auskunft** geben. Außerdem kann ein Teil einen durchgeführten Verarbeitungsschritt als **erledigt** kennzeichnen, indem es ihn **aus dem Rezept entfernt** und in die **Historie transferiert**. Zu jedem **Verarbeitungsschritt** der in die **Historie** transferiert wird, ist außerdem die Information hinzuzufügen, auf welcher **Produktionseinrichtung** er durchgeführt wurde. Über seine weiteren Attribute kann das Teil Auskunft geben. Letztlich besitzt das Teil noch die **Fähigkeit, einen Selbsttest durchzuführen, ob es vollständig bearbeitet wurde**.

## Fertigungsinsel

Eine Fertigungsinsel ist eine spezielle Produktionseinrichtung, die entsprechend ihrer **Verarbeitungsfähigkeiten** Teile in der, durch die Konfiguration vorgegebenen **Zeit**, verarbeiten kann. Dafür **speichert sie bis zu welchem Zeitpunkt sie mit dem Teil beschäftigt sein wird**. Fertigungsinseln starten mit dem aktuellen Status **EMPFANGSBEREIT**. Nur in diesem **Status** kann die Fertigungsinsel Teile **entgegennehmen**. Sobald eine Fertigungsinsel ein Teil **entgegennimmt**, geht sie in den **Status BELEGT** und **speichert den Zeitpunkt zu dem**

sie den aktuellen Verarbeitungsschritt beenden wird. In diesem Status bleibt sie, bis alle sequentiell durch sie bearbeitbaren Schritte erledigt sind. Zum Entgegennehmen und Zurückgeben hat jede Fertigungsinsel entsprechende Funktionen.

Eine Fertigungsinsel verarbeitet das aktuelle Teil. Ist der Endzeitpunkt der Bearbeitung erreicht und kein weiterer Verarbeitungsschritt des aktuellen Teils an der Fertigungsinsel durchführbar, so setzt sie ihren Status auf ABHOLBEREIT. In diesem Zustand kann das bearbeitete Teil an den Transportroboter übergeben werden, wobei die Fertigungsinsel das bearbeitete Teil zurückgibt und wieder in den Status EMPFANGSBEREIT geht. Die Fertigungsinsel kann Auskunft darüber geben, wie lange sie für einen bestimmten Bearbeitungsschritt brauchen wird.

### Eingangslager und Ausgangslager

Teile werden in automatisch betriebenen Hochregallagern aufbewahrt. Es gibt Eingangs- und Ausgangslager, die aber, abgesehen davon, dass sie beide spezielle Produktionseinrichtung sind, keine Gemeinsamkeiten haben.

#### Eingangslager

Eingangslager können Teile ausgeben und sie zwischenspeichern. Beim Instanzieren des Eingangslagers werden alle neuen Teile instanziiert, gespeichert und der Reihe nach zur Abholung bereitgehalten. Wird mindestens ein Teil zur Abholung bereitgehalten, besitzt das Eingangslager den Status ABHOLBEREIT, ansonsten den Status EMPFANGSBEREIT.

#### Ausgangslager

Ein Ausgangslager kann seinen Status berechnen. Dieser ist immer EMPFANGSBEREIT, denn es kann unbegrenzt viele Teile einlagern. Beim Einlagern und für den Versand vorbereiten von Teilen wird deren Historie ausgegeben und der Abarbeitungserfolg überprüft.

#### Transportroboter

Der Transportroboter kann Produktionseinrichtungen registrieren, so dass er alle Produktionseinrichtungen kennt und mit ihnen kommunizieren kann. Zur Vereinfachung können Sie davon ausgehen, dass der Transportroboter auf vernachlässigbare Art und Weise zwischen den Produktionseinrichtungen hin und her bewegt wird. Kümmern Sie sich nicht darum! Der Transportroboter kann abholbereite Produktionseinrichtungen ermitteln, ein abholbereites Teil von einer Produktionseinrichtung übernehmen, es speichern und zu der ermittelten Produktionseinrichtung transportieren, die für den nächsten Verarbeitungsschritt geeignet ist. Gibt es mehrere Produktionseinrichtungen, die den Verarbeitungsschritt ausführen können, so ist die schnellste als geeignet anzusehen. Sind alle geeigneten Produktionseinrichtungen aktuell belegt, so wird das Teil wieder dem Bestand des Eingangslagers hinzugefügt. Gibt es keine Produktionseinrichtung, die den Verarbeitungsschritt ausführen kann, so wird das Teil im Ausgangslager eingelagert. Der Transportroboter besitzt auch eine Funktion mit der er von außen gestartet werden kann.

