Karlsruher Institut für Technologie

Software Design and Quality

Erstklausur

Softwaretechnik II

Wintersemester 2021/22

Prof. Dr. Ralf H. Reussner

04.04.2022

Musterlösung

Zur Klausur sind keine Hilfsmittel und kein eigenes Papier zugelassen. Die Bearbeitungszeit beträgt 90 Minuten. Die Klausur ist vollständig und geheftet abzugeben. Verwenden Sie ausschließlich dokumentenechte Stifte. Mit Bleistift oder roter Farbe geschriebene Angaben werden nicht bewertet. Lösen Sie die Aufgaben in leserlicher Schrift. Unlesbare Lösungen können naturgemäß nicht positiv bewertet werden.

Aufgabe	**1**	**2**	**3**	**4**	**5**	**6**	Σ
Maximal	10	16	20	12	16	16	90
K1							
K2							
K3							

Aufgabe 1: Wissensfragen *(10 Punkte)*

a) Nennen Sie stichpunktartig zwei Hauptcharakteristiken von *Software as a Service* Cloud-Anwendungen aus der Vorlesung. (2 Punkte)

Musterlösung: [redacted]

- **Network-based access** to, and management of, commercially available software
- Activities that are **managed from central locations** rather than at each customer's site, enabling customers to access applications remotely via the Web
- Application delivery that typically is closer to a **one-to-many model** (single instance, multi-tenant architecture) than to a one-to-one model, including architecture, pricing, partnering, and management characteristics
- **Centralized feature updating**, which obviates the need for downloadable patches and upgrades

b) Nennen Sie die vier Bestandteile eines standardisierten Containerabbilds *(Container Image)*. (2 Punkte)

Musterlösung: [redacted]

- **Image Manifest**: indicates schema version and references all configurations
- **File System Layers**: specifies archives forming the container's content
- **Image Index**: declares platform requirements and contained manifests
- **Container Configuration**: describes deployment of container

c) Nennen Sie drei der fünf Dimensionen der Verlässlichkeit *(Dimensions of Dependability)* aus der Vorlesung Software-Qualität. (1,5 Punkte)

Musterlösung: [redacted]

- Availabilty (Verfügbarkeit)
- Reliability (Zuverlässigkeit)
- Safety (Betriebssicherheit)
- Security (Informationssicherheit)
- Resilience (Widerstandsfähigkeit)

d) Handelt es sich bei dem im folgenden beschriebenen Echtzeitsystem um ein Monitoring- (*monitoring*) oder Kontrollsystem (*control system*)? Begründen Sie Ihre Antwort kurz. (1 Punkt)

Der Landeassistent einer Mondlandefähre überwacht, wenn aktiviert, die Fallgeschwindigkeit im Landeanflug und zündet, sobald eine Fallgeschwindigkeit von 1 m/s *überschritten wird, selbstständig kurzzeitig die Bremstriebwerke.*

Musterlösung: [redacted]

e) Nennen Sie die drei Echtzeitentwurfsmuster aus der Vorlesung, die keinen *Fail-Safe*-Zustand benötigen, um die Zuverlässigkeit zu erhöhen. (1,5 Punkte)

Musterlösung: [redacted]

- Homogeneous Redundancy: Protection against random faults
- Triple Modular Redundancy: Protection against random fault with continuation of functionality
- Heterogeneous Redundancy: Protection against random and systematic faults

f) Nennen Sie vier der zehn Prinzipien zur Entwicklung sicherer Software aus der Vorlesung. (2 Punkte)

Musterlösung: [redacted]

1. Secure the weakest link
2. Practice defence in depth
3. Fail securely
4. Secure by default
5. Follow the principle of the least privilege („Need to Know“ Principle)
6. No security through obscurity (à Kerkhoff’s principle)
7. Minimize attack surface
8. Privileged Core
9. Input Validation and Output Encoding
10. Don‘t Mix Data and Code

Aufgabe 2: Anforderungserhebung *(16 Punkte)*

Im Folgenden betrachten wir die Entwicklung eines Softwaresystems für ein Computerfachgeschäft *(computer retail store)*, abgekürzt *CRS*:

> *Dieses Softwaresystem soll zum einen ein Online-Geschäft* (online shop) *für Ihre Kunden bereitstellen und zum anderen von den Angestellten als Kassensystem* (accounting), *Lagerverwaltung* (stock management) *und zur Bestellungsabwicklung* (order processing) *genutzt werden können. Da das Computerfachgeschäft seinen Kunden zusätzlich die Reparatur eigener Geräte anbietet, muss das Softwaressystem auch die Abwicklung von Reparaturen in der Werkstatt* (repair workhop) *unterstützen.*

Während der Anforderungserhebung wurden für das Online-Geschäft unter anderem die folgenden Anforderungen identifiziert:

Nr.	Anforderung
/R1/	*Der Softwareentwicklungsprozess muss den ISO Standard 9001 für die Entwicklung, Wartung und Qualitätssicherung erfüllen.*
/R2/	*Die Anzahl Klicks, die ein Kunde im Online-Geschäft benötigt, um ein Produkt zu kaufen, muss minimal sein.*
/R3/	*Klickt ein Kunde auf* Registrieren, *erhebt das Online-Geschäft zuerst die Kundendaten und legt dann ein nicht freigeschaltetes Kundenkonto im System an. Abschließend wird eine Bestätigungsmail an den Kunden versandt.*
/R4/	*Die von einem Kunden erhobenen Daten sind eine eindeutige Email-Adresse, der Vor- und Nachname, ein Passwort sowie ein Anschrift.*
/R5/	*Zur Erhöhung der Datensicherheit darf das Online-Geschäft keine Passwörter im Klartext speichern.*
/R6/	*Das Online-Geschäft muss alle Vorgaben der Datenschutz-Grundverordnung (DSGVO) erfüllen.*
/R7/	*Sobald der Link einer Bestätigungsmail aufgerufen wird, schaltet das Online-Geschäft das zugehörige Kundenkonto automatisch frei.*
/R8/	*Das Online-Geschäft muss bei 100.000 gleichzeitigen Kunden eine Antwortzeit von unter* 1s *für jede Seitenanfrage haben.*

a) Nennen Sie vier weitere Stakeholder, die neben *Kunden* und *Angestellten* für die Entwicklung des Online-Geschäfts relevant sind. (2 Punkte)

