

Aufgabe 1 – Implementierung 1 (10 Punkte)

Als Grundlage für diese Implementierungsaufgabe stehen Ihnen die Musterlösung des Klassendiagramms für diese Teilaufgabe und ein ausgewähltes Sequenzdiagramme zur Verfügung. Versuchen Sie diese Diagramme zu verstehen, denn sie dienen als Grundlage für Ihre erste Programmieraufgabe.

In der Implementierung möchten wir, dass Sie eine Klasse *Fabrik* implementieren, die *Bestellungen* entgegennimmt und diese auf der Konsole ausgibt. Die Klasse *Fabrik* enthält als globale Variable eine *ArrayList* in der alle *Bestellungen* gespeichert werden. Mit der Methode *bestellungAufgeben* ist es möglich eine neue *Bestellung* zu platzieren, in der die Anzahl bestellter *Stühle* und *Sofas* angegeben werden. Die Methode *bestellungenAusgeben* durchläuft die Liste der *Bestellungen* und gibt jeweils die Information zu den einzelnen *Bestellungen* auf der Konsole aus. Die Klasse *Fabrik* enthält die Methode *main*, die den Einstieg in das Programm ermöglicht.

Die Klasse *Bestellung* verwaltet eine *ArrayList*, in der alle bestellten *Produkte* gespeichert werden. *Produkte* können entweder *Stühle* oder *Sofas* sein. Als globale Variablen enthält die Klasse *Bestellung* die *Bestellbestätigung*, *Beschaffungszeit*, *Bestellnummer*, wie auch die *Anzahl bestellter Stühle* und die *Anzahl bestellter Sofas*. Zu jeder globalen Variablen muss jeweils eine Methode implementiert werden, um die Information abzufragen. Bei gewissen globalen Variablen muss auch eine Methode vorhanden sein, um die Information in den Variablen zu ändern.

Die Klassen *Stuhl* und *Sofa* sind Erweiterungen von der Klasse *Produkt* und erben somit die Funktionalität und die globalen Variablen der Klasse *Produkt*. Jedes Produkt befindet sich in einem gewissen *Zustand*, beispielsweise *bestellt*, *in Produktion* etc.. Dieser könnte auch in einer globalen Variablen abgespeichert werden. Für die Klasse *Stuhl* und *Sofa* ist Information vorhanden, die für alle Instanzen dieser Klassen gleich ist. Beispielsweise wären das die *Anzahl Holzeinheiten*, *Anzahl Schrauben*, etc. Auch die Methoden, um auf diese Information zuzugreifen, gehören der Klasse an und nicht den einzelnen Instanzen.

Natürlich sind noch weitere Methoden und Variablen notwendig, um das Programm zum Laufen zu bringen. Auch eine Testklasse für die *Fabrik* wird benötigt, um *Bestellungen* aufzugeben und die bestellten *Produkte* ausgeben.

Punkte: 10

Abgabedatum: 30.10.2022 (23:59 CET)

Einzureichen:

- Implementierter und funktionsfähiger Code
- Testklasse
- Ein zip File mit Ihrem gesamten BlueJ Projekt

Bewertungskriterien:

- Die Qualität Ihrer Codeimplementierung – Strukturierung, Benennung, Kommentare, Fehlerbehandlung – **2 Punkte**
- Die richtige Funktionsweise des implementierten Codes – **5 Punkte**
- Genügend Abdeckung der implementierten Tests, um die Funktionalität des Codes vollständig überprüfen zu können – **3 Punkte**

Aufgabe 2 – Implementierung 2 (10 Punkte)

Nachdem Sie die Bestellverwaltung programmiert haben, erweitern wir nun das Aeki Programm mit folgender Funktionalität: Zu jeder *Bestellung* muss nun eine *Auftragsbestätigung* erfolgen. Vor der Rücksendung einer *Auftragsbestätigung* prüft die *Fabrik* mit dem Lager, ob ausreichende

Lagerbestände für die Fertigung der Bestellung vorhanden sind. Wir gehen in unserem Programm von einem **Make-To-Order** Vorgang aus. Bei geringen Lagerbeständen veranschlagt die Auftragsbestätigung zwei Tage zusätzlich zur Standardlieferzeit, welche Sie mit einem Tag festlegen können. Die Lieferzeit errechnet sich durch die Produktionszeit der bestellten Stühle/Sofas plus die Beschaffungszeit der notwendigen Materialien plus die Standardlieferzeit. Wenn die Bestände niedrig sind, sollte das Lager eine Bestellung beim *Lieferanten* aufgeben können.

Das erstellte Programm wird somit um zwei weitere Klassen erweitert. Implementieren Sie die Klassen *Lager* und die Klasse *Lieferant*. Auch für diese Aufgabe stellen wir Ihnen das dazugehörige Klassendiagramm zur Verfügung.