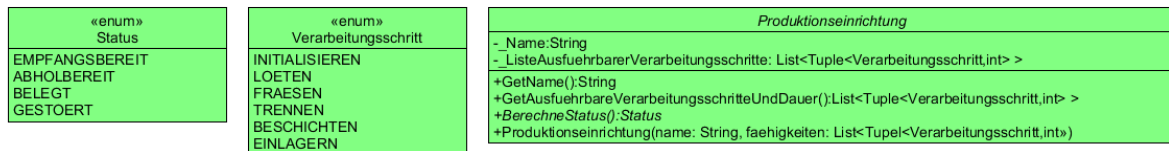


Abbildung 2 – Klasse Produktionseinrichtung und die benötigten Enumerationen

## 4 PRAKTIKUMSAUFGABE

Hinweis: Für die Bearbeitung dieser Aufgabenstellung ist das Verständnis der Basiskonzepte der Objektorientierung zwingend erforderlich. Das Wesen von Klassen und Objekten sollte Ihnen klar sein, bevor Sie hier weiterlesen. Verschaffen Sie sich zunächst einen Überblick über die Bestandteile der im Szenario beschriebenen Fertigungsabteilung sowie deren Funktionsweise und Interaktionen. Verwenden Sie UMLet oder UMLetino (<https://www.umlet.com/>) zum Erstellen der Diagramme.

### TEILAUFGABE 1 – ANALYSE

**TA-1.1** Erstellen Sie entsprechend Abbildung 3 ein **Objektdiagramm**. Ergänzen Sie in Ihrer UML-Darstellung Objekte für die Fertigungsabteilung und das Teil mit der Seriennummer T22. Stellen Sie die impliziten Relationen zwischen allen fünf Objekten dar, sofern diese bestehen. Halten Sie sich dabei zwingend an die UML-Notation und nutzen Sie UMLet/UMLetino.

**TA-1.2** Erstellen Sie ein **Analyseklassendiagramm** der Fertigungsabteilung mit allen beteiligten Klassen. Verwenden Sie dabei die vorgegebenen Klassen (vgl. Abbildung 2). Notieren Sie alle benötigten Rollennamen. Als Ausgangspunkt nutzen Sie bitte zwingend die Ihnen zur Verfügung gestellte „.uxf“-Datei, diese können Sie mit UMLet/UMLetino bearbeiten. Sie können diese über folgenden Link personalisiert herunterladen: [https://wwwpub.zih.tu-dresden.de/~mwinfo2/direkt\\_2020/AKD.php](https://wwwpub.zih.tu-dresden.de/~mwinfo2/direkt_2020/AKD.php)

### Abgabe

Speichern Sie die Diagramme jeweils als .uxf (UMLet/UMLetino) und packen Sie diese gemäß der Anleitung im OPAL in ein ZIP-Archiv und laden Sie dieses im OPAL Kurs bis zum **31.05.2020** hoch.


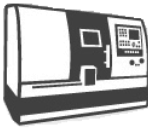
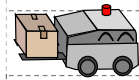
		
Name: LagerEin  AktuellerStatus: ABHOLBEREIT  FähigkeitenMitDauer: (INITIALISIEREN:0)	Name: Fert03  AktuellerStatus: EMPFANGSBEREIT  FähigkeitenMitDauer: (TRENNEN:7); (LOETEN:5)	Name: Robot01  AktuellesTeil: Teil22

Abbildung 3 – Szenario für TA-1.1



## TEILAUFGABE 2 – ENTWURF

Hinweis: Verwenden Sie UMLet oder UMLetino (<https://www.umlet.com/>) zum Erstellen der Diagramme.