Musterlösung: [redacted]

- Buchhalter
- Paketdienste
- Lagerverwalter
- Gerätehersteller
- Datenschutzbeauftragter

- Prüfer der ISO-Zertifizierung
- ~~Finanzamt~~ (gibt keine Punkte)

b) Klassifizieren Sie die oben stehenden Anforderungen nach der in der Vorlesung behandelten Klassifikation von Glinz. Ordnen Sie hierzu die acht Anforderungen des Online-Geschäfts den in der Vorlesung genannten Anforderungsarten *(kind: functional, quality, constraint)*, deren Unterkategorie *(subkind: functions, data, performance, security, availability, legal, …)* sowie deren Repräsentation *(representation: operational, quantitative, qualitative, declarative)* zu. (12 Punkte)

Musterlösung: [redacted]

Nr.	Anforderung	Art	Unterkategorie	Repräsentation
/R1/	*ISO 9001*	constraint	design & impl.	declarative (qualitative)
/R2/	*Minimale Anzahl Klicks*	quality	usability	qualitative (declarative)
/R3/	*Kunde registrieren*	functional	behavior (function)	operational operational
/R4/	*Kundendaten*	functional	data	operational
/R5/	*Keine Passwörter im Klartext*	quality	security	declarative (operational)
/R6/	*DSGVO*	constraint	legal	declarative
/R7/	*Bestätigungsmail*	functional	reaction (behavior)	operational operational
/R8/	*100.000 Kunden*	quality	performance	quantitative

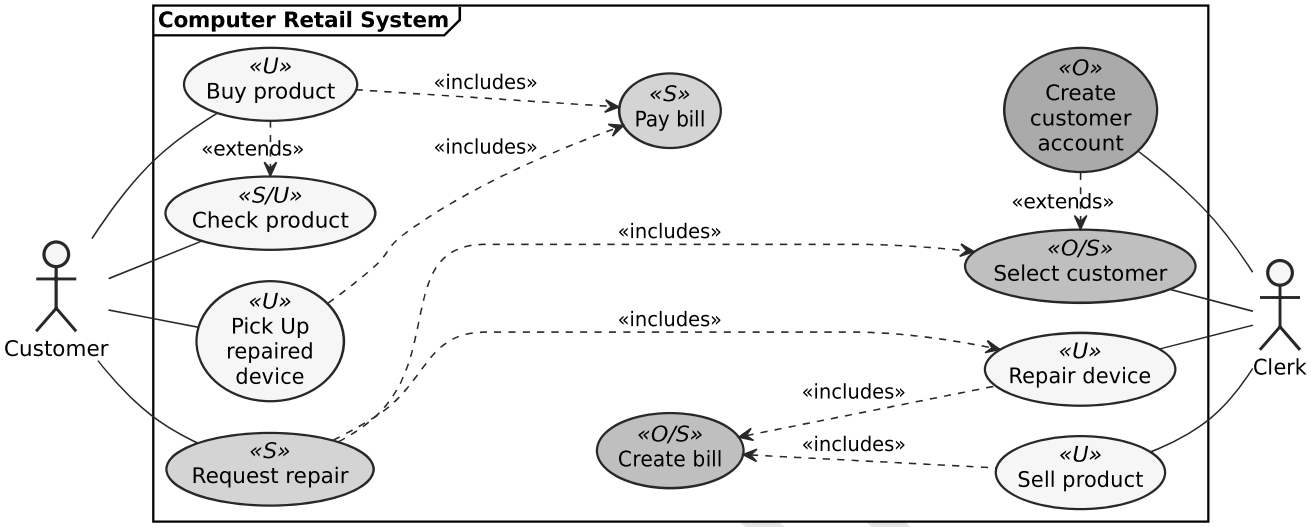
c) Einige der oben stehenden Anforderungen verstoßen gegen die Empfehlungen für gut geschriebene Anforderungen *(basic writing recommendations)* aus der Vorlesung. Finden Sie **zwei** Anforderungen die gegen **unterschiedliche** Richtlinien verstoßen. Nennen Sie jeweils die Nummer der Anforderung und Richtlinie, gegen die verstoßen wurde. (2 Punkte)

Musterlösung: [redacted]

- /R2/: weak language: *„minimal“*
- /R3/: short sentence: *Mehrere Sätze in einer Anforderung.*
- /R4/: passive language: *„Die von einem Kunden erhobenen Daten“*
- /R7/: passive language: *„Sobald der Link einer Bestätigungsmail aufgerufen wird“*

Aufgabe 3: Anwendungsfallbeschreibung *(20 Punkte)*

Diese Aufgabe betrachtet die Anwendungsfälle der Kunden *(customers)*, Angestellten *(clerks)* und Ingenieure *(engineers)* des Computerfachgeschäfts. Diese sind im folgenden Anwendungsfalldiagramm zusammengefasst und in der nachfolgenden Tabelle kurz erläutert.