1. Die Klasse *Lager* beinhaltet die Information zu den maximal lagerbaren Materialeinheiten. Diese sind in den Klassenvariablen *maxHolzeinheiten*, *maxSchrauben*, *maxFarbeinheiten*, *maxKartoneinheiten* und *maxKissen*

Implementieren Sie in der Klasse *Lager* folgende Methoden:

- *gibBeschaffungszeit*

Die Methode erhält als Parameter eine Kundenbestellung und liefert *0 Tage* zurück, wenn alle Materialien für die Produktion aller bestellter Produkte vorhanden sind, und *2 Tage*, wenn das Material beim Lieferanten nachbestellt werden muss. Dafür muss die Liste mit allen Produkten der Bestellung durchsucht und die Anzahl benötigter Materialien ausgerechnet werden. Für die Berechnung müssen Sie wissen, ob es sich um einen Stuhl oder um ein Sofa handelt. Mit folgender Abfrage könne Sie dies erfahren:

```
if(produkt instanceof Stuhl){  
    ...  
}else if(produkt instanceof Sofa){  
    ...  
}
```

Die Variable *product* ist vom Typ *Produkt*.

- *lagerAuffuellen*

Diese Methode bestellt fehlende Produkte beim Lieferanten nach und füllt nach Erhalt das Lager wieder auf.

- *lagerBestandAusgeben*

Diese Methode druckt die im Lager vorhandenen Materialeinheiten auf die Konsole aus.

2. Die Klasse *Lieferant* besitzt nur eine Methode, welche der Fabrik ermöglicht, eine Bestellung aufzugeben.

Die Klassen *Lager* und *Lieferant* müssen auch noch erzeugt werden. Überlegen Sie sich genau, wo diese Klassen instanziiert werden müssen. Auch sind Anpassungen in den Klassen *Bestellung* und *Fabrik* notwendig.

In der Klasse *Bestellung* muss zu jeder Instanz auch die korrekter *Beschaffungszeit* und die entsprechende *Lieferzeit* gesetzt werden können. Eine Methode, welche die Liste mit allen bestellten Produkten einer Bestellung retourniert, wird auch in der Klasse *Bestellung* benötigt.

In der Klasse *Fabrik* ändert sich die Methode *bestellungAufgeben*, indem die *Beschaffungszeit* und die *Lieferzeiterrechnet* und in der jeweiligen Bestellung gespeichert werden müssen. Irgendwann muss auch das Lager wieder aufgefüllt werden.

Erstellen Sie die Testklasse für Ihre Fabrik so, dass mehrere Bestellungen aufgegeben und die Auftragsbestätigungen überprüft werden können. Testen Sie auch, ob das Lager eine Bestellung an den Lieferanten senden kann.

Punkte: 10

Abgabedatum: 13.11.2022 (23:59 CET)

Einzureichen:

- Implementierter und funktionsfähiger Code
- Testklasse
- Ein zip File mit Ihrem gesamten BlueJ Projekt

Bewertungskriterien:

- Die Qualität Ihrer Codeimplementierung – Strukturierung, Benennung, Kommentare, Fehlerbehandlung – **2 Punkte**
- Die richtige Funktionsweise des implementierten Codes – **5 Punkte**
- Genügend Abdeckung der implementierten Tests, um die Funktionalität des Codes vollständig überprüfen zu können – **3 Punkte**

Aufgabe 3 – Implementierung 3 (20 Punkte)

Setzen Sie die vorherige Aufgabe fort und erweitern Sie die Klasse *Lieferant* so, dass die bestellten Teile erst in zwei Tagen an das Lager geliefert werden. Natürlich werden wir in unserer Software nicht zwei Tage warten; lassen Sie uns die Zeit beschleunigen und gehen von 1 Stunde = 1 Sekunde aus.

Um dies zu ermöglichen, müssen Sie die Klasse *Lieferant* als *Thread* implementieren.

Benutzen Sie die Testklasse der Fabrik, damit Sie überprüfen können, ob Ihre Bestellungen erst nach der definierten Zeit geliefert werden.

Auch für diese Aufgabe stellen wir Ihnen die Musterlösung des Klassendiagramms zur Verfügung.

Wenn die Implementierung der Klasse *Lieferant* als *Thread* funktioniert, fahren Sie folgendermassen fort: Die Fabrik übergibt nun die Bestellungen dem *Produktionsmanager*. Implementieren Sie dazu die Klasse *Produktions_Manager*. Diese Klasse sollte auch als *Thread* implementiert werden, damit sie immer wieder neu eintreffende Bestellungen abarbeiten und den Robotern zum Produzieren geben kann. Im Konstruktor der Klasse *Produktions_Manager* sollten alle *Roboter* – *Holzbearbeitungs_Roboter*, *Montage_Roboter*, *Lackier_Roboter*, *Verpackungs_Roboter* – als *Threads* instanziiert und gestartet werden. Auch sind zwei *LinkedLists* notwendig, um die zu verarbeitende Bestellungen und die Bestellungen in Produktion zu verwalten. In der Klasse *run* ist eine unendliche Schleife notwendig...

```
while(true){  
  
    //Ist eine neue Bestellung eingetroffen, dann  
  
    //hole die nächste Bestellung und starte die Produktion
```

```

...
//dann lass den Thread eine kurze Weile schlafen
try{
Thread.sleep(zeit);
}catch (InterruptedException ie){
    ie.printStackTrace();
}
}
}

```

...welche immer wieder prüft, ob eine neue Bestellung eingetroffen ist. Diese Bestellung wird dann aus der Liste der zu verarbeitenden Bestellungen rausgenommen und in die Liste der zu produzierenden Bestellungen gespeichert. Dann wird die Produktion gestartet.