- Zur Unterstützung wird Ihnen ein Framework (siehe [http://wwwpub.zih.tu-dresden.de/~mwinfo2/direkt\\_2020/Dokumentation/index.html](http://wwwpub.zih.tu-dresden.de/~mwinfo2/direkt_2020/Dokumentation/index.html)) zur Verfügung gestellt, das wesentliche Klassen (vgl. Abbildung 2) sowie Enumerationen vorgibt. Weiterhin steht Ihnen nach der Abgabe von **TA1** das zum **EKD** erweiterte Diagramm aus TA1.2 zur Verfügung (Ab 01.06.2020). Nutzen Sie dieses sowie das zur Verfügung gestellte Framework **zwingend** im weiteren Verlauf. Schauen Sie sich die Dokumentation dazu genau an und analysieren Sie die gegebenen Klassen und Enumerationen. Passen Sie ggf. Ihre Attribut- und Funktionsnamen entsprechend an.

**TA-2.1** Erstellen Sie ein **Zustandsdiagramm** für den inneren Status der Fertigungsinsel. Überlegen Sie zunächst noch einmal, welche Zustände eine Fertigungsinsel haben kann. Beschreiben Sie in entsprechender UML-Notation alle nötigen Zustände und die Übergänge zwischen ihnen.

- Für die Aufgabe TA-2.2 gilt folgende Szenariobeschreibung: Die Fertigungsinsel ist im Zustand EMPFANGSBEREIT. Es soll das Bauteil Teil01 verarbeitet werden, dass sich abholbereit im Eingangslager **LagerEin** (Zustand ABHOLBEREIT) befindet. Für das Teil und die betrachtete Fertigungsinsel gelten folgende Konfigurationen:

Teil	Rezept
<b>Teil01</b>	LOETEN; BESCHICHTEN; EINLAGERN

Fertigungsinsel	Fähigkeiten mit Dauer
<b>Fert01</b>	LOETEN:9; FRAESEN:2; TRENNEN:2; BESCHICHTEN:6

**TA-2.2** Stellen Sie in einem **Sequenzdiagramm** die Interaktion zwischen Transportroboter, Eingangslager, Teil **Teil01** und der Fertigungsinsel **Fert01** dar. Gehen Sie davon aus, dass der Transportroboter bereits gestartet wurde. Modellieren Sie alle Selbstaufrufe der beteiligten Objekte. Gehen Sie für diese Aufgabe davon aus, dass **Fert01** die einzige Fertigungsinsel der Fertigungsabteilung ist. (Diese Vereinfachung gilt nur für TA-2.2). Sie können die Annahme treffen, dass bei **Fert01** zu jeder zweiten Statusüberprüfung die Bearbeitungszeit für den jeweiligen Rezeptschritt an **Teil01** abgelaufen ist. Sie können das Sequenzdiagramm beenden, sobald der Transportroboter **Teil01**, nach Bearbeitung auf **Fert01**, wieder entgegengenommen hat. Beachten Sie die Details zu den Funktionen des Transportroboters auch die Beschreibungen in TA-4.2. Für entsprechende Einblicke in die Fertigungsinsel konsultieren Sie TA-3.2, für das Eingangslager Lager TA-3.3 und für das Teil TA-4.1.

### Abgabe

Exportieren Sie alle Diagramme als .uxf (UMLet/UMLetino), packen Sie diese mit den UML-Projekten in ein ZIP-Archiv und laden Sie dieses bis zum **21.06.2020** hoch.

### TEILAUFGABE 3 – IMPLEMENTIERUNG

**Hinweis:** Es muss das vorgegebene Framework verwendet werden, das bereits eine Grundstruktur bietet und Sie dahingehend unterstützt, dass Sie vom ersten Moment an ein funktionierendes, ausführbares Projekt haben, welches Ihnen beim genauen Hinsehen auch eine Anfangshilfestellung gibt. Dieses Framework ist in einer für Sie personalisierten Projektmappe enthalten, die Sie von der URL [http://www.pub.zih.tu-dresden.de/~mwinfo2/direkt\\_2020/](http://www.pub.zih.tu-dresden.de/~mwinfo2/direkt_2020/) herunterladen können. Geben Sie dafür Ihren Namen sowie Ihr ZIH-Login an und achten Sie dabei auf die korrekte Schreibweise. Entpacken Sie die Archivdatei „Beleg\_2020\_sxxxxxxx.zip“ **in Ihren persönlichen Ordner**. Anschließend können Sie dort das Projekt „Belegaufgabe2020.sln“ in Visual Studio 2019 öffnen. Bearbeiten Sie die Aufgabenstellung direkt in dieser Projektmappe – nur so kann die Teilaufgabe bewertet werden. Die Projektmappe enthält neben dem eingebundenen Framework auch schon einige Klassen, die Sie nur erweitern müssen, anderen müssen Sie hingegen neu erstellen. Achten Sie auch besonders auf die zum entsprechenden Zeitpunkt bereitgestellten Musterlösungen zu den Teilaufgaben 1 und 2!