Anwendungsfall	Beschreibung
Check product	Betrachten von Produkten durch einen Kunden.
Buy product	Kaufen von Produkten durch einen Kunden.
Sell product	Verkaufen von Produkten durch einen Angestellten.
Create bill	Erstellen einer Rechnung für Produkte und Leistungen.
Pay bill	Begleichen des Rechnungsbetrags durch den Kunden.
Request repair	Stellen einer Reperaturanfrage durch einen Kunden.
Select customer	Auswahl eines Kunden in der Kundendatenbank.
Create customer account	Erstellen eines neuen Kundenkontoobjekts in der Kundendatenbank.
Repair device	Reparatur eines vom Kunden abgegebenen Gerätes.
Pickup repaired device	Abholen des reparierten Gerätes durch den Kunden.

a) Klassifizieren Sie jeden Anwendungsfall im obigen Anwendungsfalldiagram (use case diagram) entweder als Nutzerziel *(user goal)* mit U, als Teilfunktion *(subfunction)* mit S oder als Operation *(operation)* mit O. Tragen Sie hierzu die entsprechende Klassifikation in die mit „✎“ markierten Bereiche der Anwendungsfälle ein. (5 Punkte)

Musterlösung: [redacted]

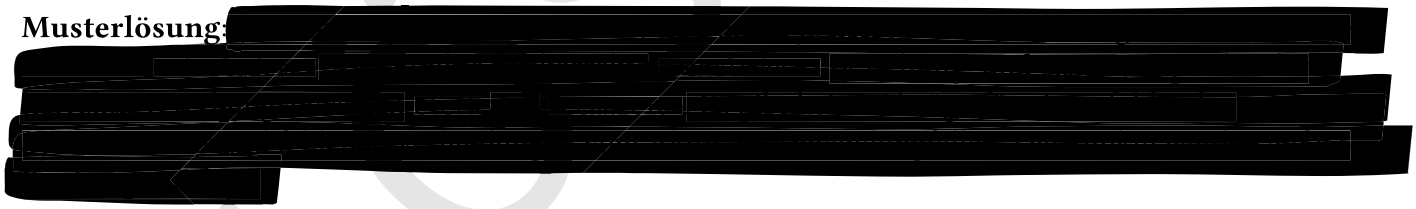
Die Musterlösung ist im obigen Anwendungsfalldiagramm dargestellt.

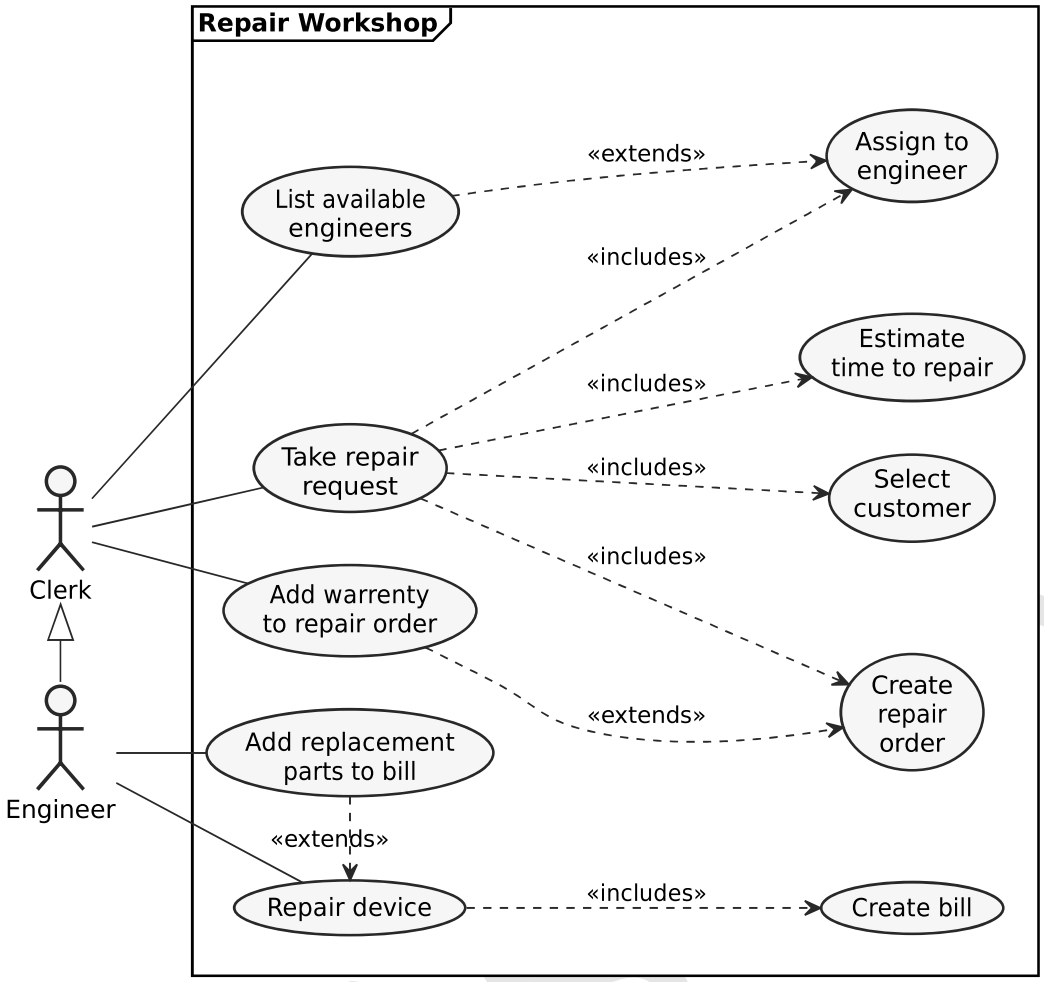
Da diese Anwendungsfälle nicht detailliert genug sind, ist es Ihre Aufgabe ein weiteres Anwendungsfalldiagramm zu erstellen. Hierzu charakterisiert die folgende Beschreibung die Aufgaben die in der angeschlossenen Reparaturwerkstatt erfüllt werden:

> *Für die folgende Beschreibung wird angenommen, dass Angestellte und Ingenieure am CRS angemeldet sind. In der Reparaturwerkstatt* (repair workhop) *können Angestellte* (clerk) *Reparaturanfragen entgegennehmen* (take repair request). *Hierzu wählt der Angestellte zuerst das Kundenkonto im CRS aus* (select customer). *Daraufhin erstellt er für diesen einen Reparaturauftrag* (create repair order). *Während der Erstellung kann der Angestellte über eine Schaltfläche einen Garantiebeleg zum Auftrag hinzufügen* (add warrenty to repair order), *falls dieser vorgelegt wurde. Nach Erstellung des Reparaturauftrags weist der Angestellte den Auftrag einem Ingenieur zu* (assign to engineer). *Falls für die Reparatur ein bestimmter Ingenieur notwendig ist, kann sich der Angestellte in diesem Schritt eine Liste aller verfügbaren Ingenieure anzeigen lassen* (list available engineers). *Nach der Zuweisung des Ingenieurs ermittelt der Angestellte die voraussichtliche Reparaturzeit* (estimate time to repair). *Reparaturen werden ausschließlich von Ingenieuren* (engineer) *durchgeführt* (repair device). *Falls hierfür Ersatzteile verwendet werden müssen, kann der Ingenieur deren Kosten über eine Schaltfläche zusätzlich zur Rechnung hinzufügen* (add replacement parts to bill). *Am Ende der Reparatur erstellt der Ingenieur eine Rechnung* (create bill). *Im CRS werden alle Ingenieure als spezielle Angestellte betrachtet, die zusätzlich Reparaturen (sowie zugehörige Teilaufgaben) durchführen können.*

b) Erstellen Sie nun ein Anwendungsfalldiagramm *(use case diagram)* genau so, dass es die Anforderung im obigen Text beschreibt. Zeichnen Sie nur Anwendungsfälle und Aktoren ein, die im Text referenziert werden. Achten Sie darauf, an den geeigneten Stellen Beziehungen zwischen den Anwendungsfällen *(use cases)* und Aktoren *(actors)* zu modellieren. (8 Punkte)

Musterlösung:




Repair Workshop
Assign to engineer
«extends»
List available engineers
«includes»
Estimate time to repair
«includes»
Take repair request
«includes»
Select customer
«includes»
Clerk
Add warrenty to repair order
Create repair order
«extends»
Add replacement parts to bill
Engineer
«extends»
Repair device
«includes»
Create bill

c) Verwenden Sie das textuelle Anwendungsfallbeschreibungsschema *Fully-Dressed* zur vollständigen Spezifikation des Erfolgsfalls der Aufnahme einer Reparaturanfrage *(take repair request)*. Nutzen Sie hierfür die unten gegebene Schablone. Nennen Sie zwei Stakeholder und geben Sie mindestens eine Erweiterung an. (7 Punkte)

Musterlösung: [redacted]

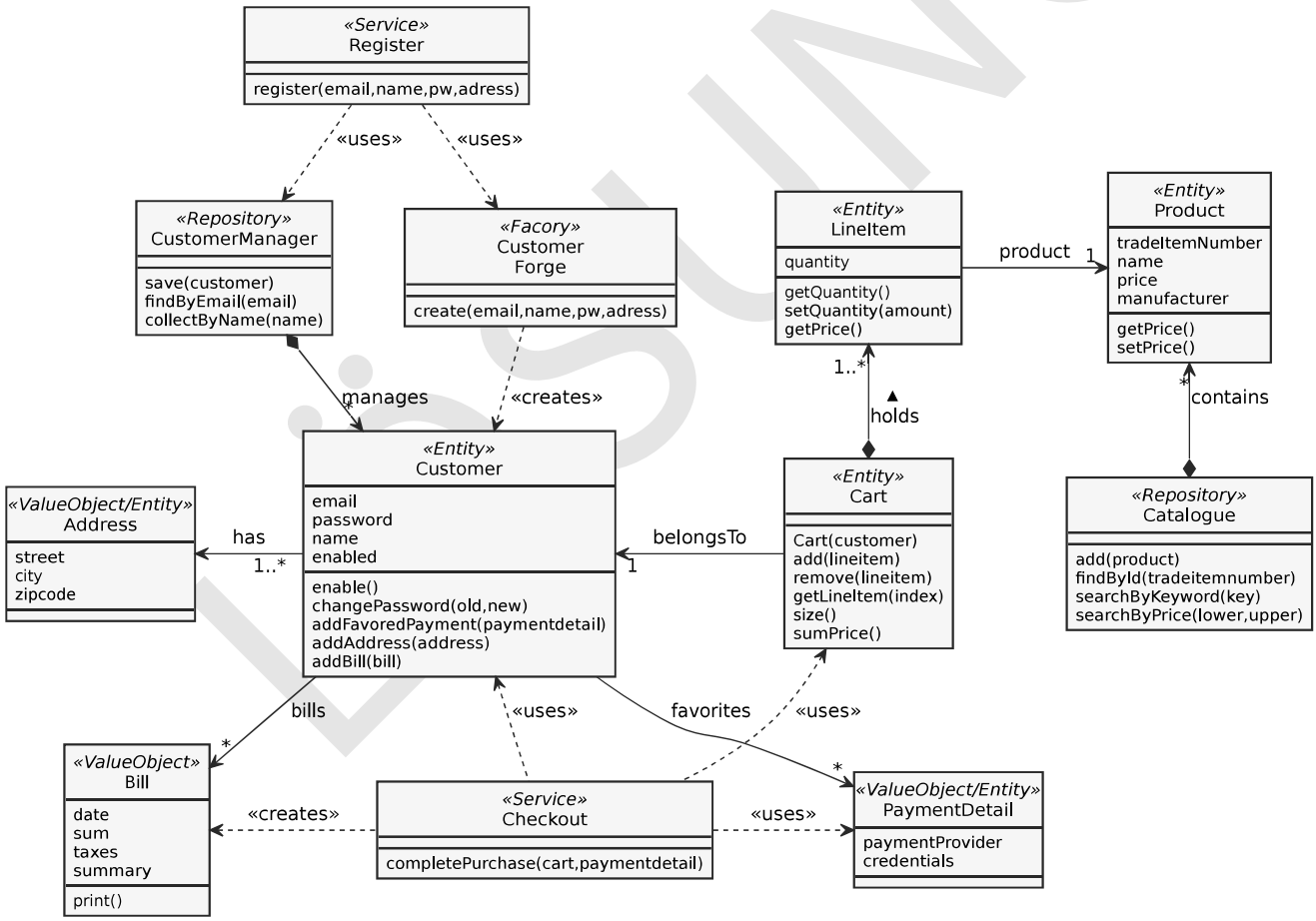
Name d. Anwendungsfalls *(Use Case Name)*	**Reparaturanfrage aufnehmen** *(take repair request)*
Zielebene *(Goal Level)*	Nutzerziel (User Goal)
Primäraktor(en) *(Primary Actor(s))*	Angestellte
Stakeholder(s)	Kunde, Ingenieur, Gerätehersteller
Vorbedingungen *(Preconditions)*	• Der Angestellte ist im CRS angemeldet • Der passende Kunde wurde ausgewählt
Nachbedingungen *(Postconditions)*	Der Reperaturauftrag wurde * erstellt, * einem Techniker zugewiesen, und * eine Reparaturzeit ermittelt
Primäres Erfolgsszenario *(Main Success Scenario)*	1) Reparaturauftrag erstellen 2) Reparaturauftrag einem Techniker zuweisen 3) Abschätzen der Reparaturzeit
Erweiterung (Weiterer Ablauf) *(Extension; Additional Flow)*	• 1a) Falls einen Garantiebeleg vorgelegt wurde, hinzufügen des Garantiebelegs zum Reparaturauftrag. • 2a) Falls bestimmter Techniker für Reparatur notwendig ist, auflisten der verfügbar Techniker