Abhängig von den Kundenaufträgen und Lagerbeständen plant die Fabrik die notwendigen *Roboter* für die Fertigung ein. Jeder Artikel einer Bestellung muss von allen Robotern verarbeitet werden. Zum Starten der Produktion implementieren Sie die Methode *starteProduktion*. Diese kriegt eine Bestellung als Parameter übergeben und alloziert jedem Produkt der Bestellung die jeweiligen Roboter in der richtigen Reihenfolge. Achten Sie darauf, dass sich die Produktionsreihenfolge der Roboter für Stühle und Sofas unterscheiden. Die Liste mit den allozierten und in der richtigen Reihenfolge abgespeicherten Robotern wird im Produkt selbst abgespeichert. Somit hat jedes *Produkt* die Kontrolle über die zu durchlaufenden Produktionsstationen.

In der Klasse *Produkt* ist somit eine neue Methode *naechsteProduktionsStation* zu implementieren, welche aus der Liste der zu durchlaufenden Roboter, den jeweils nächsten Roboter rausholt und diesem das aktuelle Produkt in der Liste der zu produzierenden Produkte hinzufügt.

Für die Produktion müssen Sie die Klassen *Holzbearbeitungs_Roboter*, *Montage_Roboter*, *Lackier_Roboter* und *Verpackungs_Roboter* implementieren. Diese sind alle Erweiterungen der Klasse *Roboter*, welche wiederum eine Erweiterung der Klasse *Thread* ist.

Die Klasse *Roboter* verwaltet eine Liste (Warteschlange) mit allen zu produzierenden Produkten. In der Methode *run* ist wieder eine unendliche Schleife notwendig, um immer wieder zu schauen, ob neue Produkte, welche produziert werden müssen, eingetroffen sind. Ist ein Produkt in der Warteschlange eingetroffen, so produziert der Roboter dieses Produkt. Die Produktion des Produkts wird in der Methode *produziereProdukt* simuliert, indem der Thread für eine Zeit schlafen gelegt wird. Danach kann das Produkt zum nächsten Roboter weitergeleitet werden.

Im *Produktions_Manager* wird geprüft, ob eine Bestellung abgeschlossen ist. Wenn ja, wird dies in der Klasse *Bestellung* vermerkt und eine entsprechende Nachricht im Terminal ausgegeben.

Beachten Sie, dass sich die Lagerbestände während der Produktion verringern und bei Bedarf nachbestellt werden müssen. Dafür sollte die Klasse *Lager* um diese neue Funktionalität ergänzt werden. Wir gehen davon aus, dass wir nie einen Kundenauftrag erhalten, der Rohstoffe benötigt, die die Lagerkapazität überschreiten.

Damit der Ablauf in der Fabrik nachvollzogen werden kann, verwenden Sie das Terminal, um jeweils Meldungen auszugeben.

Erweitern Sie die Testklasse der Fabrik, wenn nötig, damit Sie überprüfen können, ob Ihre Bestellungen geliefert werden.

Punkte: 20

Abgabedatum: 04.12.2022 (23:59 CET)

Einzureichen:

- Implementierter und funktionsfähiger Code
- Testklasse
- Ein zip File mit Ihrem gesamten BlueJ Projekt

Bewertungskriterien:

- Implementierung Ihres Produktionsmanagement- und Lieferantenbestellprozesses, der zeigt, wie Sie die Bestellungen abarbeiten und die Produkte produzieren – **10 Punkte**
- Die richtige Funktionsweise des implementierten Codes – **3 Punkte**
- Die Qualität Ihrer Codeimplementierung – Strukturierung, Benennung, Kommentare, Fehlerbehandlung – **2 Punkte**
- Genügend Abdeckung der implementierten Tests, um die Funktionalität des Codes vollständig überprüfen zu können – **5 Punkte**

Aufgabe 4 – Benutzeroberfläche (20 Punkte)

Erstellen Sie eine schöne GUI, die dem Fabrikleiter einen Überblick über den gesamten Prozess gibt. Die gleiche GUI kann verwendet werden, um die Aufträge einzugeben und ihren Status anzuzeigen. Seien Sie kreativ!

Punkte: 20

Abgabedatum: 31.12.2022 (23:59 CET)

Einzureichen:

- Ein zip File mit Ihrem gesamten BlueJ Projekt

Bewertungskriterien:

- Die GUI unterstützt die primären Anwendungsfälle – **10 Punkte**
- Die GUI verfügt über eine Implementierung und das verwendete Designmuster zeigt, dass sie leicht für zukünftige Anwendungsfälle erweitert werden kann – **10 Punkte**