Erzeugen Sie für jede Klasse eine eigene .cs-Datei, die den Klassennamen als Dateinamen trägt.

**TA-3.1** Verschaffen Sie sich zunächst einen Überblick über die vorhandenen Elemente.

- Machen Sie sich mit dem vorhandenen Klassen vertraut und versuchen Sie nachzuvollziehen, welche Aufrufe aus dem bereitgestellten Framework zum Einlesen der Textdateien notwendig sind.
- Erstellen Sie im Namespace *Beleg2020* die benötigten Klassen, Funktionen, Attribute und Relationen entsprechend der bekanntgegebenen Musterlösung zu TA1.2. Beachten Sie dabei die beschriebenen Vererbungen sowie damit einhergehende Overrides und die returns.
- Passen Sie die Datei *Fertigungsabteilung.cs* so an, dass die Instanzen der Produktionseinrichtung und des Transportroboters erzeugt werden. Achten Sie darauf, dass in *Fertigungsabteilung.cs* die Funktion zum Einlesen einer Datei bereits vorhanden ist und nur wenige Anpassungen von Ihnen benötigt. Beachten Sie dafür die entsprechenden Kommentare im Quelltext. Andere Funktionen in dieser Klasse sind angelegt, jedoch nicht implementiert. Letztere erkennen Sie an der „*NotImplementedException*“. Denken Sie daran die Zeile mit der Exception zu entfernen nachdem Sie eine entsprechende Implementierung vorgenommen haben. Implementieren Sie diese Funktionen entsprechend ihrer Dokumentation im Quelltext bzw. nach ihrem Funktionsnamen. Erzeugen Sie beim Betreten und Verlassen der von Ihnen implementierten Funktionen sinnvolle Textausgaben auf der Konsole, z.B. "Initialisieren der Produktionseinrichtungen gestartet". Wenn der Initialisierungsvorgang erfolgreich beendet wurde, starten Sie die eigentliche Simulation über die *Start()*-Funktion des Transportroboterobjekts.
- Implementieren Sie in der Klasse *Transportroboter* noch die Funktion *RegistriereProduktionsEinrichtung()* (nur diese – die weiteren Funktionen folgen in TA4.). Sie wird vom Objekt des Typs *Fertigungsabteilung* aufgerufen und nimmt eine Datenstruktur mit allen Produktionseinrichtungen entgegen. *Tipp:* Wenn Sie



die an den Transportroboter übermittelten Produktionseinrichtungen entsprechend ihres Typs (Eingangslager, Ausgangslager, Fertigungsinsel) ablegen, erleichtern Sie sich im Verlauf die Implementierungen.

Starten Sie anschließend die Anwendung und verfolgen Sie die Initialisierung Ihrer Produktionseinrichtungen anhand der Ausgaben. Sofern Sie Teilaufgabe 3.1d umgesetzt haben, sollte Ihr Programm starten und ohne Fehlermeldungen oder Exceptions enden.

### TA-3.2 Implementierung der Fertigungsinsel

Die Klasse Produktionseinrichtung enthält eine abstrakte Funktion, die Sie in der TA-3.1 bereits in der Fertigungsinsel angelegt bzw. überschrieben haben.