Aufgabe 4: Domänen-getriebener Entwurf *(12 Punkte)*

Der folgende Aufgabenkomplex widmet sich der Anwendung des Domänen-getriebenen Entwurfs für das Online-Geschäfts *(online shop)* des Computerfachgeschäfts. Dieses erlaubt es Kunden nach Produkten im Produktkatalog *(catalogue)* zu suchen, ausgewählte Produkte in einen Einkaufswagen *(cart)* zu legen und die Produkte im Einkaufswagen an der Kasse zu bezahlen *(checkout)*.

a) Durch die Domänenanalyse des Online-Geschäfts ist das folgende Klassendiagramm entstanden. Klassifizieren Sie jedes Domänenkonzept entsprechend der Bausteine (building blocks) des Domänen-getriebenen Entwurfs entweder als `ValueObject` mit V, `Entity` mit E, `Service` mit S, `Factory` mit F oder `Repository` mit R. Tragen Sie hierzu die Klassifikation in die mit „✎“ markierten Bereiche der Domänenklassen ein. (6 Punkte)

Musterlösung: [redacted]



b) Begründen Sie Ihre Entscheidung für die Klassifikation der Domänenklasse `Address` in Stichpunkten. (1 Punkt)

Musterlösung:

Falls als `<<Entity>>` klassifiziert

- `Address`-Objekte haben eine eindeutige Identität
- Änderungen an `zipcode` und `street` Attribut der selben Adresse müssen über Ihre Lebenszeit nachverfolgbar sein

Falls als `<<ValueObject>>` klassifiziert

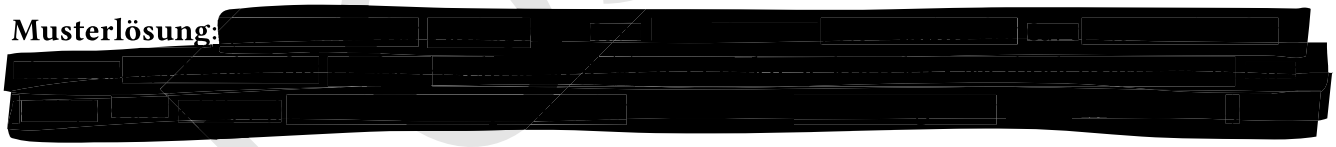
- Keine eigene Identität, Identität bestimmt durch `street` und `zipcode` Attribut
- Betrachtung von `Address`-Objekten als unveränderlich

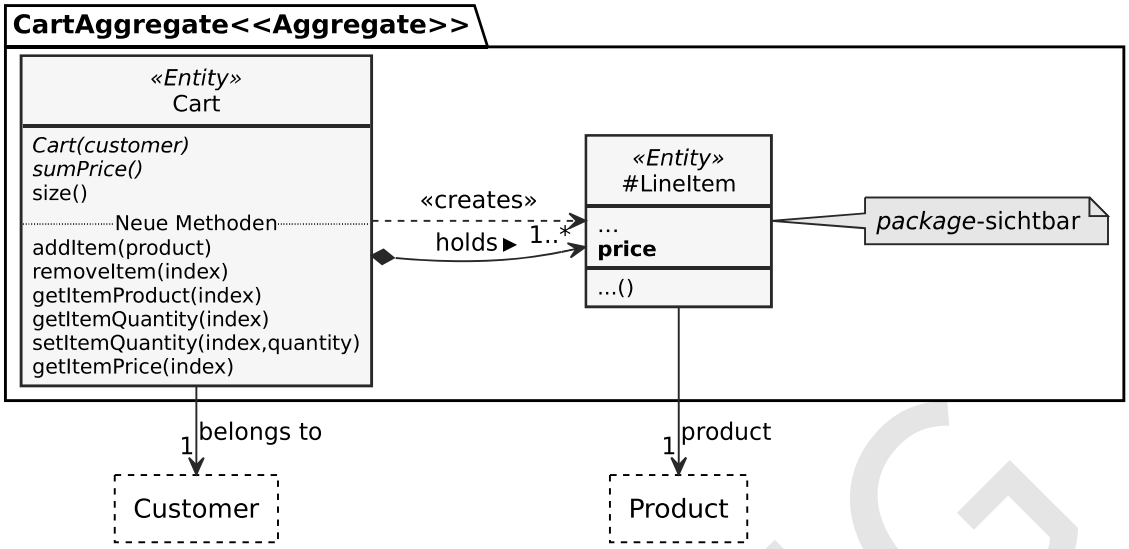
Hinweis: Beide Lösungen sind Valide, solange eine passende Begründung geliefert wird. Die bevorzugte Lösung klassifiziert `Address` als `<<ValueObject>>`.