- a) Implementieren Sie zunächst die `TeilZurueckgeben()` Funktion. Generieren Sie auch hier eine sinnvolle Ausgabe für die Konsole. Beachten Sie explizit unter welchen Umständen ein Teil von einer Fertigungsinsel abgeholt werden kann! Sehe Sie eine entsprechende Fehlerbehandlung vor.
- b) Implementieren Sie nun die Auskunftsfunktion `GetBearbeitungsdauerFuerSchritt()`, entsprechend des Entwurfs.
- c) Implementieren Sie nun die `TeilEntgegennehmen()`-Funktion. Auch hier sind geeignete Ausgaben auf der Konsole zu erzeugen. Denken Sie an evtl. stattfindende Statuswechsel der Fertigungsinsel, wenn diese ein Teil übernimmt! Wird von einer Fertigungsinsel ein Teil übernommen, kann an dieser Stelle auch berechnet werden, wie lange die Bearbeitung des ersten Rezeptschrittes dauern wird. Die Funktion `DateTime.Now.AddSeconds()` wird hilfreich sein.
- d) Implementieren Sie schließlich `BerechneStatus()`. Da die Simulation vollständig sequentiell abläuft, wird der interne Status einer Fertigungsinsel immer dann ermittelt/berechnet, wenn diese von extern nach ihrem Status gefragt wird. In der veröffentlichten Beispiellösung von TA-2.1 werden alle notwendigen Schritte beschrieben. Die Lösung zu TA-2.2 kann Ihnen auch helfen, denken Sie jedoch daran, dass in TA-2.2 vereinfachende Annahmen getroffen wurden, die hier nicht mehr gelten.

Achten Sie darauf, dass mehrere aufeinander folgende, auf der aktuellen Fertigungsinsel durchführbare Rezeptschritte auch auf dieser Fertigungsinsel durchgeführt werden. Beachten Sie an dieser Stelle auch unbedingt, dass die Fertigungsinsel dafür verantwortlich ist, das Teil anzuweisen, dass es einen durchgeführten Schritt in die Historie überträgt.

### TA-3.3 Implementierung der Lager

- a) Implementieren Sie zunächst das **Ausgangslager**. Abstrakt in Produktionseinrichtung definierte Funktionen müssen auch hier implementiert werden. `BerechneStatus()` liefert auch für das Ausgangslager immer den aktuellen Status, dieser kann sich nach dem Anlegen des Ausgangslagers jedoch laut Aufgabenstellung nicht mehr ändern, machen Sie sich dies zu Nutze.
- b) Implementieren Sie die Funktion `GibHistorieAus()`. Die Funktion gibt für ein ankommendes Teil des Ausgangslagers die Historie auf der Konsole aus. Formatieren Sie die Ausgabe dabei ähnlich wie „Teil mit der Seriennummer {0}“

- wurde mit Schritt {1} bearbeitet von Maschine {2}“. Achten Sie darauf, dass überprüft werden muss, ob ein Teil vollständig abgearbeitet wurde.
- c) Implementieren Sie schließlich die Funktion `TeilFuerDenVersandEmpfangen()`. Sie können davon ausgehen, dass Bestand an abgearbeiteten Teilen im Ausgangslager unbegrenzt sein kann. Überlegen Sie, wie mit den bislang implementierten Funktionen der Klasse die notwendigen Überprüfungen beim Empfang eines fertigen Teils realisiert werden können.
  - d) Implementieren Sie nun das **Eingangslager**. `TeilAusgeben()` liefert dabei solange Teile auf Anfrage an den Transportroboter wie diese in `BestandZubearbeiten` vorhanden sind. `TeilZwischenlagern()` wird benötigt, wenn Teile temporär zwischengelagert werden müssen. Sollte ein Teil in das Eingangslager zurückkehren, so wird es dem `BestandZubearbeiten` wieder hinzugefügt.
  - e) Implementieren Sie `BerechneStatus()` für das Eingangslager.
  - f) Implementieren Sie zum Abschluss die Funktionen zum Initialisieren von Teilen aus der .csv-Datei. Diese Funktion sollte genau einmal zur Erstellung des Eingangslager-Objektes aufgerufen werden. Für das Einlesen orientieren Sie sich am Einlesen der Fertigungsabteilung die Ihnen zur Verfügung gestellt wurde.

### Abgabe

Packen Sie Ihre Visual Studio-Projektmappe gemäß der Anleitung im OPAL in ein ZIP-Archiv und laden Sie dieses im OPAL Kurs bis **30.06.2020** hoch.

## TEILAUFGABE 4 – IMPLEMENTIERUNG

### TA-4.1 Vervollständigung der Klasse Teil

Sie haben ein Gerüst der Klasse Teil erhalten. Implementieren Sie noch fehlende Funktionen bzw. vervollständigen Sie die vorhandenen Funktionen.

- a) Die Funktion `GetNaechsterSchritt()` liefert den nächsten am Teil auszuführenden Verarbeitungsschritt.
- b) Die Funktion `GetSeriennummer()` liefert die Seriennummer des Teils.
- c) Implementieren Sie die Funktion `LiefereHistorie()` entsprechend ihres Namens.
- d) Die Funktion `SelbstTestTeil()` implementieren Sie so, dass sie `true` liefert, wenn das Teil vollständig bearbeitet wurde, also LOETEN, FRAESEN, TRENNEN, BESCHICHTEN nicht mehr im Rezept auftreten.
- e) Die wesentliche Funktion der Klasse Teil ist `TransferiereSchrittInHistorie()`. Sie wird von Objekten der Klasse Fertigungsinsel immer dann aufgerufen, wenn ein Bearbeitungsschritt am Teil ausgeführt wurde. Die Funktion erzeugt ein Datenelement welches jeweils den soeben bearbeiteten Schritt und den Namen der Fertigungsinsel, auf dem dieser ausgeführt wurde, enthält. Sie können dabei die vorgeschlagene Struktur einer Liste von Tupeln (`List<Tuple<Verarbeitungsschritt, string>>`) verwenden. Konsultieren Sie dafür die Dokumentation zu Tuple (<https://docs.microsoft.com/en-us/dotnet/csharp/tuples>). Sollte Ihnen das nicht gelingen, konstruieren Sie einen entsprechenden String und speichern diesen in der Historie (halbiert Ihre Bewertung für diese Funktion).
- f) Implementieren Sie die Funktion `SelbstTestTeil()`. Diese Funktion gibt den Wahrheitswert `true` zurück, wenn das Rezept keine regulären Bearbeitungsschritte mehr enthält (LOETEN, FRAESEN, TRENNEN, BESCHICHTEN), sonst wird `false` zurückgegeben.
- g) Implementieren Sie zuletzt `LiefereHistorie()` so dass die Historie zurückgegeben wird.

### TA-4.2 Implementierung der Klasse TransportRoboter

- a) Die Funktion `GetAbholbereiteProduktionseinrichtung()` liefert eine Produktionseinrichtung die aktuell ABHOLBEREIT ist. Sie können selbst entscheiden, ob allgemein Produktionseinrichtungen zurückgegeben werden oder speziell Fertigungsinsel. In jedem Fall sollten Sie darüber nachdenken, welcher Type von Produktionseinrichtungen bei der Abholung Priorität haben sollte.
- b) Die Funktion `GetSchnellsteFertigungsinsel()` liefert die schnellste Fertigungsinsel, die den folgenden Schritt des aktuellen Teils des Transportroboters erledigen kann. Hier sollten Sie darauf achten, dass diese schnellste Fertigungseinrichtung auch EMPFANGSBEREIT ist.
- c) Die `Start()` Funktion des Transportroboters wird nach erfolgreicher Initialisierung von der Fertigungsabteilung aufgerufen. Sie läuft bis alle Teile im Ausgangslager angekommen sind, also alle Fertigungsinseln und das Eingangslager leer sind. Die wesentliche Funktionalität der Funktion `Start()` ist das immer wiederkehren Übernehmen von Teilen aus Produktionseinrichtungen und Übergeben an die entsprechend nächste Produktionseinrichtung.
- d) Denken Sie daran, die Funktion `Start()` in der Fertigungsabteilung aufzurufen. Anderenfalls wird Ihr Transportroboter nie beginnen.

**TA-4.3** Dokumentieren Sie im Quelltext alle Attribute, Funktionen und deren Parameter. Erzeugen Sie mit dem Tool Doxygen eine Dokumentation zu Ihrem Projekt.

### Abgabe

Packen Sie Ihre Visual Studio-Projektmappe und die Dokumentation gemäß der Anleitung im OPAL in ein ZIP-Archiv und laden Sie dieses im OPAL Kurs bis **19.07.2020** hoch.

**ACHTUNG:** Packen Sie das komplette Projekt ein. Prüfen Sie das Zip-Archiv vor der endgültigen Abgabe noch einmal genau (Größe, Inhalt, etc.). Leere Zip-Archive oder einzelne Solution-Dateien (\*.sln) ohne Projektordner lassen sich nicht positiv bewerten.

### QR-Codes

Für einen direkten Zugriff auf die benötigten URLs können Sie diese QR-Codes nutzen.



OPAL Kurs "Informatik II"



Framework-Dokumentation



personalisierte Projektmappe