c) Entwerfen Sie ein Aggregat für den Warenkorb (`Cart`), welches sicherstellt, dass der Bestellwert nie 3000€ übersteigen kann. Zeichnen Sie hierfür ein UML-Klassendiagramm, welches ein Paket *(package)* für das Aggregat sowie mindestens die Klassen `Cart` und `LineItem` enthält. Spezifizieren Sie für die Wurzel *(root)* des Aggregates alle neuen oder geänderten Methoden, um Bestelleinträge (`LineItem`) zu erstellen, zu bearbeiten und zu löschen. Markieren Sie alle Klassen, die nicht außerhalb des Aggregates sichtbar sein sollen, mit dem Prefix `#`. (5 Punkte)

Hinweis: Datentypen (wie zum Beispiel `int`, `long`, `Date`, `String`, ...) müssen nicht angegeben werden. Beschriften Sie jede Assoziation mit mindestens einem Rollennamen oder Assoziationsnamen sowie mindestens einer Kardinalität.

Musterlösung:




CartAggregate<<Aggregate>>
«Entity»
Cart
Cart(customer)
sumPrice()
size()
Neue Methoden
addItem(product)
removeItem(index)
getItemProduct(index)
getItemQuantity(index)
setItemQuantity(index,quantity)
getItemPrice(index)
«creates»
holds
1..*
«Entity»
#LineItem
...
price
...()
package-sichtbar
1
belongs to
Customer
1
product
Product

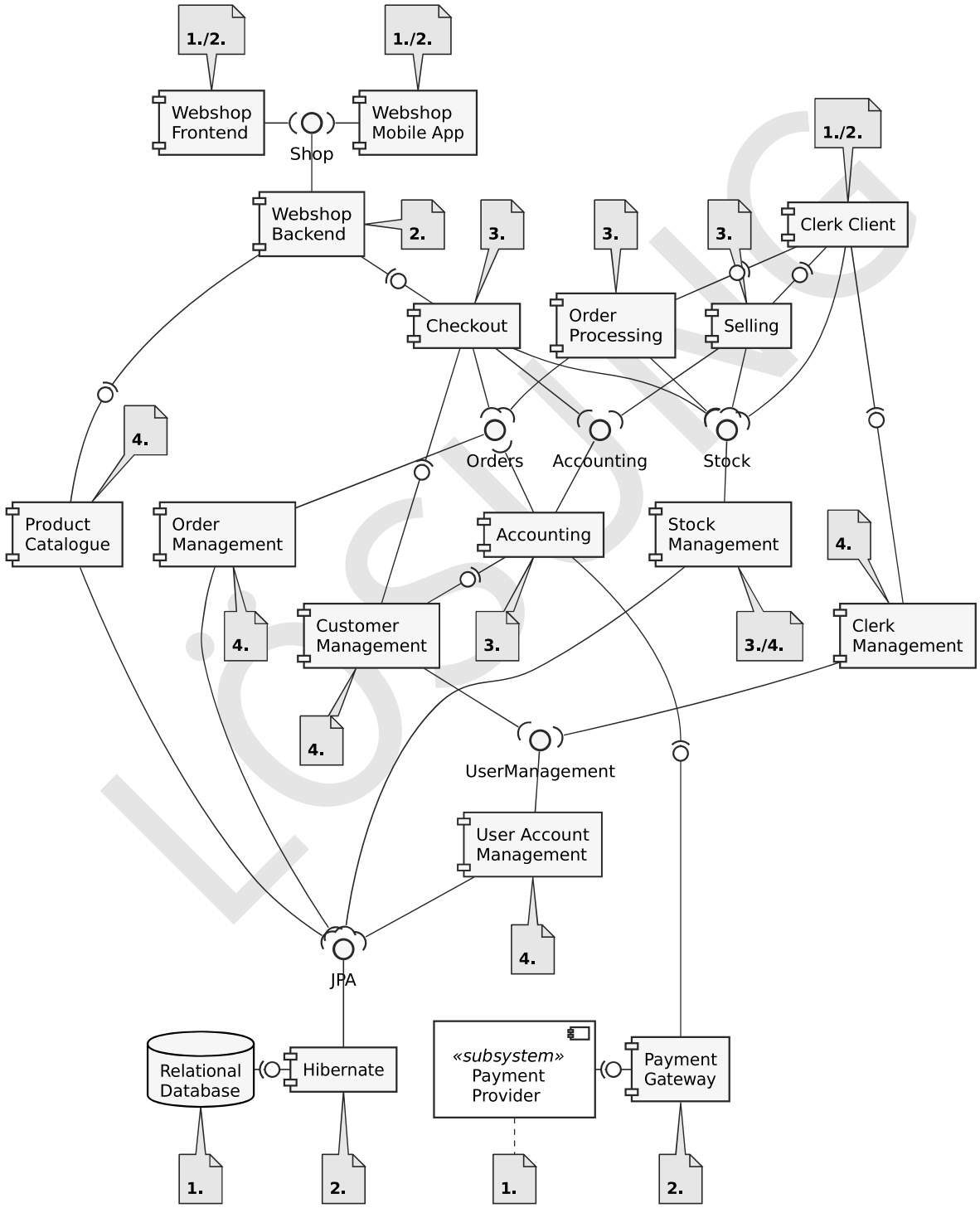
Aufgabe 5: Saubere Unternehmensarchitekturen *(16 Punkte)*

a) Tragen Sie in die Tabelle die vier Schichten der *sauberen* Architektur *(clean architecture)* so ein, dass sie zur angegebenen Richtung der Abhängigkeiten passen *(dependency rule)*. (2 Punkte)

Musterlösung: [redacted]

Schicht	Abhängigkeitsregel	Schichtenname
1.	↓	Framework & Drivers (External Interfaces)
2.	↓	Interface Adapter (Controllers)
3.	↓	Application Business Rules (Use Cases)
4.	↓	Enterprise Business Rules (Entities)

b) Das folgende Komponentendiagramm stellt die Softwarearchitektur des Softwaresystems des Computerfachgeschäfts dar, welches das Online-Geschäft *(online shop)*, den Direktverkauf *(selling)*, die Buchhaltung *(accounting)*, die Lagerverwaltung *(stock management)* sowie die Bestellungsabwicklung *(order processing)* unterstützt. Ordnen Sie nun jede Komponente einer der Schichten *sauberer* Architekturen zu, indem Sie die Nummer der Schicht in die mit „✎“ markierten Bereiche eintragen. (9 Punkte)



Musterlösung: [redacted]

Der Quelltext in Listing 1 stellt einen Ausschnitt der Implementierung der Kundenverwaltung (`CustomerManager`) im Online-Geschäft dar. Genauer werden hier sowohl angemeldete Kunden (`Customer`) wie auch unangemeldete Seitenbesucher (`Anonymous`) verwaltet. Die hierfür entwickelte Anwendung enthält jedoch einige Verstöße gegen Praktiken des sauberen Programmierens (Clean-Code-Prinzipien und Coding Conventions aus der Vorlesung).

c) Identifizieren Sie **fünf** Zeilen im Quelltextbeispiel (nächste Seite), die **unterschiedliche** Arten von Verstößen gegen die Praktiken des sauberen Programmierens (Clean-Code-Prinzipien und Coding Conventions aus der Vorlesung) enthalten. Benennen Sie jeweils die **Art** des Verstoßes mit der zugehörigen **Zeilennummer**. (5 Punkte)

Musterlösung: [redacted]

- Zeile 2: Attribut `email` nicht als `private` deklariert.

[redacted]

- Zeile 6: Formatierung, schwer zu lesen da Zeilenumbrüche fehlen.
- Zeile 11–15: *Principle of Least Surprise*, `oldPassword` wird nicht geprüft.
- Zeile 12: Formatierung, Auskommentierte Eingabevalidierung. (Boy Scout Rule)
- Zeile 20–23: `Anonymous` verstößt gegen *Liskov Substitution Principle* durch überschreiben von `changePassword`.
- Zeile 20–23: *Interface Segregation Principle*, `Anonymous` sollte keine `changePassword`-Methode brauchen.
- Zeile 22: *Principle of Least Surprise*: `changePassword` ändert den Zustand nie.

[redacted]

- Zeile 29,33: Verstoß gegen *Law of Demeter*, direkter Zugriff auf `email`.
- Zeile 33: Formatierung, schwer zu lesen da Zeilenumbrüche fehlen.

[redacted]

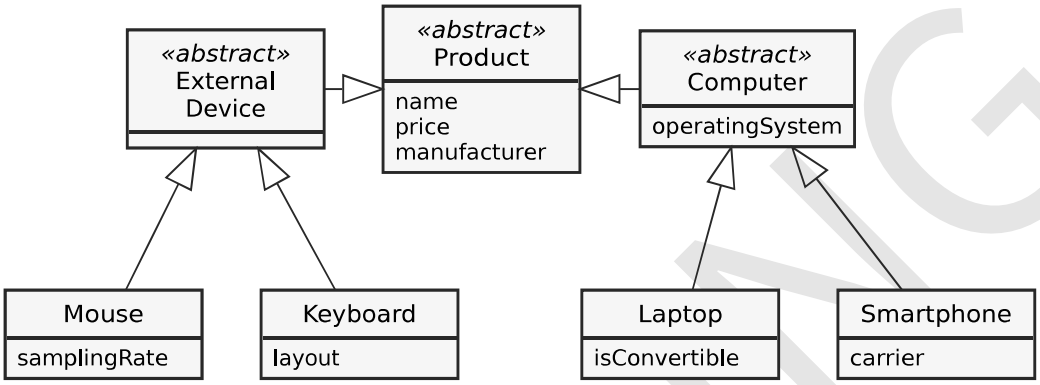
- Zeile 38: *Code Commenting*, unnötiger Kommentar beschreibt Code eins zu eins.
- Zeile 39: Verstoß gegen *Law of Demeter*, Zugriff auf `getName`.
- Zeile 42: Verstoß gegen *Single Responsibility Principle*, `authenticate`-Methode sollte in eigener Klasse implementiert werden.
- Zeile 44: Verstoß gegen *Open Closed Principle*, Prüfung auf Typungleichheit mit `Anonymous`.
- Zeile 44: Verstoß gegen *Law of Demeter*, Zugriff auf `getPassword`.

```java
public class Customer{
  final String email;
  private String password, name;

  public Customer(String email, String password; String name){
    if (email==null || email.isBlank()) throw new IllegalArgumentException();
    this.email = email;
    this.password = password;
    this.name = name;
  }
  public boolean changePassword(String oldPassword, String newPassword){
    //if (oldPassword.equals(password) && !newPassword.isBlank())
    password = newPassword;
    return true;
  }
  public String getName(){ return name; }
  public String getPassword(){ return password; }
}
/* Special customer for all users who are not yet logged in */
public class Anonymous extends Customer {
  public Anonymous() { super("anoymous@playtech.com", "", "Anoymous"); }
  public boolean changePassword(){ return false; }
}

public class CustomerManager {
  private HashMap<String, Customer> customers = new HashMap<>();

  public void save(Customer customer) {
    if (customer!=null && !customers.containsKey(customer.email))
      customers.put(customer.email, customer);
  }
  public Customer findBy(String email) {
    for (var customer : customers.values()) if (customer.email.equals(email))
      return customer;
    return null;
  }
  public List<Customer> collectByName(String name) {
    // Filter customers by name and collect result as list
    return customers.values().stream().filter(customer -> customer.getName().equals(name))
      .collect(Collectors.toList());
  }
  public boolean authenticate(String email, String password) {
    var customer = findBy(email);
    return customer!=null && !(customer instanceof Anonymous)
      && customer.getPassword().equals(password);
  }
  /* ... */
}
```

Listing 1: CustomerManager-Klasse mit Bad Smells

Aufgabe 6: Entwurfsmuster für Unternehmenssoftware *(16 Punkte)*

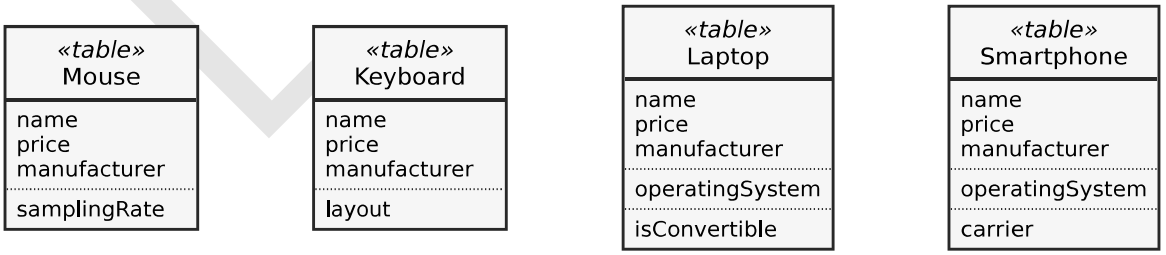
Das Domänenmodell des Computerfachgeschäfts enthält unter anderem die folgende Klassenhierarchie für die unterschiedlichen Produkte die Verkauft werden – Rechner *(computer)* und externe Geräte *(external device)*. Um diese Klassenhierarchie in einer Datenbank abzubilden, sollen Sie in dieser Aufgabe die unterschiedlichen Objekt-Relationalen Strukturmuster (*object-relational structural patterns*), welche in der Vorlesung behandelt wurden, anwenden.



a) Erstellen Sie ein Datenbankschema nach dem Strukturmuster *Concrete Table Inheritance*. Erstellen Sie hierzu ein UML-Klassendiagramm der nötigen Tabellen (dargestellt als UML-Klasse mit dem `<<table>>` Stereotyp) und ihrer Attribute. Führen Sie, falls nötig, neue Attribute ein, um *Primärschlüssel* und *Vererbung* darzustellen. (7 Punkte)

Hinweis: Datentypen (wie zum Beispiel `int`, `long`, `Date`, `String`, ...) müssen nicht angegeben werden.

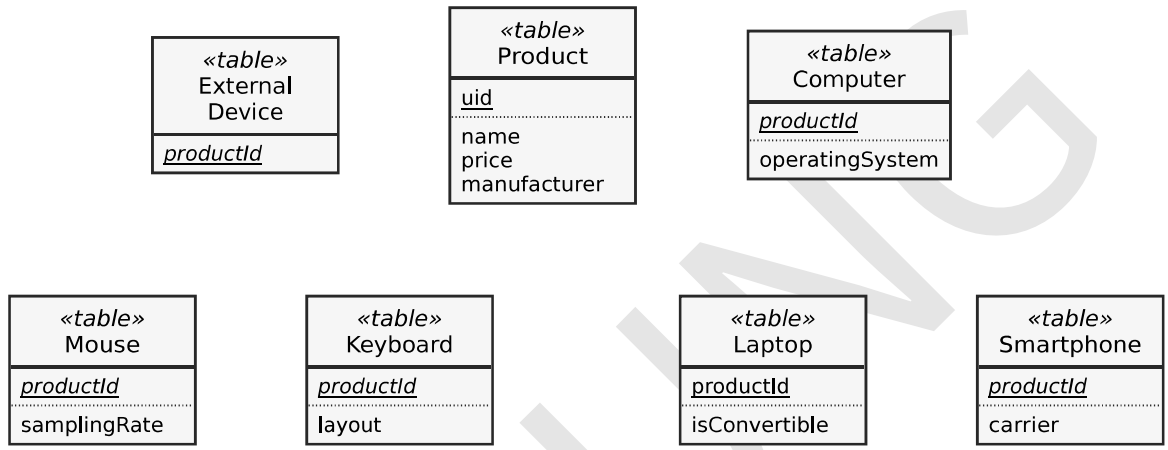
Musterlösung:



b) Erstellen Sie ein Datenbankschema nach dem Strukturmuster *Class Table Inheritance*. Erstellen Sie hierzu ein UML-Klassendiagramm der nötigen Tabellen (dargestellt als UML-Klasse mit dem `<<table>>` Stereotyp) und ihrer Attribute. Führen Sie, falls nötig, neue Attribute ein, um die *Primärschlüssel* und *Vererbung* darzustellen. (8 Punkte)

Hinweis: Datentypen (wie zum Beispiel `int`, `long`, `Date`, `String`, ...) müssen nicht angegeben werden.

Musterlösung:



c) Geben Sie stichpunktartig einen Vorteil und einen Nachteil der *Concrete Table Inheritance* gegenüber der *Class Table Inheritance* an. (1 Punkt)

Musterlösung: [redacted]

Vorteile:

- Benötigt keine Joins, um auf Datenzugreifen zu können. (Tabelle ist *self-contained*)
- Bessere Performanz, da auf Daten nur lokal zugegriffen wird.

Nachteile:

- Anfragen nach abstrakter Oberklasse, müssen stattdessen alle Unterklassen überprüfen.
- Änderungen von Attributen einer Oberklasse erfordert Änderung der Tabellen aller konkreten Unterklassen.

[redacted]